



Cumulus Linux 3.2.1

User Guide



Table of Contents

Introducing Cumulus Linux	13
What's New in Cumulus Linux 3.2.1	14
Open Source Contributions	14
Hardware Compatibility List	14
Quick Start Guide	14
Contents	15
Installation	16
Getting Started	17
Configuring Breakout Ports with Splitter Cables	20
Testing Cable Connectivity	20
Configuring Switch Ports	21
Configuring a Loopback Interface	24
Installation Management	25
Managing Cumulus Linux Disk Images	26
Contents	26
Installing a New Cumulus Linux Image	26
Upgrading Cumulus Linux	26
x86 vs ARM Switches	26
Reprovisioning the System (Restart Installer)	27
Uninstalling All Images and Removing the Configuration	27
Booting into Rescue Mode	28
Inspecting Image File Contents	29
Related Information	30
Installing a New Cumulus Linux Image	31
Upgrading Cumulus Linux	43
Using Snapshots	55
Adding and Updating Packages	62
Contents	62
Updating the Package Cache	62
Listing Available Packages	64
Adding a Package	65
Listing Installed Packages	65
Upgrading to Newer Versions of Installed Packages	66
Adding Packages from Another Repository	66
Related Information	68
Zero Touch Provisioning - ZTP	68
Contents	68
Zero Touch Provisioning Using a Local File	69
Zero Touch Provisioning Using USB (ZTP-USB)	69



Zero Touch Provisioning over DHCP	70
Writing ZTP Scripts	71
Testing and Debugging ZTP Scripts	74
Manually Using the ztp Command	78
Notes	79
System Configuration 79	
Network Command Line Utility	80
Contents	81
Installing NCLU	81
Getting Started	81
Adding More NCLU Users or Groups	85
Restarting the netd Service	86
Advanced Configuration	86
Setting Date and Time	87
Contents	87
Setting the Time Zone	88
Setting the Date and Time	89
Setting Time Using NTP	90
Specifying the NTP Source Interface	90
NTP Default Configuration	91
Related Information	92
Authentication, Authorization, and Accounting	92
SSH for Remote Access	92
User Accounts	95
Using sudo to Delegate Privileges	96
LDAP Authentication and Authorization	102
TACACS Plus	111
Netfilter - ACLs	121
Contents	123
Understanding Traffic Rules In Cumulus Linux	125
Installing and Managing ACL Rules with NCLU	133
Installing and Managing ACL Rules with cl-acltool	135
Installing Packet Filtering (ACL) Rules	136
Specifying which Policy Files to Install	138
Hardware Limitations on Number of Rules	138
Supported Rule Types	140
Common Examples	142
Example Scenario	147
Useful Links	151
Caveats and Errata	151
Default Cumulus Linux ACL Configuration	153
Managing Application Daemons	163
Contents	163
Using systemd and the systemctl Command	163



Identifying Active Listener Ports for IPv4 and IPv6	165
Identifying Daemons Currently Active or Stopped	166
Identifying Essential Services	170
Configuring switchd	171
Contents	171
The switchd File System	171
Configuring switchd Parameters	173
Restarting switchd	173
Power over Ethernet - PoE	174
Contents	174
How It Works	174
Configuring PoE	175
Troubleshooting PoE and PoE+	180
Configuring a Global Proxy	183
Related Information	184
Interface Configuration and Management	184
Contents	185
Basic Commands	185
ifupdown2 Interface Classes	186
Bringing All auto Interfaces Up or Down	187
Configuring a Loopback Interface	188
ifupdown Behavior with Child Interfaces	188
ifupdown2 Interface Dependencies	189
ifup Handling of Upper (Parent) Interfaces	192
Configuring IP Addresses	193
Specifying IP Address Scope	194
Purging Existing IP Addresses on an Interface	195
Specifying User Commands	195
Sourcing Interface File Snippets	196
Using Globs for Port Lists	197
Using Templates	198
Commenting out Mako Templates	199
Adding Descriptions to Interfaces	199
Caveats and Errata	200
Related Information	200
Layer 1 and Switch Port Attributes	201
Contents	201
Interface Types	201
Interface Settings	201
Configuring Breakout Ports	212
Logical Switch Port Limitations	214
Using ethtool to Configure Interfaces	215
Verification and Troubleshooting Commands	215
Caveats and Errata	217



Related Information	217
Buffer and Queue Management	218
Contents	218
Commands	219
Configuration Files	219
Configuring Traffic Marking through ACL Rules	221
Configuring Priority Flow Control	222
Configuring Link Pause	225
Configuring Explicit Congestion Notification	226
Caveats and Errata	227
Related Information	227
Configuring Hardware-enabled DDOS Protection	228
Configuring DHCP Relays	229
Contents	229
Configuring IPv4 DHCP Relays	230
Configuring IPv6 DHCP Relays	231
Configuring the DHCP Relay Service Manually (Advanced)	232
Troubleshooting the DHCP Relays	232
Configuring DHCP Servers	233
Contents	234
Configuring DHCP Server on Cumulus Linux Switches	234
Troubleshooting the Log from a DHCP Server	236
Layer One and Two	236
Spanning Tree and Rapid Spanning Tree	237
Contents	237
Supported Modes	237
Viewing Bridge and STP Status/Logs	238
Customizing Spanning Tree Protocol	242
Caveats and Errata	251
Related Information	251
Link Layer Discovery Protocol	251
Contents	252
Configuring LLDP	252
Example lldpcli Commands	252
Enabling the SNMP Subagent in LLDP	257
Caveats and Errata	257
Related Information	257
Prescriptive Topology Manager - PTM	257
Contents	258
Supported Features	258
Configuring PTM	258
Basic Topology Example	259
ptmd Scripts	260
Configuration Parameters	260



Bidirectional Forwarding Detection (BFD)	263
Checking Link State with Quagga	264
Using ptmd Service Commands	264
Using ptmctl Commands	265
Caveats and Errata	268
Related Information	268
Bonding - Link Aggregation	268
Contents	268
Hash Distribution	268
Creating a Bond	269
Example Configuration: Bonding 4 Slaves	270
Caveats and Errata	272
Related Information	272
Ethernet Bridging - VLANs	272
Contents	274
Creating a VLAN-aware Bridge	274
Creating a Traditional Mode Bridge	274
Configuring Bridge MAC Addresses	274
Configuring an SVI (Switch VLAN Interface)	274
Caveats and Errata	277
Related Information	277
VLAN-aware Bridge Mode for Large-scale Layer 2 Environments	277
Traditional Mode Bridges	288
VLAN Tagging	292
Multi-Chassis Link Aggregation - MLAG	300
Contents	301
MLAG Requirements	302
LACP and Dual-Connectedness	303
Configuring MLAG	304
Example MLAG Configuration	309
Checking the MLAG Configuration Status	320
Configuring MLAG with a Traditional Mode Bridge	321
Peer Link Interfaces and the protodown State	322
Monitoring Dual-Connected Peers	325
Configuring Layer 3 Routed Uplinks	326
IGMP Snooping with MLAG	327
Monitoring the Status of the clagd Service	327
MLAG Best Practices	328
STP Interoperability with MLAG	331
Troubleshooting MLAG	332
Caveats and Errata	333
LACP Bypass	333
Contents	333
Understanding the LACP Bypass All-active Mode	334
Configuring LACP Bypass	334



Virtual Router Redundancy - VRR	337
Contents	338
Configuring a VRR-enabled Network	338
ifplugd	340
IGMP and MLD Snooping	342
Contents	342
Configuring IGMP/MLD Querier	342
Disable IGMP and MLD Snooping	343
Debugging IGMP/MLD Snooping	344
Related Information	346
Network Virtualization	346
Caveats/Errata	347
Cut-through Mode	347
MTU Size for Virtual Network Interfaces	348
Useful Links	348
Integrating with VMware NSX	348
Contents	349
Getting Started	349
Bootstrapping the NSX Integration	350
Configuring the Transport Layer	355
Configuring the Logical Layer	356
Verifying the VXLAN Configuration	360
Troubleshooting VXLANs in NSX	360
Integrating Hardware VTEPs with Midokura MidoNet and OpenStack	361
Contents	361
Getting Started	362
Troubleshooting MidoNet and Cumulus VTEPs	370
Lightweight Network Virtualization - LNV Overview	378
Contents	378
Understanding LNV Concepts	379
Requirements	381
Sample LNV Configuration	382
Configuring the VLAN to VXLAN Mapping	388
Verifying the VLAN to VXLAN Mapping	390
Enabling and Managing Service Node and Registration Daemons	391
Configuring the Registration Node	392
Configuring the Service Node	394
Verification and Troubleshooting	395
Advanced LNV Usage	400
Related Information	406
LNV VXLAN Active-Active Mode	406
LNV Full Example	427
Static MAC Bindings with VXLAN	434
Contents	435



Requirements	435
Example VXLAN Configuration	435
Configuring the Static MAC Bindings VXLAN	436
Troubleshooting VXLANs in Cumulus Linux	437
Ethernet Virtual Private Network - EVPN	438
Contents	438
Installing the EVPN Package	439
Enabling Quagga	440
Configuring EVPN	441
Handling BUM Traffic	444
EVPN and VXLAN Active-Active Mode	444
Example Configuration	444
Testing Connectivity between Servers	450
Cumulus Linux Output Commands	451
BGP Output Commands	451
EVPN Output Commands	452
Troubleshooting EVPN	462
Caveats	462
VXLAN Hyperloop	463
Contents	464
Exiting a VXLAN with a Hyperloop	464
VXLAN Routing and Hyperloop Troubleshooting Matrix	467
Static VXLAN Tunnels	471
Contents	472
Requirements	472
Example Configuration	472
Configuring Static VXLAN Tunnels	473
Verifying the Configuration	477
Layer Three	477
Routing	478
Contents	478
Managing Static Routes	478
Supported Route Table Entries	480
Caveats and Errata	483
Related Information	484
Introduction to Routing Protocols	484
Contents	484
Defining Routing Protocols	485
Configuring Routing Protocols	485
Protocol Tuning	485
Network Topology	486
Contents	486
Clos Topologies	486
Over-Subscribed and Non-Blocking Configurations	487



Containing the Failure Domain	487
Load Balancing	488
Quagga Overview	488
Contents	488
Architecture	488
About zebra	489
Related Information	489
Configuring Quagga	489
Contents	489
Configuring Cumulus Quagga	489
Interface IP Addresses and VRFs	492
Using the Quagga vtysh Modal CLI	492
Reloading the Quagga Configuration	497
Related Information	498
Comparing NCLU and vtysh Commands	498
Open Shortest Path First - OSPF - Protocol	500
Contents	501
Scalability and Areas	501
Configuring OSPFv2	502
Scaling Tips	505
Unnumbered Interfaces	511
Applying a Route Map for Route Updates	512
ECMP	513
Topology Changes and OSPF Reconvergence	513
Debugging OSPF	514
Related Information	514
Open Shortest Path First v3 - OSPFv3 - Protocol	514
Contents	515
Configuring OSPFv3	515
Unnumbered Interfaces	515
Debugging OSPF	516
Related Information	516
Border Gateway Protocol - BGP	516
Contents	516
Autonomous System Number (ASN)	518
eBGP and iBGP	518
Route Reflectors	518
ECMP with BGP	519
Configuring BGP	520
Using BGP Unnumbered Interfaces	521
BGP add-path	526
Fast Convergence Design Considerations	530
Using Peer Groups to Simplify Configuration	531
Configuring BGP Dynamic Neighbors	531
Configuring BGP Peering Relationships across Switches	532



Configuring MD5-enabled BGP Neighbors	534
Configuring BGP TTL Security	536
Configuration Tips	538
Troubleshooting BGP	539
Enabling Read-only Mode	545
Applying a Route Map for Route Updates	545
Protocol Tuning	546
Caveats and Errata	548
Bidirectional Forwarding Detection - BFD	549
Contents	549
Using BFD Multihop Routed Paths	549
BFD Parameters	549
Configuring BFD	550
Troubleshooting BFD	555
Equal Cost Multipath Load Sharing - Hardware ECMP	556
Contents	556
Understanding Equal Cost Routing	556
Understanding ECMP Hashing	557
Resilient Hashing	560
Caveats and Errata	564
Redistribute Neighbor	564
Contents	565
Availability	565
Target Use Cases and Best Practices	565
How It Works	566
Configuration Steps	566
Known Limitations	570
Troubleshooting	571
Virtual Routing and Forwarding - VRF	574
Contents	575
Configuring VRF	576
Quagga Operation in a VRF	578
Example Commands to Show VRF Data	580
Using BGP Unnumbered Interfaces with VRF	586
Using DHCP with VRF	588
Using ping or traceroute	592
Caveats and Errata	592
Management VRF	593
Contents	593
Enabling Management VRF	594
OSPF and BGP	598
SNMP Traps Use eth0 Only	599
Using SSH within a Management VRF Context	599
Viewing the Routing Tables	599
Using the mgmt Interface Class	600



Management VRF and DNS	600
Incompatibility with cl-ns-mgmt	601
Protocol Independent Multicast - PIM	601
Contents	601
PIM Overview	602
PIM Sparse Mode (PIM-SM)	605
Configuration	609
Troubleshooting PIM	618
Caveats and Errata	622
Monitoring and Troubleshooting	623
Contents	624
Using the Serial Console	624
Configuring the Serial Console on ARM Switches	624
Configuring the Serial Console on x86 Switches	625
Getting General System Information	626
Diagnostics Using cl-support	626
Sending Log Files to a syslog Server	627
Local Logging	627
Enabling Remote syslog	628
Writing to syslog with Management VRF Enabled	629
Advanced Logging	629
Rate-limiting syslog Messages	630
Harmless syslog Error: Failed to reset devices.list	631
Next Steps	631
Single User Mode - Boot Recovery	632
Resource Diagnostics Using cl-resource-query	633
Monitoring System Hardware	634
Contents	635
Monitoring Hardware Using decode-syseeprom	635
Monitoring Hardware Using sensors	636
Monitoring Switch Hardware Using SNMP	638
Monitoring System Units Using smond	638
Keeping the Switch Alive Using the Hardware Watchdog	639
Related Information	639
Monitoring Virtual Device Counters	639
Contents	639
Sample VXLAN Statistics	640
Sample VLAN Statistics	641
Configuring the Counters in switchd	642
Caveats and Errata	643
Understanding the cl-support Output File	643
Troubleshooting Log Files	644
Troubleshooting the etc Directory	647
Troubleshooting Network Interfaces	658



Contents	658
Enabling Logging for Networking	658
Using ifquery to Validate and Debug Interface Configurations	659
Debugging Mako Template Errors	660
ifdown Cannot Find an Interface that Exists	661
Removing All References to a Child Interface	661
MTU Set on a Logical Interface Fails with Error: "Numerical result out of range"	662
Interpreting iproute2 batch Command Failures	662
Understanding the "RTNETLINK answers: Invalid argument" Error when Adding a Port to a Bridge	663
MLAG Peerlink Interface Drops Many Packets	663
Monitoring Interfaces and Transceivers Using ethtool	663
Network Troubleshooting	667
Contents	667
Checking Reachability Using ping	668
Printing Route Trace Using traceroute	668
Manipulating the System ARP Cache	669
Generating Traffic Using mz	669
Creating Counter ACL Rules	670
Configuring SPAN and ERSPAN	671
Monitoring Control Plane Traffic with tcpdump	679
Caveats and Errata	679
Related Information	680
Using NCLU to Troubleshoot Your Network Configuration	680
Monitoring System Statistics and Network Traffic with sFlow	682
Using netq to Troubleshoot the Network	685
SNMP Monitoring	696
Contents	696
Introduction to SNMP (Simple Network Management Protocol)	697
Configuring Ports for SNMP to Listen for Requests	697
Starting the SNMP Daemon	697
Configuring SNMP	698
snmpwalk a Switch from Another Linux Device	703
SNMP Traps	705
Supported MIBs	710
Using Nutanix Prism as a Monitoring Tool	713
Monitoring Best Practices	722
Contents	723
Overview	723
Hardware	724
System Data	726
Process Restart	728
Layer 1 Protocols and Interfaces	729
Layer 2 Protocols	736
Layer 3 Protocols	738



Logging	740
Protocols and Services	742
Device Management	742
Network Solutions	743
Data Center Host to ToR Architecture	744
Contents	744
Layer 2 - Architecture	744
Layer 3 Architecture	748
Network Virtualization	756
Cumulus Networks Services Demos	757
Contents	757
Reference Topology	758
Docker on Cumulus Linux	760
Contents	760
Installing Docker	760
Caveats and Errata	761
OpenStack Neutron ML2 and Cumulus Linux	762
Contents	763
Installing and Configuring the REST API	763
Installing and Configuring the Cumulus Networks Modular Layer 2 Mechanism Driver	764
Demo	765
Anycast Design Guide	765
Anycast Architecture	766
Anycast with TCP and UDP	767
Resilient Hashing	768
Applications for Anycast	770
Conclusion	771
Index	772

©2018 Cumulus Networks. All rights reserved

CUMULUS, the Cumulus Logo, CUMULUS NETWORKS, and the Rocket Turtle Logo (the “Marks”) are trademarks and service marks of Cumulus Networks, Inc. in the U.S. and other countries. You are not permitted to use the Marks without the prior written consent of Cumulus Networks. The registered trademark Linux® is used pursuant to a sublicense from LMI, the exclusive licensee of Linus Torvalds, owner of the mark on a worldwide basis. All other marks are used under fair use or license from their respective owners.



Introducing Cumulus Linux

Cumulus Linux is the networking industry's first full-featured Linux operating system. The [Debian Jessie](#)-based, networking-focused distribution runs on hardware produced by a [broad partner ecosystem](#), ensuring unmatched customer choice regarding silicon, optics, cables, and systems.

This user guide provides in-depth documentation covering installing Cumulus Linux, system configuration and management, network solutions, and monitoring and troubleshooting recommendations. In addition, the quick start guide provides an end-to-end setup process to get you started with Cumulus Linux.

This documentation is current as of May 1, 2017 for version 3.2.1. Please visit the [Cumulus Networks Web site](#) for the most up to date documentation.

Read the [release notes](#) for new features and known issues in this release.

What's New in Cumulus Linux 3.2.1

Cumulus Linux 3.2.1 adds these new features and platforms, including:

- **Network Command Line Utility:** We've improved the syntax so it's even easier for network operators to configure Cumulus Linux with [NCLU](#) (see page 80).
- **Platform Independent Multicast (PIM):** We've improved [multicast latency](#) (see page 601) on Mellanox switches.
- **Explicit Congestion Notification (ECN):** We've expanded support for [ECN](#) (see page 226) to Tomahawk switches.
- **New 100G platform:** Early access support for the [Edge-Core AS7412-32X](#), which uses the Mellanox Spectrum ASIC.

For further information regarding these new features, and for information regarding bug fixes and known issues present in this release, refer to the [product release notes](#).

Open Source Contributions

Cumulus Networks has forked various software projects, like CFEngine, Netdev and some Puppet Labs packages in order to implement various Cumulus Linux features. The forked code resides in the Cumulus Networks [GitHub repository](#).

Cumulus Networks developed and released as open source some new applications as well.

The list of open source projects is on the [open source software](#) page.

Hardware Compatibility List

You can find the most up to date hardware compatibility list (HCL) [here](#). Use the HCL to confirm that your switch model is supported by Cumulus Networks. The HCL is updated regularly, listing products by port configuration, manufacturer, and SKU part number.

Quick Start Guide

This quick start guide provides an end-to-end setup process for installing and running Cumulus Linux, as well as a collection of example commands for getting started once installation is complete.

Prerequisites

Prior intermediate Linux knowledge is assumed for this guide. You should be familiar with basic text editing, Unix file permissions, and process monitoring. A variety of text editors are pre-installed, including `vi` and `nano`.

You must have access to a Linux or UNIX shell. If you are running Windows, you should use a Linux environment like [Cygwin](#) as your command line tool for interacting with Cumulus Linux.



If you're a networking engineer but are unfamiliar with Linux concepts, refer to [this reference guide](#) for examples of the Cumulus Linux CLI and configuration options, and their equivalent Cisco Nexus 3000 NX-OS commands and settings for comparison. You can also [watch a series of short videos](#) introducing you to Linux in general and some Cumulus Linux-specific concepts in particular.

Contents

This chapter covers ...

- Installation (see page 16)
 - Upgrade to the Latest Version (see page 17)
- Getting Started (see page 17)
 - Login Credentials (see page 17)
 - Serial Console Management (see page 17)
 - Wired Ethernet Management (see page 17)
 - Configuring the Hostname and Timezone (see page 18)
 - Verifying the System Time (see page 19)
 - Installing the License (see page 19)
- Configuring Breakout Ports with Splitter Cables (see page 20)
- Testing Cable Connectivity (see page 20)
- Configuring Switch Ports (see page 21)
 - Layer 2 Port Configuration (see page 21)
 - Layer 3 Port Configuration (see page 23)
- Configuring a Loopback Interface (see page 24)



Installation

To install Cumulus Linux, you use [ONIE](#) (Open Network Install Environment), an extension to the traditional U-Boot software that allows for automatic discovery of a network installer image. This facilitates the ecosystem model of procuring switches, with a user's own choice of operating system loaded, such as Cumulus Linux.



If Cumulus Linux 3.0.0 or later is already installed on your switch, and you need to upgrade the software only, you can skip to [Upgrading Cumulus Linux \(see page 17\)](#) below.

The easiest way to install Cumulus Linux with ONIE is via local HTTP discovery:

1. If your host (like a laptop or server) is IPv6-enabled, make sure it is running a web server.
If the host is IPv4-enabled, make sure it is running DHCP as well as a web server.
2. [Download](#) the Cumulus Linux installation file to the root directory of the web server. Rename this file `onie-installer`.
3. Connect your host via Ethernet cable to the management Ethernet port of the switch.
4. Power on the switch. The switch downloads the ONIE image installer and boots it. You can watch the progress of the install in your terminal. After the installation finishes, the Cumulus Linux login prompt appears in the terminal window.



These steps describe a flexible unattended installation method. You should not need a console cable. A fresh install via ONIE using a local web server should generally complete in less than 10 minutes.

You have more options for installing Cumulus Linux with ONIE. Read [Installing a New Cumulus Linux Image \(see page 31\)](#) to install Cumulus Linux using ONIE in the following ways:

- DHCP/web server with and without DHCP options
- Web server without DHCP
- FTP or TFTP without a web server
- Local file
- USB

ONIE supports many other discovery mechanisms using USB (copy the installer to the root of the drive), DHCPv6 and DHCPv4, and image copy methods including HTTP, FTP, and TFTP. For more information on these discovery methods, refer to the [ONIE documentation](#).

After installing Cumulus Linux, you are ready to:

- Log in to Cumulus Linux on the switch.
- Install the Cumulus Linux license.
- Configure Cumulus Linux. This quick start guide provides instructions on configuring switch ports and a loopback interface.



Upgrade to the Latest Version

If you are running a Cumulus Linux version earlier than 3.0.0, you must perform a complete install, as described above (see page [25](#)). If you already have Cumulus Linux 3.0.0 or later installed on your switch, read [Upgrading Cumulus Linux \(see page 26\)](#) for considerations before start the process.

Getting Started

When bringing up Cumulus Linux for the first time, the management port makes a DHCPv4 request. To determine the IP address of the switch, you can cross reference the MAC address of the switch with your DHCP server. The MAC address should be located on the side of the switch or on the box in which the unit was shipped.

Login Credentials

The default installation includes one system account, `root`, with full system privileges, and one user account, `cumulus`, with `sudo` privileges. The `root` account password is set to null by default (which prohibits login), while the `cumulus` account is configured with this default password:

```
CumulusLinux!
```

In this quick start guide, you will use the `cumulus` account to configure Cumulus Linux.



For best security, you should change the default password (using the `passwd` command) before you configure Cumulus Linux on the switch.

All accounts except `root` are permitted remote SSH login; `sudo` may be used to grant a non-root account root-level access. Commands which change the system configuration require this elevated level of access.

For more information about sudo, read [Using sudo to Delegate Privileges \(see page 96\)](#).

Serial Console Management

Users are encouraged to perform management and configuration over the network, either in band or out of band (see page [46](#)). Use of the serial console is fully supported; however, many customers prefer the convenience of network-based management.

Typically, switches ship from the manufacturer with a mating DB9 serial cable. Switches with ONIE are always set to a 115200 baud rate.

Wired Ethernet Management

Switches supported in Cumulus Linux always contain at least one dedicated Ethernet management port, which is named `eth0`. This interface is geared specifically for out-of-band management use. The management interface uses DHCPv4 for addressing by default. You can set a static IP address with the Network Command Line Utility (NCLU).



ⓘ Example IP Configuration

Set the static IP address with the `interface` address and `interface gateway` NCLU commands:

```
cumulus@switch:~$ net add interface eth0 ip address 192.0.2.42  
/24  
cumulus@switch:~$ net add interface eth0 ip gateway 192.0.2.1  
cumulus@switch:~$ net pending  
cumulus@switch:~$ net commit
```

These commands produce the following snippet in the `/etc/network/interfaces` file:

```
auto eth0  
iface eth0  
    address 192.0.2.42/24  
    gateway 192.0.2.1
```

Configuring the Hostname and Timezone

To change the hostname, run `net add hostname`, which modifies both the `/etc/hostname` and `/etc/hosts` files with the desired hostname.

```
cumulus@switch:~$ net add hostname <hostname>  
cumulus@switch:~$ net pending  
cumulus@switch:~$ net commit
```



The command prompt in the terminal doesn't reflect the new hostname until you either log out of the switch or start a new shell.

To update the timezone, use the NTP interactive mode:

1. Run the following command in a terminal:

```
sudo dpkg-reconfigure tzdata
```

2. Follow the on screen menu options to select the correct location.



Programs that are already running (including log files), and users currently logged in, will not see timezone changes made with interactive mode.

Verifying the System Time

Before you install the license, you should verify that the switch's date and time are correct. You must [correct the time \(see page 87\)](#) if it is wrong, as the wrong date may have other impacts on the switch like an inability to synchronize with Puppet or return errors like this one after you restart `switchd`:

```
Warning: Unit file of switchd.service changed on disk, 'systemctl
daemon-reload' recommended.
```

Installing the License

Cumulus Linux is licensed on a per-instance basis. Each network system is fully operational, enabling any capability to be utilized on the switch with the exception of forwarding on switch panel ports. Only eth0 and console ports are activated on an unlicensed instance of Cumulus Linux. Enabling front panel ports requires a license.

You should have received a license key from Cumulus Networks or an authorized reseller. Here is a sample license key:

```
user@company.com|thequickbrownfoxjumpsoverthelazydog312
```

There are three ways to install the license onto the switch:

- Copy it from a local server. Create a text file with the license and copy it to a server accessible from the switch. On the switch, use the following command to transfer the file directly on the switch, then install the license file:

```
cumulus@switch:~$ scp user@my_server:/home/user/my_license_file.txt .
cumulus@switch:~$ sudo cl-license -i my_license_file.txt
```

- Copy the file to an HTTP server (not HTTPS), then reference the URL when you run `cl-license`:

```
cumulus@switch:~$ sudo cl-license -i <URL>
```



- Copy and paste the license key into the `cl-license -i` command:

```
cumulus@switch:~$ sudo cl-license -i  
<paste license key>  
^+d
```



You don't have to reboot the switch to activate the switch ports. Once you install the license, restart the `switchd` service. All front panel ports become active and show up as `swp1`, `swp2`, and so forth.

```
cumulus@switch:~$ sudo systemctl restart switchd.service
```



If a license is not installed on a Cumulus Linux switch, the `switchd` service does not start. Once you install the license, start `switchd` as described above.

Configuring Breakout Ports with Splitter Cables

If you are using 4x10G DAC or AOC cables, or want to break out 100G or 40G switch ports, configure the breakout ports. For more details, see [Layer 1 and Switch Port Attributes \(see page 212\)](#).

Testing Cable Connectivity

By default, all data plane ports (every Ethernet port except the management interface, `eth0`) are disabled.

To test cable connectivity, administratively enable a port:

```
cumulus@switch:~$ net add interface swp1  
cumulus@switch:~$ net pending  
cumulus@switch:~$ net commit
```

To administratively enable all physical ports, run the following command, where `swp1-52` represents a switch with switch ports numbered from `swp1` to `swp52`:

```
cumulus@switch:~$ net add interface swp1-52  
cumulus@switch:~$ net pending  
cumulus@switch:~$ net commit
```

To view link status, use `net show interface all`. The following examples show the output of ports in "admin down", "down" and "up" modes:



```
cumulus@switch:~$ net show interface all
      Name           Speed     MTU     Mode
Summary
-----
UP    lo            N/A      65536   Loopback   IP:
10.0.0.11/32, 127.0.0.1/8, ::1/128
UP    eth0          1G      1500    Mgmt      IP:
192.168.0.11/24(DHCP)
UP    swp1 (hypervisor_port_1) 1G      1500    Access/L2
Untagged: br0
UP    swp2          1G      1500    NotConfigured
ADMDN swp45         0M      1500    NotConfigured
ADMDN swp46         0M      1500    NotConfigured
ADMDN swp47         0M      1500    NotConfigured
ADMDN swp48         0M      1500    NotConfigured
ADMDN swp49         0M      1500    NotConfigured
ADMDN swp50         0M      1500    NotConfigured
UP    swp51          1G      1500    BondMember
Master: bond0(DN)
UP    blue          N/A      65536   NotConfigured
DN    bond0          N/A      1500    Bond      Bond
Members: swp51(UN)
UP    br0            N/A      1500    Bridge/L3   IP:
172.16.1.1/24

Untagged Members: swp1                         802.1
q Tag: Untagged                                STP:

RootSwitch(32768)
UP    red            N/A      65536   NotConfigured
ADMDN rename13        0M      1500    NotConfigured
ADMDN vagrant        0M      1500    NotConfigured
```

Configuring Switch Ports

Layer 2 Port Configuration

Cumulus Linux does not put all ports into a bridge by default. To create a bridge and configure one or more front panel ports as members of the bridge, use the following examples as guides.

Examples

① Example One

In the following configuration example, the front panel port swp1 is placed into a bridge called `bridge`. The NCLU commands are:

```
cumulus@switch:~$ net add bridge bridge ports swp1
cumulus@switch:~$ net pending
cumulus@switch:~$ net commit
```

The commands above produce the following `/etc/network/interfaces` snippet:

```
auto bridge
iface bridge
    bridge-ports swp1
    bridge-vlan-aware yes
```

① Example Two

A range of ports can be added in one command. For example, add swp1 through swp10, swp12, and swp14 through swp20 to `bridge`:

```
cumulus@switch:~$ net add bridge bridge ports swp1-10,12,14-20
cumulus@switch:~$ net pending
cumulus@switch:~$ net commit
```

This creates the following `/etc/network/interfaces` snippet:

```
auto bridge
iface bridge
    bridge-ports swp1 swp2 swp3 swp4 swp5 swp6 swp7 swp8
    swp9 swp10 swp12 swp14 swp15 swp16 swp17 swp18 swp19 swp20
    bridge-vlan-aware yes
```

To view the changes in the kernel, use the `brctl` command:

```
cumulus@switch:~$ brctl show
bridge name      bridge id          STP enabled    interfaces
bridge           8000.443839000004    yes           swp1
                                         swp2
```



A script is available to generate a configuration that [places all physical ports in a single bridge](#).

Layer 3 Port Configuration

The NCLU can also be used to configure a front panel port or bridge interface as a layer 3 port.

In the following configuration example, the front panel port swp1 is configured as a layer 3 access port:

```
cumulus@switch:~$ net add interface swp1 ip address 10.1.1.1/30
cumulus@switch:~$ net pending
cumulus@switch:~$ net commit
```

This creates the following /etc/network/interfaces snippet:

```
auto swp1
iface swp1
    address 10.1.1.1/30
```

To add an IP address to a bridge interface, it must be put into a VLAN interface:

```
cumulus@switch:~$ net add vlan 100 ip address 10.2.2.1/24
cumulus@switch:~$ net pending
cumulus@switch:~$ net commit
```

This creates the following /etc/network/interfaces snippet:

```
auto bridge
iface bridge
    bridge-vids 100
    bridge-vlan-aware yes

auto vlan100
iface vlan100
    address 192.168.10.1/24
    vlan-id 100
    vlan-raw-device bridge
```

To view the changes in the kernel use the `ip addr show` command:

```
cumulus@switch:~$ ip addr show
...
4. swp1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast
    master bridge state UP group default qlen 1000
        link/ether 44:38:39:00:6e:fe brd ff:ff:ff:ff:ff:ff
...
14: bridge: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue
    state UP group default
        link/ether 44:38:39:00:00:04 brd ff:ff:ff:ff:ff:ff
        inet6 fe80::4638:39ff:fe00:4/64 scope link
            valid_lft forever preferred_lft forever
...
...
```

Configuring a Loopback Interface

Cumulus Linux has a loopback preconfigured in `/etc/network/interfaces`. When the switch boots up, it has a loopback interface, called `lo`, which is up and assigned an IP address of 127.0.0.1.



The loopback interface `lo` must always be specified in `/etc/network/interfaces` and must always be up.

To see the status of the loopback interface (`lo`), use the `net show interface lo` command:

```
cumulus@switch:~$ net show interface lo
      Name      MAC                  Speed      MTU      Mode
---  -----  -----
UP   lo      00:00:00:00:00:00  N/A       65536  Loopback

IP Details
-----
IP:                         127.0.0.1/8, ::1/128
IP Neighbor(ARP) Entries:  0
```

Note that the loopback is up and is assigned an IP address of 127.0.0.1.

To add an IP address to a loopback interface, configure the `lo` interface with NCLU:

```
cumulus@switch:~$ net add loopback lo ip address 10.1.1.1/32
cumulus@switch:~$ net pending
cumulus@switch:~$ net commit
```



Multiple loopback addresses can be configured by adding additional address lines:

```
cumulus@switch:~$ net add loopback lo ip address 172.16.2.1/24
cumulus@switch:~$ net pending
cumulus@switch:~$ net commit
```

This creates the following snippet in /etc/network/interfaces:

```
auto lo
iface lo inet loopback
    address 10.1.1.1/32
    address 172.16.2.1/24
```



Installation Management

A Cumulus Linux switch can have only one image of the operating system installed. This section discusses installing new and updating existing Cumulus Linux disk images, and configuring those images with additional applications (via packages) if desired.

Zero touch provisioning is a way to quickly deploy and configure new switches in a large-scale environment.

Managing Cumulus Linux Disk Images

The Cumulus Linux operating system resides on a switch as a *disk image*. This section discusses how to manage the image, including installation and upgrading.

Contents

This chapter covers ...

- [Installing a New Cumulus Linux Image \(see page 26\)](#)
- [Upgrading Cumulus Linux \(see page 26\)](#)
- [x86 vs ARM Switches \(see page 26\)](#)
- [Reprovisioning the System \(Restart Installer\) \(see page 27\)](#)
- [Uninstalling All Images and Removing the Configuration \(see page 27\)](#)
- [Booting into Rescue Mode \(see page 28\)](#)
- [Inspecting Image File Contents \(see page 29\)](#)
- [Related Information \(see page 30\)](#)

Installing a New Cumulus Linux Image

For details, read the chapter, [Installing a New Cumulus Linux Image \(see page 31\)](#).

Upgrading Cumulus Linux

There are two ways you can upgrade Cumulus Linux:

- Upgrading only the changed packages, using `apt-get update` and `apt-get upgrade`. **This is the preferred method.**
- Perform a binary (full image) install of the new version, using [ONIE](#). This is used when moving between major versions or if you want to install a clean image.

The entire upgrade process is described in [Upgrading Cumulus Linux \(see page 43\)](#).

x86 vs ARM Switches

You can easily determine whether your switch is on an x86 or ARM platform by using the `uname -m` command.

For example, on an x86 platform, `uname -m` outputs `x86_64`:



```
cumulus@x86switch$ uname -m  
x86_64
```

While on an ARM platform, `uname -m` outputs `armv7l`:

```
cumulus@ARMswitch$ uname -m  
armv7l
```

You can also visit the HCL ([hardware compatibility list](#)) to look at your hardware to determine the processor type.

Reprovisioning the System (Restart Installer)

You can reprovision the system, wiping out the contents of the whole switch.

To initiate the provisioning and installation process, use `onie-select -i`:

```
cumulus@switch:~$ sudo onie-select -i  
WARNING:  
WARNING: Operating System install requested.  
WARNING: This will wipe out all system data.  
WARNING:  
Are you sure (y/N)? y  
Enabling install at next reboot...done.  
Reboot required to take effect.
```



A reboot is required for the reinstall to begin.



If you change your mind, you can cancel a pending reinstall operation by using `onie-select -c`:

```
cumulus@switch:~$ sudo onie-select -c  
Cancelling pending install at next reboot...done.
```

Uninstalling All Images and Removing the Configuration

To remove all installed images and configurations, returning the switch to its factory defaults, use `onie-select -k`:



```
cumulus@switch:~$ sudo onie-select -k
WARNING:
WARNING: Operating System uninstall requested.
WARNING: This will wipe out all system data.
WARNING:
Are you sure (y/N)? y
Enabling uninstall at next reboot...done.
Reboot required to take effect.
```



A reboot is required for the uninstall to begin.



If you change your mind you can cancel a pending uninstall operation by using `onie-select -c`

:

```
cumulus@switch:~$ sudo onie-select -c
 Cancelling pending uninstall at next reboot...done.
```

Booting into Rescue Mode

If your system becomes broken in some way, you may be able to correct things by booting into ONIE rescue mode. In rescue mode, the file systems are unmounted and you can use various Cumulus Linux utilities to try and fix the problem.

To reboot the system into the ONIE rescue mode, use `onie-select -r`:

```
cumulus@switch:~$ sudo onie-select -r
WARNING:
WARNING: Rescue boot requested.
WARNING:
Are you sure (y/N)? y
Enabling rescue at next reboot...done.
Reboot required to take effect.
```



A reboot is required to boot into rescue mode.



- If you change your mind you can cancel a pending rescue boot operation by using `onie-select -c`:

```
cumulus@switch:~$ sudo onie-select -c  
Cancelling pending rescue at next reboot...done.
```

Inspecting Image File Contents

The Cumulus Linux installation disk image file is executable. From a running switch, you can display the contents of the Cumulus Linux image file by passing the `info` option to the image file. For example, if the image file is called `onie-installer` and is located in `/var/lib/cumulus/installer`, you can get information about the disk image by running:

```
cumulus@switch:~$ sudo /var/lib/cumulus/installer/onie-installer info  
Verifying image checksum ... OK.  
Preparing image archive ... OK.  
Control File Contents  
=====  
Description: Cumulus Linux  
OS-Release: 2.1.0-0556262-201406101128-NB  
Architecture: amd64  
Date: Tue, 10 Jun 2014 11:44:28 -0700  
Installer-Version: 1.2  
Platforms: im_n29xx_t40n mlx_sx1400_i73612 dell_s6000_s1220  
Homepage: http://www.cumulusnetworks.com/  
  
Data Archive Contents  
=====  
 128 2014-06-10 18:44:26 file.list  
    44 2014-06-10 18:44:27 file.list.sha1  
104276331 2014-06-10 18:44:27 sysroot-internal.tar.gz  
    44 2014-06-10 18:44:27 sysroot-internal.tar.gz.sha1  
 5391348 2014-06-10 18:44:26 vmlinuz-initrd.tar.xz  
    44 2014-06-10 18:44:27 vmlinuz-initrd.tar.xz.sha1  
cumulus@switch:~$
```

You can also extract the contents of the image file by passing the `extract` option to the image file:



```
cumulus@switch:~$ sudo /var/lib/cumulus/installer/onie-installer
extract PATH
Verifying image checksum ... OK.
Preparing image archive ... OK.
file.list
file.list.sha1
sysroot-internal.tar.gz
sysroot-internal.tar.gz.sha1
vmlinuz-initrd.tar.xz
vmlinuz-initrd.tar.xz.sha1
Success: Image files extracted OK.
cumulus@switch:~$ sudo ls -l
total 107120
-rw-r--r-- 1 1063 3000 128 Jun 10 18:44 file.list
-rw-r--r-- 1 1063 3000 44 Jun 10 18:44 file.list.sha1
-rw-r--r-- 1 1063 3000 104276331 Jun 10 18:44 sysroot-internal.tar.gz
-rw-r--r-- 1 1063 3000 44 Jun 10 18:44 sysroot-internal.tar.gz.
sha1
-rw-r--r-- 1 1063 3000 5391348 Jun 10 18:44 vmlinuz-initrd.tar.xz
-rw-r--r-- 1 1063 3000 44 Jun 10 18:44 vmlinuz-initrd.tar.xz.
sha1
```

Finally, you can verify the contents of the image file by passing the `verify` option to the image file:

```
cumulus@switch:~$ sudo /var/lib/cumulus/installer/onie-installer
verify
Verifying image checksum ... OK.
Preparing image archive ... OK.
file.list
file.list.sha1
sysroot-internal.tar.gz
sysroot-internal.tar.gz.sha1
vmlinuz-initrd.tar.xz
vmlinuz-initrd.tar.xz.sha1
Success: Image files extracted OK.
cumulus@switch:~$ sudo ls -l
total 107120
-rw-r--r-- 1 1063 3000 128 Jun 10 18:44 file.list
-rw-r--r-- 1 1063 3000 44 Jun 10 18:44 file.list.sha1
-rw-r--r-- 1 1063 3000 104276331 Jun 10 18:44 sysroot-internal.tar.gz
-rw-r--r-- 1 1063 3000 44 Jun 10 18:44 sysroot-internal.tar.gz.
sha1
-rw-r--r-- 1 1063 3000 5391348 Jun 10 18:44 vmlinuz-initrd.tar.xz
-rw-r--r-- 1 1063 3000 44 Jun 10 18:44 vmlinuz-initrd.tar.xz.
sha1
```



Related Information

- Open Network Install Environment (ONIE) Home Page

Installing a New Cumulus Linux Image

Before you install Cumulus Linux, the switch can be in two different states:

- The switch has no image on it (so the switch is only running [ONIE](#)) or you desire or require a clean installation. In this case, you can install Cumulus Linux in one of the following ways, using:
 - DHCP/a web server with DHCP options (see page 32)
 - DHCP/a web server without DHCP options (see page 32)
 - A web server with no DHCP (see page 33)
 - FTP or TFTP without a web server (see page 33)
 - Local file installation (see page 34)
 - USB (see page 34)
- The switch already has Cumulus Linux installed on it, so you only need to [upgrade it](#) (see page 43)



[ONIE](#) is an open source project, equivalent to PXE on servers, that enables the installation of network operating systems (NOS) on bare metal switches.

Understanding these Examples

The sections in this chapter are ordered from the most repeatable to the least repeatable methods. For instance, DHCP can scale to hundreds of switch installs with zero manual input, compared to something like USB installs. Installing via USB is fine for a single switch here and there but is not scalable.

- You can name your Cumulus Linux installer binary using any of the [ONIE naming schemes mentioned here](#).
- In the examples below, [PLATFORM] can be any supported Cumulus Linux platform, such as `x86_64`, or `arm`.

Contents

This chapter covers ...

- Understanding these Examples (see page 31)
- Installing via a DHCP/Web Server Method with DHCP Options (see page 32)
- Installing via a DHCP/Web Server Method without DHCP Options (see page 32)
- Installing via a Web Server with no DHCP (see page 33)
- Installing via FTP or TFTP without a Web Server (see page 33)
- Installing via a Local File (see page 34)
- Installing via USB (see page 34)
 - Preparing for USB Installation (see page 34)
 - Instructions for x86 Platforms (see page 36)
 - Instructions for ARM Platforms (see page 39)

- Installing a New Image when Cumulus Linux Is already Installed (see page 42)
- Entering ONIE Mode from Cumulus Linux (see page 42)

Installing via a DHCP/Web Server Method with DHCP Options

Installing Cumulus Linux in this manner is as simple as setting up a DHCP/web server on your laptop and connecting the eth0 management port of the switch to your laptop.

Once you connect the cable, the installation proceeds as follows:

1. The bare metal switch boots up and asks for an address (DHCP request).
2. The DHCP server acknowledges and responds with DHCP option 114 and the location of the installation image.
3. ONIE downloads the Cumulus Linux binary, installs and reboots.
4. Success! You are now running Cumulus Linux.



The most common method is for you to send DHCP option 114 with the entire URL to the web server (this could be the same system). However, there are many other ways to use DHCP even if you don't have full control over DHCP. See the [ONIE user guide](#) for help.

Here's an example DHCP configuration with an [ISC DHCP server](#):

```
subnet 172.0.24.0 netmask 255.255.255.0 {
    range 172.0.24.20 172.0.24.200;
    option default-url = "http://172.0.24.14/onie-installer-[PLATFORM]" ;
}
```

Here's an example DHCP configuration with [dnsmasq](#) (static address assignment):

```
dhcp-host=sw4,192.168.100.14,6c:64:1a:00:03:ba,set:sw4
dhcp-option>tag:sw4,114,"http://roz.rtplab.test/onie-installer-
[PLATFORM]"
```

If you don't have a web server, you can use [this free Apache example](#).

Installing via a DHCP/Web Server Method without DHCP Options

If you have a laptop on same network and the switch can pull DHCP from the corporate network, but you cannot modify DHCP options (maybe it's controlled by another team), do the following:



1. Place the Cumulus Linux binary in a directory on the web server.
2. Run the `onie-nos-install` command manually, since DHCP options can't be modified:

```
ONIE:/ #onie-nos-install http://10.0.1.251/path/to/cumulus-
install-[PLATFORM].bin
```

Installing via a Web Server with no DHCP

Use the following method if your laptop is on the same network as the switch eth0 interface but no DHCP server is available.

One thing to note is ONIE is in *discovery mode*, so if you are setting a static IPv4 address for the eth0 management port, you need to disable discovery mode or else ONIE may get confused.

1. To disable discovery mode, run:

```
onie# onie-discovery-stop
```

or, on older ONIE versions if that command isn't supported:

```
onie# /etc/init.d/discover.sh stop
```

2. Assign a static address to eth0 via ONIE (using `ip addr add`):

```
ONIE:/ #ip addr add 10.0.1.252/24 dev eth0
```

3. Place the Cumulus Linux installer image in a directory on your web server.
4. Run the `onie-nos-install` command manually since there are no DHCP options:

```
ONIE:/ #onie-nos-install http://10.0.1.251/path/to/cumulus-
install-[PLATFORM].bin
```

Installing via FTP or TFTP without a Web Server

1. Set up DHCP or static addressing for eth0, as in the examples above.
2. If you are utilizing static addressing, disable ONIE discovery mode.
3. Place the Cumulus Linux installer image into a TFTP or FTP directory.



4. If you are not utilizing DHCP options, run one of the following commands (`tftp` for TFTP or `ftp` for FTP):

```
ONIE# onie-nos-install ftp://local-ftp-server/cumulus-install-[PLATFORM].bin  
ONIE# onie-nos-install tftp://local-tftp-server/cumulus-install-[PLATFORM].bin
```

Installing via a Local File

1. Set up DHCP or static addressing for eth0, as in the examples above.
2. If you are utilizing static addressing, disable ONIE discovery mode.
3. Use `scp` to copy the Cumulus Linux binary to the switch.
Note: Windows users can use [WinScp](#).
4. Run the following command:

```
ONIE# onie-nos-install /path/to/local/file/cumulus-install-[PLATFORM].bin
```

Installing via USB

Following the steps below produces a clean installation of Cumulus Linux. This wipes out all pre-existing configuration files that may be present on the switch. Instructions are offered for x86 and ARM platforms, and also cover the installation of a license after the software installation.



Make sure to [back up \(see page 43\)](#) any important configuration files that you may need to restore the configuration of your switch after the installation finishes.

Preparing for USB Installation

1. Download the appropriate Cumulus Linux image for your x86 or ARM platform from the [Cumulus Networks Downloads page](#).
2. Prepare your flash drive by formatting in one of the supported formats: FAT32, vFAT or EXT2.
Optional: Preparing a USB Drive inside Cumulus Linux



It is possible that you could severely damage your system with the following utilities, so please use caution when performing the actions below!



- a. Insert your flash drive into the USB port on the switch running Cumulus Linux and log in to the switch.
- b. Determine and note at which device your flash drive can be found by using output from `cat /proc/partitions` and `sudo fdisk -l [device]`. For example, `sudo fdisk -l /dev/sdb`.



These instructions assume your USB drive is the `/dev/sdb` device, which is typical if the USB stick was inserted after the machine was already booted. However, if the USB stick was plugged in during the boot process, it is possible the device could be `/dev/sda`. Make sure to modify the commands below to use the proper device for your USB drive!

- c. Create a new partition table on the device:

```
sudo parted /dev/sdb mklabel msdos
```



The `parted` utility should already be installed. However, if it is not, install it with:
`sudo apt-get install parted`

- d. Create a new partition on the device:

```
sudo parted /dev/sdb -a optimal mkpart primary 0% 100%
```

- e. Format the partition to your filesystem of choice using ONE of the examples below:

```
sudo mkfs.ext2 /dev/sdb1
sudo mkfs.msdos -F 32 /dev/sdb1
sudo mkfs.vfat /dev/sdb1
```



To use `mkfs.msdos` or `mkfs.vfat`, you need to install the `dosfstools` package from the [Debian software repositories](#) (step 3 here shows you how to add repositories from Debian), as they are not included by default.

- f. To continue installing Cumulus Linux, mount the USB drive in order to move files to it.

```
sudo mkdir /mnt/usb
sudo mount /dev/sdb1 /mnt/usb
```



3. Copy the image and license files over to the flash drive and rename the image file to:

- `onie-installer-x86_64`, if installing on an x86 platform
- `onie-installer-arm`, if installing on an ARM platform



You can also use any of the [ONIE naming schemes mentioned here](#).



When using a Mac or Windows computer to rename the installation file the file extension may still be present. Make sure to remove the file extension otherwise ONIE will not be able to detect the file!

4. Insert the USB stick into the switch, then continue with the appropriate instructions below for your x86 or ARM platform.

Instructions for x86 Platforms

Click to expand x86 instructions...

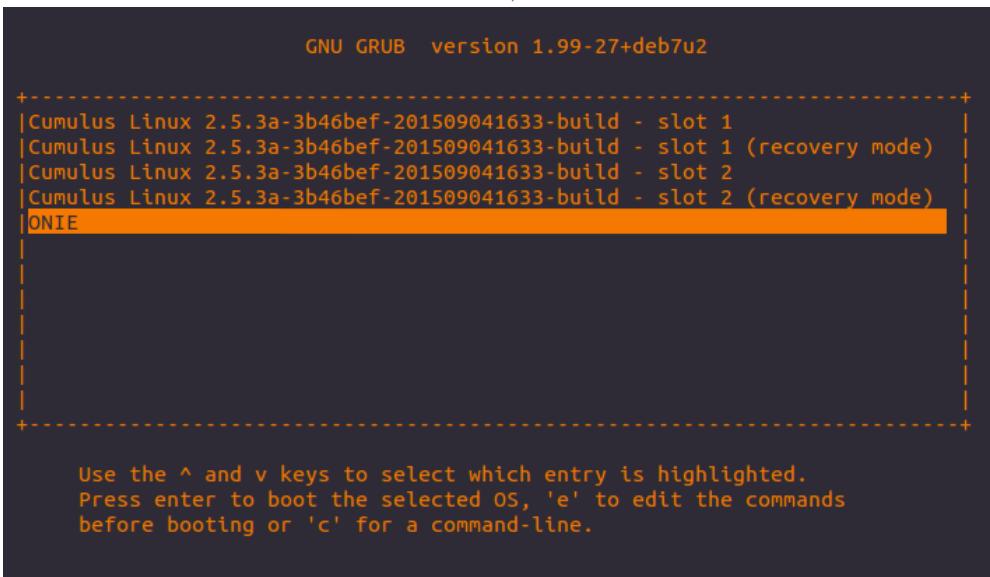
1. Prepare the switch for installation:

- If the switch is offline, connect to the console and power on the switch.
- If the switch is already online in Cumulus Linux, connect to the console and reboot the switch into the ONIE environment with the `sudo onie-select -i` command, followed by `sudo reboot`. Then skip to step 4 below.
- If the switch is already online in ONIE, use the `reboot` command.



SSH sessions to the switch get dropped after this step. To complete the remaining instructions, connect to the console of the switch. Cumulus Linux switches display their boot process to the console, so you need to monitor the console specifically to complete the next step.

2. Monitor the console and select the ONIE option from the first GRUB screen shown below.



GNU GRUB version 1.99-27+deb7u2

```
+-----+
| Cumulus Linux 2.5.3a-3b46bef-201509041633-build - slot 1
| Cumulus Linux 2.5.3a-3b46bef-201509041633-build - slot 1 (recovery mode)
| Cumulus Linux 2.5.3a-3b46bef-201509041633-build - slot 2
| Cumulus Linux 2.5.3a-3b46bef-201509041633-build - slot 2 (recovery mode)
| ONIE
+-----+
```

Use the ^ and v keys to select which entry is highlighted.
Press enter to boot the selected OS, 'e' to edit the commands
before booting or 'c' for a command-line.

3. Cumulus Linux on x86 uses GRUB chainloading to present a second GRUB menu specific to the ONIE partition. No action is necessary in this menu to select the default option *ONIE: Install OS*.



GNU GRUB version 2.02~beta2+e4a1fe391

```
+-----+
| *ONIE: Install OS
| ONIE: Rescue
| ONIE: Uninstall OS
| ONIE: Update ONIE
| ONIE: Embed ONIE
+-----+
```

Use the ^ and v keys to select which entry is highlighted.
Press enter to boot the selected OS, 'e' to edit the commands
before booting or 'c' for a command-line.



4. At this point, the USB drive should be automatically recognized and mounted. The image file should be located and automatic installation of Cumulus Linux should begin. Here is some sample output:

```
ONIE: OS Install Mode ...
Version : quanta_common_rangeley-2014.05.05-6919d98-201410171013
Build Date: 2014-10-17T10:13+0800
Info: Mounting kernel filesystems... done.
Info: Mounting LABEL=ONIE-BOOT on /mnt/onie-boot ...
initializing eth0...
scsi 6:0:0:0: Direct-Access SanDisk Cruzer Facet 1.26 PQ: 0
ANSI: 6
sd 6:0:0:0: [sdb] 31266816 512-byte logical blocks: (16.0 GB/14.
9 GiB)
sd 6:0:0:0: [sdb] Write Protect is off
sd 6:0:0:0: [sdb] Write cache: disabled, read cache: enabled,
doesn't support DPO or FUA
sd 6:0:0:0: [sdb] Attached SCSI disk
<...snip...
ONIE: Executing installer: file://dev/sdb1/onie-installer-x86_64
Verifying image checksum ... OK.
Preparing image archive ... OK.
Dumping image info...
Control File Contents
=====
Description: Cumulus Linux
OS-Release: 3.0.0-3b46bef-201509041633-build
Architecture: amd64
Date: Fri, 27 May 2016 17:10:30 -0700
Installer-Version: 1.2
Platforms: accton_as5712_54x accton_as6712_32x
mlx_sx1400_i73612 dell_s6000_s1220 dell_s4000_c2338
dell_s3000_c2338 cel_redstone_xp cel_smallstone_xp cel_pebble
quanta_panther quanta_ly8_rangeley quanta_ly6_rangeley
quanta_ly9_rangeley
Homepage: http://www.cumulusnetworks.com/
```

5. After installation completes, the switch automatically reboots into the newly installed instance of Cumulus Linux.
6. Determine and note at which device your flash drive can be found by using output from `cat /proc/partitions` and `sudo fdisk -l [device]`. For example, `sudo fdisk -l /dev/sdb`.



These instructions assume your USB drive is the `/dev/sdb` device, which is typical if the USB stick was inserted after the machine was already booted. However, if the USB stick was plugged in during the boot process, it is possible the device could be `/dev/sda`. Make sure to modify the commands below to use the proper device for your USB drive!



7. Create a mount point to mount the USB drive to:

```
sudo mkdir /mnt/mountpoint
```

8. Mount the USB drive to the newly created mount point:

```
sudo mount /dev/sdb1 /mnt/mountpoint
```

9. Install your license file with the `cl-license` command:

```
sudo cl-license -i /mnt/mountpoint/license.txt
```

10. Check that your license is installed with the `cl-license` command.

11. Reboot the switch to utilize the new license.

```
sudo reboot
```

Instructions for ARM Platforms

Click to expand ARM instructions...

1. Prepare the switch for installation:

- If the switch is offline, connect to the console and power on the switch.
- If the switch is already online in Cumulus Linux, connect to the console and reboot the switch into the ONIE environment with the `sudo onie-select -i` command, followed by `sudo reboot`. Then skip to step 4 below.
- If the switch is already online in ONIE, use the `reboot` command.



SSH sessions to the switch get dropped after this step. To complete the remaining instructions, connect to the console of the switch. Cumulus Linux switches display their boot process to the console, so you need to monitor the console specifically to complete the next step.



2. Interrupt the normal boot process before the countdown (shown below) completes. Press any key to stop the autobooting.

```
U-Boot 2013.01-00016-gddbf4a9-dirty (Feb 14 2014 - 16:30:46)
Accton: 1.4.0.5
CPU0: P2020, Version: 2.1, (0x80e20021)
Core: E500, Version: 5.1, (0x80211051)
Clock Configuration:
    CPU0:1200 MHz, CPU1:1200 MHz,
    CCB:600 MHz,
    DDR:400 MHz (800 MT/s data rate) (Asynchronous), LBC:37.500 MHz
L1: D-cache 32 kB enabled
    I-cache 32 kB enabled
<...snip...
USB: USB2513 hub OK
Hit any key to stop autoboot: 0
```

3. A command prompt appears, so you can run commands. Execute the following command:

```
run onie_bootcmd
```



4. At this point the USB drive should be automatically recognized and mounted. The image file should be located and automatic installation of Cumulus Linux should begin. Here is some sample output:

```
Loading Open Network Install Environment ...
Platform: arm-as4610_54p-r0
Version : 1.6.1.3
WARNING: adjusting available memory to 30000000
## Booting kernel from Legacy Image at ec040000 ...
    Image Name:    as6701_32x.1.6.1.3
    Image Type:    ARM Linux Multi-File Image (gzip compressed)
    Data Size:    4456555 Bytes = 4.3 MiB
    Load Address: 00000000
    Entry Point:  00000000
    Contents:
        Image 0: 3738543 Bytes = 3.6 MiB
        Image 1: 706440 Bytes = 689.9 KiB
        Image 2: 11555 Bytes = 11.3 KiB
    Verifying Checksum ... OK
## Loading init Ramdisk from multi component Legacy Image at
ec040000 ...
## Flattened Device Tree from multi component Image at EC040000
    Booting using the fdt at 0xec47d388
    Uncompressing Multi-File Image ... OK
    Loading Ramdisk to 2ff53000, end 2ffff788 ... OK
    Loading Device Tree to 03ffa000, end 03ffd22 ... OK
<...snip...
ONIE: Starting ONIE Service Discovery
ONIE: Executing installer: file://dev/sdb1/onie-installer-arm
Verifying image checksum ... OK.
Preparing image archive ... OK.
Dumping image info...
Control File Contents
=====
Description: Cumulus Linux
OS-Release: 3.0.0-3b46bef-201509041633-build
Architecture: arm
Date: Fri, 27 May 2016 17:08:35 -0700
Installer-Version: 1.2
Platforms: accton_as4600_54t, accton_as6701_32x, accton_5652,
accton_as5610_52x, dni_6448, dni_7448, dni_c7448n, cel_kennisis,
cel_redstone, cel_smallstone, cumulus_p2020, quanta_lb9,
quanta_ly2, quanta_ly2r, quanta_ly6_p2020
Homepage: http://www.cumulusnetworks.com/
```

5. After installation completes, the switch automatically reboots into the newly installed instance of Cumulus Linux.



6. Determine and note at which device your flash drive can be found by using output from `cat /proc/partitions` and `sudo fdisk -l [device]`. For example, `sudo fdisk -l /dev/sdb`.



These instructions assume your USB drive is the `/dev/sdb` device, which is typical if the USB stick was inserted after the machine was already booted. However, if the USB stick was plugged in during the boot process, it is possible the device could be `/dev/sda`. Make sure to modify the commands below to use the proper device for your USB drive!

7. Create a mount point to mount the USB drive to:

```
sudo mkdir /mnt/mountpoint
```

8. Mount the USB drive to the newly created mount point:

```
sudo mount /dev/sdb1 /mnt/mountpoint
```

9. Install your license file with the `cl-license` command:

```
sudo cl-license -i /mnt/mountpoint/license.txt
```

10. Check that your license is installed with the `cl-license` command.

11. Reboot the switch to utilize the new license.

```
sudo reboot
```

Installing a New Image when Cumulus Linux Is already Installed

Follow these upgrade steps for both major and minor releases, where:

- A major release upgrade is 2.X.X to 3.X.X (for example, 2.5.6 to 3.2.0)
- A minor release upgrade is X.2.X to X.3.X (for example, 3.1.2 to 3.2.0)

For more information, see [Upgrading Cumulus Linux \(see page 52\)](#).

Entering ONIE Mode from Cumulus Linux

If Cumulus Linux is already installed on the switch, you can enter ONIE mode in one of two ways, using:

- ONIE Recovery Mode to manually install an image from the ONIE prompt:

```
cumulus@switch:~$ sudo onie-select -r
cumulus@switch:~$ sudo reboot
```



- ONIE Install Mode to attempt to automatically discover the image from a DHCP server:

```
cumulus@switch:~$ sudo onie-select -i  
cumulus@switch:~$ sudo reboot
```

Upgrading Cumulus Linux

Cumulus Networks software melds the Linux host world with the networking devices world. Each world comes with its own paradigm on how to upgrade software. Before we discuss the various ways to upgrade Cumulus Linux switches, let's review the general considerations and strategies used to upgrade network devices and Linux hosts.

Contents

This chapter covers ...

- Upgrades: Comparing the Network Device Worldview vs. the Linux Host Worldview (see page 44)
 - Manual vs. Automated Configuration (see page 44)
 - Pre-deployment Testing of Production Environments (see page 44)
 - Locations of Configuration Data vs. Executables (see page 44)
 - Upgrade Procedure (see page 44)
 - Rollback Procedure (see page 45)
 - Third Party Packages (see page 45)
- Upgrading Cumulus Linux Devices: Strategies and Processes (see page 45)
 - Automated Configuration Is Preferred over Manual Configuration (see page 45)
 - Out-of-Band Management Is Worth the Investment (see page 46)
 - Pre-Deployment Testing of New Releases Is Advised and Enabled (see page 46)
 - Understanding the Locations of Configuration Data is Required for Successful Upgrades, Migration, and Backup (see page 46)
 - Upgrading Switches in an MLAG Pair (see page 50)
- Upgrading Cumulus Linux: Choosing between a Binary Install vs. Package Upgrade (see page 50)
 - Upgrading Using Package Installs (`apt-get update && apt-get upgrade`) (see page 51)
 - Upgrading via Binary Install (ONIE) (see page 52)
- Rolling Back a Cumulus Linux Installation (see page 52)
- Third Party Package Considerations (see page 53)
- Installation and Upgrade Workflow in Cumulus Linux 3.0 and Later (see page 53)
- Using Snapshots during Upgrades (see page 53)
- Caveats When Migrating Configuration Files Between Cumulus Linux 2.5.z and 3.0 and Later (see page 54)
- Using the Config File Migration Script to Identify and Move Files to Cumulus Linux 3.0 and Later (see page 54)
- Using Automation Tools to Back Up 2.5.z Configurations (see page 55)



Upgrades: Comparing the Network Device Worldview vs. the Linux Host Worldview

Manual vs. Automated Configuration

Historically, *network devices* were configured in place, and most network devices required customized configurations, which led predominantly to configuring the hardware manually. A lack of standardization between vendors, device types, and device roles hampered the development of APIs and automation tools. However, in the case of very large data centers, configurations became uniform and repeatable, and therefore scriptable. Some larger enterprises had to develop their own custom scripts to roll out data center network configurations. Virtually no industry-standard provisioning tools existed.

In contrast to data center network devices, *Linux hosts* in the data center number in the thousands and tend to have similar configurations. This increased scale led Linux sysadmins long ago to move to common tools to automate installation and configuration, since manually installing and configuring hosts did not work at the scale of a data center. Nearly all tasks are done via commonly available provisioning and orchestration tools.

Pre-deployment Testing of Production Environments

Historically, the cost of *network device* testing has been hampered by the cost of a single device. Setting up an appropriately sized lab topology can be very expensive. As a result, it is difficult to do comprehensive topology-based testing of a release before deploying it. Thus, many network admins cannot or will not do comprehensive system testing of a new release before deploying it.

Alternatively, the cost of a *Linux host* is cheap (or nearly free when using virtualization), so rigorous testing of a release before deploying it is not encumbered by budgeting concerns. Most sysadmins extensively test new releases in the complete application environment.

Locations of Configuration Data vs. Executables

Network devices generally separate configuration data from the executable code. On bootup, the executable code looks into a different file system and retrieves the configuration file or files, parses the text and uses that data to configure the software options for each software subsystem. The model is very centralized, with the executables generally being packaged together, and the configuration data following a set of rules that can be read by a centralized parser. Each vendor controls the configuration format for the entire box, since each vendor generally supports only their own software. This made sense since the platform was designed as an application-specific appliance.

Since a *Linux host* is a general purpose platform, with applications running on top of it, the locations of the files are much more distributed. Applications install and read their configuration data from text files usually stored in the /etc directory tree. Executables are generally stored in one of several *bin* directories, but the bin and etc directories are often on the same physical device. Since each *module* (application or executable) was often developed by a different organization and shared with the world, each module was responsible for its own configuration data format. Most applications are community supported, and while there are some generally accepted guiding principles on how their configuration data is formatted, no central authority exists to control or ensure compliance.

Upgrade Procedure

Both network admins and sysadmins generally plan upgrades only to gain new functionality or to get bug fixes when the workarounds become too onerous. The goal is to reduce the number of upgrades as much as possible.



The *network device* upgrade paradigm is to leave the configuration data in place, and *replace the executable files* either all at once from a single binary image or in large chunks (subsystems). A full release upgrade comes with risk due to unexpected behavior changes in subsystems where the admin did not anticipate or need changes.

The *Linux host* upgrade paradigm is to independently *upgrade a small list of packages* while leaving most of the OS untouched. Changing a small list of packages reduces the risk of unintended consequences. Usually upgrades are a "forward only" paradigm, where the sysadmins generally plan to move to the latest code within the same major release when needed. Every few years, when a new kernel train is released, a major upgrade is planned. A major upgrade involves wiping and replacing the entire OS and migrating configuration data.

Rollback Procedure

Even the most well planned and tested upgrades can result in unforeseen problems, and sometimes the best solution to new problems is to roll back to the previous state.

Since *network devices* clearly separate data and executables, generally the process is to *overwrite the new release executable* with the previously running executable. If the configuration was changed by the newer release, then you either have to manually back out or repair the changes, or restore from an already backed up configuration.

The *Linux host* scenario can be more complicated. There are three main approaches:

- Back out individual packages: If the problematic package is identified, the sysadmin can downgrade the affected package directly. In rare cases the configuration files may have to be restored from backup, or edited to back out any changes that were automatically made by the upgrade package.
- Flatten and rebuild: If the OS becomes unusable, you can use orchestration tools to reinstall the previous OS release from scratch and then automatically rebuild the configuration.
- Backup and restore: Another common strategy is to restore to a previous state via a backup captured before the upgrade.

Third Party Packages

Third party packages are rare in the *network device* world. Because the network OS is usually proprietary, third party packages are usually packaged by the network device vendor and upgrades of those packages is handled by the network device upgrade system.

Third party packages in *Linux host* world often use the same package system as the distribution into which it is to be installed (for example, Debian uses `apt-get`). Or the package may be compiled and installed by the sysadmin. Configuration and executable files generally follow the same filesystem hierarchy standards as other applications.

Upgrading Cumulus Linux Devices: Strategies and Processes

Because Cumulus Linux is both Linux *and* a network device, it has characteristics of both paradigms. The following describes the Cumulus Linux paradigm with respect to upgrade planning and execution.

Automated Configuration Is Preferred over Manual Configuration

Because Cumulus Linux *is* Linux, Cumulus Networks recommends that even with small networks or test labs, network admins should make the jump to deploy, provision, configure, and upgrade switches using automation from the very beginning. The small up front investment of time spent learning an orchestration tool, even to provision a small number of Cumulus Linux devices, will pay back dividends for a long time. The biggest gain is realized during the upgrade process, where the network admin can quickly upgrade dozens of devices in a repeatable manner.



Switches, like servers, should be treated like *cattle, not pets*.

Out-of-Band Management Is Worth the Investment

Because network devices are reachable via the IP addresses on the front panel ports, many network admins of small-to-medium sized networks use *in-band* networks to manage their switches. In this design, management traffic like SSH, SNMP, and console server connections use the same networks that regular network traffic traverses — there is no separation between the *management plane* and the *data plane*. Larger data centers create a separate *out-of-band* network with a completely separate subnet and reachability path to attach to the management ports — that is accessible via eth0 and the serial console.

This is a situation where smaller companies should learn from the big companies. A separate management network isn't free, but it is relatively cheap. With an inexpensive [Cumulus RMP](#) management switch, an inexpensive console server, and a separate cable path, up to 48 devices can be completely controlled via the out-of-band network in the case of a network emergency.

There are many scenarios where in-band networking can fail and leave the network admin waiting for someone to drive to the data center or remote site to connect directly to the console of a misconfigured or failing device. The cost of one outage would usually more than pay for the investment in a separate network. For even more security, attach remote-controllable power distribution units (PDUs) in each rack to the management network, so you can have complete control to remote power cycle every device in that rack.



However, if an out-of-band network is not available for you to upgrade, you can use the `dtach` tool instead to upgrade in band.

Pre-Deployment Testing of New Releases Is Advised and Enabled

White box switches and virtualization (Cumulus VX) bring the cost of networking devices down, so the ability for network admins to test their own procedures, configurations, applications, and network topology in an appropriately-sized lab topology becomes extremely affordable.

Understanding the Locations of Configuration Data is Required for Successful Upgrades, Migration, and Backup

As with other Linux distributions, the `/etc` directory is the primary location for all configuration data in Cumulus Linux. The following list is a likely set of files that should be backed up and migrated to a new release, but any file that has been changed would need to be examined. Cumulus Networks recommends you consider making the following files and directories part of a backup strategy.

Network Configuration Files

File Name and Location	Explanation	Cumulus Linux Documentation	Debian Documentation



File Name and Location	Explanation	Cumulus Linux Documentation	Debian Documentation
/etc/network/	Network configuration files, most notably /etc/network/interfaces and /etc/network/interfaces.d/	Interface Configuration and Management (see page 184)	wiki.debian.org/NetworkConfiguration
/etc/resolv.conf	DNS resolution	Not unique to Cumulus Linux: wiki.debian.org/NetworkConfiguration#The_resolv.conf_configuration_file	www.debian.org/doc/manuals/debian-reference/ch05.en.html
/etc/quagga/	Routing application (responsible for BGP and OSPF)	Quagga Overview (see page 488)	packages.debian.org/wheezy/quagga
/etc/hostname	Configuration file for the hostname of the switch	Quick Start Guide#ConfiguringtheHostnameandTimeZone (see page)	wiki.debian.org/HowTo/ChangeHostname
/etc/cumulus/acl/*	Netfilter configuration	Netfilter - ACLs (see page 121)	Access Control Lists in Linux
/etc/cumulus/ports.conf	Breakout cable configuration file	Layer 1 and Switch Port Attributes#ConfiguringBreakoutPorts (see page)	N/A; please read the guide on breakout cables
/etc/cumulus/switchd.conf	Switchd configuration	Configuring switchd (see page 171)	N/A; please read the guide on switchd configuration

Additional Commonly Used Files

File Name and Location	Explanation	Cumulus Linux Documentation	Debian Documentation

File Name and Location	Explanation	Cumulus Linux Documentation	Debian Documentation
/etc/motd	Message of the day	Not unique to Cumulus Linux	wiki.debian.org/motd#Wheezy
/etc/passwd	User account information	Not unique to Cumulus Linux	www.debian.org/doc/manuals/debian-reference/ch04.en.html
/etc/shadow	Secure user account information	Not unique to Cumulus Linux	www.debian.org/doc/manuals/debian-reference/ch04.en.html
/etc/group	Defines user groups on the switch	Not unique to Cumulus Linux	www.debian.org/doc/manuals/debian-reference/ch04.en.html
/etc/llpd.conf	Link Layer Discover Protocol (LLDP) daemon configuration	Link Layer Discovery Protocol (see page 251)	packages.debian.org/wheezy/llpd
/etc/llpd.d/	Configuration directory for llpd	Link Layer Discovery Protocol (see page 251)	packages.debian.org/wheezy/llpd
/etc/nsswitch.conf	Name Service Switch (NSS) configuration file	LDAP Authentication and Authorization (see page 102)	wiki.debian.org/LDAP/NSS
/etc/ssh/	SSH configuration files	SSH for Remote Access (see page 92)	wiki.debian.org/SSH
/etc/ldap/ldap.conf	Lightweight Directory Access Protocol configuration file	LDAP Authentication and Authorization (see page 102)	www.debian.org/doc/manuals/debian-reference/ch04.en.html

- If you are using the root user account, consider including /root/.
- If you have custom user accounts, consider including /home/<username>/.

Files That Should Never Be Migrated Between Versions or Boxes

File Name and Location	Explanation
/etc/adjtime	System clock adjustment data. NTP manages this automatically. It is incorrect when the switch hardware is replaced. Do not copy.



File Name and Location	Explanation
/etc/bcm.d/	Per-platform hardware configuration directory, created on first boot. Do not copy.
/etc/mlx/	Per-platform hardware configuration directory, created on first boot. Do not copy.
/etc/blkid.tab	Partition table. It should not be modified manually. Do not copy.
/etc/blkid.tab.old	A previous partition table; it should not be modified manually. Do not copy.
/etc/cumulus/init	Platform hardware-specific files. Do not copy.
/etc/default/clagd	Created and managed by <code>ifupdown2</code> . Do not copy.
/etc/default/grub	Grub <code>init</code> table; it should not be modified manually.
/etc/default/hwclock	Platform hardware-specific file. Created during first boot. Do not copy.
/etc/init	Platform initialization files. Do not copy.
/etc/init.d/	Platform initialization files. Do not copy.
/etc/fstab	Static info on filesystem. Do not copy.
/etc/image-release	System version data. Do not copy.
/etc/os-release	System version data. Do not copy.
/etc/lsb-release	System version data. Do not copy.
/etc/lvm/archive	Filesystem files. Do not copy.
/etc/lvm/backup	Filesystem files. Do not copy.
/etc/modules	Created during first boot. Do not copy.
/etc/modules-load.d/	Created during first boot. Do not copy.



File Name and Location	Explanation
/etc/sensors.d	Platform-specific sensor data. Created during first boot. Do not copy.
/root/.ansible	Ansible tmp files. Do not copy.
/home/cumulus/.ansible	Ansible tmp files. Do not copy.

Upgrading Switches in an MLAG Pair

If you have a pair of Cumulus Linux switches as part of an [MLAG \(multi-chassis link aggregation\) pair](#) (see [page 300](#)), you should only upgrade each switch when it is in the *secondary role*.



If you are upgrading from Cumulus Linux 2.y.z to Cumulus Linux 3.y.z, during the time when one switch in the pair is on Cumulus Linux 3.y.z and the other switch in the pair is on Cumulus Linux 2.y.z, a complete outage occurs on these switches and their associated network segments.

The upgrade path is as follows:

1. Upgrade Cumulus Linux on the switch already in the secondary role. This is the switch with the higher `cldgdpriority` value.
2. Set the switch in the secondary role into the primary role by setting its `cldgdpriority` to a value lower than the `cldgdpriority` setting on the switch in the primary role.

```
cumulus@switch:~$ sudo cldgctl priority VALUE
```

3. Upgrade the switch that just took on the secondary role.
4. Put that switch into the primary role again, if you so choose.

```
cumulus@switch:~$ sudo cldgctl priority VALUE
```

For more information about setting the priority, see [Understanding Switch Roles](#) (see [page 308](#)).

Upgrading Cumulus Linux: Choosing between a Binary Install vs. Package Upgrade

Network admins have two ways to upgrade Cumulus Linux:

- Upgrading only the changed packages, using `apt-get update` and `apt-get upgrade`. **This is the preferred method.**
- Performing a binary (full image) install of the new version, using ONIE. This is used when moving between major versions or if you want to install a clean image.



There are advantages and disadvantages to using these methods, which are outlined below.

Upgrading Using Package Installs (`apt-get update && apt-get upgrade`)

Pros:

- Configuration data stays in place while the packages are upgraded.
- Third-party apps stay in place.

Cons:

- This method works only if you are upgrading to a later minor release (like 3.1.x to 3.2.y), or to a later maintenance release from an earlier version of that minor release (for example, 2.5.2 to 2.5.5 or 3.0.0 to 3.0.1).
- Rollback is quite difficult and tedious.
- You can't choose the exact release version that you want to run.
- When you upgrade, you upgrade all packages to the latest available version.
- The upgrade process takes a while to complete, and various switch functions may be intermittently available during the upgrade.
- Some upgrade operations will terminate SSH sessions on the in-band (front panel) ports, leaving the user unable to monitor the upgrade process. As a workaround, use the [dtach tool](#).
- Just like the binary install method, you may have to reboot after the upgrade, lengthening the downtime.
- After you upgrade, user names and group names created by packages may be different on different switches, depending the configuration and package installation history.

To upgrade the switch by updating the packages:

1. Back up the configurations off the switch.
2. Fetch the latest update meta-data from the repository.

```
cumulus@switch$ sudo apt-get update
```

3. Upgrade all the packages to the latest distribution.

```
cumulus@switch$ sudo apt-get upgrade
```

4. Reboot the switch if the upgrade messages indicate that a system restart is required.

```
cumulus@switch$ sudo apt-get upgrade  
... upgrade messages here ...  
  
*** System restart required ***  
  
cumulus@switch$ sudo reboot
```

5. Verify correct operation with the old configurations on new version.



If you are curious about which packages will be upgraded, use `apt-get upgrade --dry-run`. This displays the list of packages that will be upgraded without upgrading anything.



After you successfully upgrade Cumulus Linux, you may notice some results that you may or may not have expected:

- `apt-get upgrade` always updates the operating system to the most current version, so if you are currently running Cumulus Linux 3.0.1 and run `apt-get upgrade` on that switch, the packages will get upgraded to the latest versions contained in the latest 3.y.z release.
- When you run `cat /etc/image-release`, the output still shows the version of Cumulus Linux from the last binary install. So if you installed Cumulus Linux 3.1.0 as a full image install and then upgraded to 3.2.0 using `apt-get upgrade`, the output from `/etc/image-release` still shows: `IMAGE_RELEASE=3.0.0`. To see the current version of all the Cumulus Linux packages running on the switch, use `dpkg --list` or `dpkg -l`.

Upgrading via Binary Install (ONIE)

Pros:

- You choose the exact version that you want to upgrade to.
- This is the only method for upgrading to a new major (X.0) version. For example, when you are upgrading from 2.5.5 to 3.0.

Cons:

- Configuration data must be moved to the new OS via ZTP while the OS is first booted, or soon afterwards via out-of-band management.
- Moving the configuration file can go wrong in various ways:
 - Identifying all the locations of config data is not always an easy task. See section above on [Understanding the Locations of Configuration Data \(see page 46\)](#).
 - Config file changes in the new version may cause merge conflicts that go undetected.
- If config files aren't restored correctly, the user may be unable to attach to the switch from in-band management. Hence, out-of-band connectivity (eth0 or console) is recommended.
- The installer takes a while to complete.
- Third-party apps must be reinstalled and reconfigured afterwards.

To upgrade the switch by running a binary install:

1. Back up the configurations off the switch.
2. Install the binary image, following the instructions at [Installing a New Cumulus Linux Image \(see page 31\)](#).
3. Restore the configuration files to the new version — ideally via automation.
4. Verify correct operation with the old configurations on the new version.
5. Reinstall third party apps and associated configurations.



Rolling Back a Cumulus Linux Installation

Rolling back to an earlier release after upgrading the packages on the switch follows the same procedure as described for the Linux host OS rollback above. There are three main strategies, and all require detailed planning and execution:

- Back out individual packages: If the problematic package is identified, the network admin can downgrade the affected package directly. In rare cases the configuration files may have to be restored from backup, or edited to back out any changes that were automatically made by the upgrade package.
- Flatten and rebuild: If the OS becomes unusable, you can use orchestration tools to reinstall the previous OS release from scratch and then automatically rebuild the configuration.
- Backup and restore: Another common strategy is to restore to a previous state via a backup captured before the upgrade.

Which method you employ is specific to your deployment strategy, so providing detailed steps for each scenario is outside the scope of this document.

Third Party Package Considerations

Note that if you install any third party apps on a Cumulus Linux switch, any configuration data will likely be installed into the `/etc` directory, but it is not guaranteed. It is the responsibility of the network admin to understand the behavior and config file information of any third party packages installed on a Cumulus Linux switch.

After you upgrade the OS using a full binary install, you will need to reinstall any third party packages or any Cumulus Linux add-on packages, such as `vxsnd` or `vxrd`.

Installation and Upgrade Workflow in Cumulus Linux 3.0 and Later

Beginning with version 3.0, Cumulus Linux completely embraces the Linux and Debian upgrade workflow. In this paradigm, a base image is installed using an installer, then any upgrades within that release train (major version, like 3.y.z) are done using `apt-get update && apt-get upgrade`. Any packages that have been changed since the base install get upgraded in place from the repository.

The huge advantage of this approach is that all switch configuration files remain untouched, or in rare cases merged (using the Debian merge function) during the package upgrade.

However, when upgrading a switch from a previous release train — for example, Cumulus Linux 2.5 — a mechanism is required to migrate the configuration files over to the new installation. This is the perfect opportunity to use automation and orchestration tools to backup the configuration files, examine them to verify correctness with the new version, and then to redeploy the configuration files on the new installation.

Using Snapshots during Upgrades

Snapshots (see page 55) can aid you when upgrading the switch operating system. Cumulus Linux takes two snapshots automatically during the upgrade, one right before you run `apt-get upgrade`, and one right after. This way, if something goes wrong with the upgrade, or you need to revert to the earlier version, you can roll back to the snapshot.



Caveats When Migrating Configuration Files Between Cumulus Linux 2.5.z and 3.0 and Later

Generally, the configuration files in Cumulus Linux 2.5.z should be able to migrate to version 3.0 or later without any problems, but there are some known issues listed below and there may be additional issues with a customer's particular setup.

Known caveats when migrating files from version 2.x to 3.0 or later:

- Some configuration files should never be migrated between versions or while replacing hardware. The [Files that Should Never be Migrated \(see page 48\)](#) table above contains a list of files that should never be migrated.
- `/etc/passwd` and `/etc/shadow` should not be migrated to the new version directly. The example below and the ansible script included with [Config File Migration Script](#) explicitly excludes these two files from the backup archive. The default password for the `cumulus` user must be changed, and any locally created users should be added to the new installation after the upgrade completes.
- `/etc/apt/sources.list` must be completely updated with a new 3.0 or later repository and repository structure. Repositories from Cumulus Linux 2.5 must be removed. If there are any custom repositories on the switch, they need to be migrated into the new `sources.list` file or the `sources.d/` directory.

Using the Config File Migration Script to Identify and Move Files to Cumulus Linux 3.0 and Later

You can use the [Config File Migration Script](#) with the `--backup` option to create a backup archive of configuration files in version 2.5, copy them off the box, then install them on the new version switch. Note that you need to follow the previous section about caveats when migrating configuration files.



You **cannot** use the Config File Migration Script to upgrade from Cumulus Linux 3.0.0 to 3.0.1. Use `apt-get` instead, as documented in the [release notes](#).

The following example excludes `/etc/apt`, `/etc/passwd` and `/etc/shadow` from the backup archive.

1. Back up the version 2.5.z files.

Optional: Use the Ansible playbook included with the [Config File Migration script](#) to automate the backup of all your Cumulus Linux 2.5 switches. See the section below on [Using Automation Tools to Backup Configurations \(see page 55\)](#) for more details.

```
# Make a temp dir
loc=$(mktemp -d)
# Create a backup archive to the temp dir
sudo ./config_file_changes --backup --backup_dir $loc --exclude
/etc/apt,/etc/passwd,/etc/shadow
# Copy the archive and log file to an external server
sudo scp -r $loc/* user@my_external_server:.
```

2. Install Cumulus Linux 3.0 or later onto the switch using ONIE (see page 31).
3. Reinstall the files from the config file archive to the newly installed switch.



```
# On the switch, copy the config file archive back from the
server:
scp user@my_external_server:PATH/SWITCHNAME-config-archive-
DATE_TIME.tar.gz .
# Untar the archive to the root of the box
sudo tar -C / -xvf SWITCHNAME-config-archive-DATE_TIME.tar.gz
```



Be aware that version 2.5.z configurations are not guaranteed to work in Cumulus Linux 3.0 or later. You should test the restoration and proper operation of the Cumulus Linux 2.5.z configuration in Cumulus Linux 3.0 or later on a non-production switch or in a Cumulus VX image, since every deployment is unique.

Using Automation Tools to Back Up 2.5.z Configurations

Adopting the use of orchestration tools like Ansible, Chef or Puppet for configuration management greatly increases the speed and accuracy of the next major upgrade; they also enable the quick swap of failed switch hardware. Included with the [Config Migration Script](#) is an Ansible playbook that can be used to create a backup archive of Cumulus Linux 2.5.z switch configuration files and to retrieve them to a central server — automating step 1 of the previous section for all deployed Cumulus Linux 2.5.z switches. This is a quick start on the road to setting up automated configuration and control for your deployment. For more details on integrating automation into your Cumulus Linux deployment, see the [Automation Solutions section](#) on cumulusnetworks.com.

Using Snapshots

Cumulus Linux supports the ability to take snapshots of the complete file system as well as the ability to roll back to a previous snapshot. Snapshots are performed automatically right before and after you upgrade Cumulus Linux and right before and after you commit a switch configuration using [NCLU \(see page 80\)](#). In addition, you can take a snapshot at any time. You can roll back the entire file system to a specific snapshot or just retrieve specific files.

The primary snapshot components are:

- **btrfs** — an underlying file system in Cumulus Linux, which supports snapshots.
- **snapper** — a userspace utility to create and manage snapshots on demand as well as taking snapshots automatically before and after running `apt-get upgrade|install|remove|dist-upgrade`. You can use `snapper` to roll back to earlier snapshots, view existing snapshots, or delete one or more snapshots.
- [NCLU \(see page 80\)](#) — takes snapshots automatically before and after committing network configurations. You can use NCLU to roll back to earlier snapshots, view existing snapshots, or delete one or more snapshots.

Contents

This chapter covers ...

- [Installing the Snapshot Package \(see page 56\)](#)
- [Taking and Managing Snapshots \(see page 56\)](#)



- Viewing Available Snapshots (see page 56)
- Viewing Differences between Snapshots (see page 57)
- Deleting Snapshots (see page 58)
- Rolling Back to Earlier Snapshots (see page 60)
- Configuring Automatic Time-based Snapshots (see page 60)
- Caveats and Errata (see page 61)
 - root Partition Mounted Multiple Times (see page 61)

Installing the Snapshot Package

If you're upgrading from a version of Cumulus Linux earlier than version 3.2, you need to install the `cumulus-snapshot` package before you can use snapshots.

```
cumulus@switch:~$ sudo apt-get update
cumulus@switch:~$ sudo apt-get install cumulus-snapshot
cumulus@switch:~$ sudo apt-get upgrade
```

Taking and Managing Snapshots

As described above, snapshots are taken automatically:

- Before and after you update your switch configuration by running `net commit`, via NCLU.
- Before and after you update Cumulus Linux by running `apt-get upgrade|install|remove|dist-upgrade`, via `snapper`.

You can also take snapshots as needed using the `snapper` utility. Run:

```
cumulus@switch:~$ sudo snapper create -d SNAPSHOT_NAME
```

For more information about using `snapper`, run `snapper --help` or `man snapper(8)`.

Viewing Available Snapshots

You can use both NCLU and `snapper` to view available snapshots on the switch.

```
cumulus@switch:~$ net show commit history
# Date                                Description
--- -----
20 Thu 01 Dec 2016 01:43:29 AM UTC  nclu pre  'net commit' (user
cumulus)
21 Thu 01 Dec 2016 01:43:31 AM UTC  nclu post 'net commit' (user
cumulus)
22 Thu 01 Dec 2016 01:44:18 AM UTC  nclu pre  '20 rollback' (user
cumulus)
```



```
23 Thu 01 Dec 2016 01:44:18 AM UTC nclu post '20 rollback' (user
cumulus)
24 Thu 01 Dec 2016 01:44:22 AM UTC nclu pre '22 rollback' (user
cumulus)
31 Fri 02 Dec 2016 12:18:08 AM UTC nclu pre 'ACL' (user cumulus)
32 Fri 02 Dec 2016 12:18:10 AM UTC nclu post 'ACL' (user cumulus)
```

However, `net show commit history` only displays snapshots taken when you update your switch configuration. It does not list any snapshots taken directly with `snapper`. To see all the snapshots on the switch, run:

```
cumulus@switch:~$ sudo snapper list
Type    | # | Pre # | Date                                     | User |
Cleanup |   | Description                                | Userdata
-----+---+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+
single | 0 |           |                               | root
|       | current          |                               |
single | 1 |           | Sat 24 Sep 2016 01:45:36 AM UTC | root
|       | first root filesystem |                               |
pre    | 20 |           | Thu 01 Dec 2016 01:43:29 AM UTC | root |
number | nclu pre  'net commit' (user cumulus) | |
post   | 21 | 20        | Thu 01 Dec 2016 01:43:31 AM UTC | root |
number | nclu post 'net commit' (user cumulus) | |
pre    | 22 |           | Thu 01 Dec 2016 01:44:18 AM UTC | root |
number | nclu pre  '20 rollback' (user cumulus) | |
post   | 23 | 22        | Thu 01 Dec 2016 01:44:18 AM UTC | root |
number | nclu post '20 rollback' (user cumulus) | |
single | 26 |           | Thu 01 Dec 2016 11:23:06 PM UTC | root
|       | test_snapshot |                               |
pre    | 29 |           | Thu 01 Dec 2016 11:55:16 PM UTC | root |
number | pre-apt          | important=yes
post   | 30 | 29        | Thu 01 Dec 2016 11:55:21 PM UTC | root |
number | post-apt          | important=yes
pre    | 31 |           | Fri 02 Dec 2016 12:18:08 AM UTC | root |
number | nclu pre  'ACL' (user cumulus) | |
post   | 32 | 31        | Fri 02 Dec 2016 12:18:10 AM UTC | root |
number | nclu post 'ACL' (user cumulus) | |
```

Viewing Differences between Snapshots

To see a line by line comparison of changes between two snapshots, run:

```
cumulus@switch:~$ sudo snapper diff 20..21
--- /.snapshots/20/snapshot/etc/cumulus/acl/policy.d/50_nclu_acl.
rules      2016-11-30 23:00:42.675092103 +0000
+++ /.snapshots/21/snapshot/etc/cumulus/acl/policy.d/50_nclu_acl.
rules      2016-12-01 01:43:30.029171289 +0000
```

```

@@ -1,7 +0,0 @@
-[iptables]
-# control-plane: acl ipv4 EXAMPLE1 inbound
--A INPUT --in-interface swp+ -j ACCEPT -p tcp -s 10.0.0.11/32 -d
10.0.0.12/32 --dport 110
-
-# swp1: acl ipv4 EXAMPLE1 inbound
--A FORWARD --in-interface swp1 --out-interface swp2 -j ACCEPT -p tcp
-s 10.0.0.11/32 -d 10.0.0.12/32 --dport 110
-
--- /.snapshots/20/snapshot/var/lib/cumulus/nclu/nclu_acl.conf
2016-11-30 23:00:18.030079000 +0000
+++ /.snapshots/21/snapshot/var/lib/cumulus/nclu/nclu_acl.conf
2016-12-01 00:23:10.096136000 +0000
@@ -1,8 +1,3 @@
-acl ipv4 EXAMPLE1 priority 10 accept tcp 10.0.0.11/32 10.0.0.12/32
pop3 outbound-interface swp2

-control-plane
-    acl ipv4 EXAMPLE1 inbound

-iface swp1
-    acl ipv4 EXAMPLE1 inbound

```

You can view the diff for a single file by specifying the name in the command:

```

cumulus@switch:~$ sudo snapper diff 20..21 /var/lib/cumulus/nclu
/nclu_acl.conf
--- /.snapshots/20/snapshot/var/lib/cumulus/nclu/nclu_acl.conf
2016-11-30 23:00:18.030079000 +0000
+++ /.snapshots/21/snapshot/var/lib/cumulus/nclu/nclu_acl.conf
2016-12-01 00:23:10.096136000 +0000
@@ -1,8 +1,3 @@
-acl ipv4 EXAMPLE1 priority 10 accept tcp 10.0.0.11/32 10.0.0.12/32
pop3 outbound-interface swp2

-control-plane
-    acl ipv4 EXAMPLE1 inbound

-iface swp1
-    acl ipv4 EXAMPLE1 inbound

```

For a higher level view, displaying the names of changed/added/deleted files only, run:

```

cumulus@switch:~$ sudo snapper status 20..21
c..... /etc/cumulus/acl/policy.d/50_nclu_acl.rules
c..... /var/lib/cumulus/nclu/nclu_acl.conf

```



Deleting Snapshots

You can remove one or more snapshots using both NCLU and snapper.



Take care when deleting a snapshot, as you cannot restore it once it's been deleted.

To remove a single snapshot or a range of them created with NCLU, run:

```
cumulus@switch:~$ net commit delete SNAPSHOT|SNAPSHOT1-SNAPSHOT2
```

To remove a single snapshot or a range of snapshots using `snapper`, run:

```
cumulus@switch:~$ sudo snapper delete SNAPSHOT|SNAPSHOT1-SNAPSHOT2
```



Snapshot 0 is the running configuration. You can't roll back to it or delete it. However, you can take a snapshot of it.

Snapshot 1 is the root file system.

The `snapper` utility preserves a number of snapshots, and automatically deletes older snapshots once the limit is reached. It does this in two ways.

By default, `snapper` preserves 10 snapshots that are labeled *important*. A snapshot is labeled important if it was created when you run `apt-get`. To change this number, run:

```
cumulus@switch:~$ sudo snapper set-config NUMBER_LIMIT_IMPORTANT=<NUM>
```



You should always make `NUMBER_LIMIT_IMPORTANT` an even number since two snapshots are always taken before and after an upgrade. This does not apply to `NUMBER_LIMIT`, described next.

`snapper` also deletes unlabeled snapshots. The default number of snapshots `snapper` preserves is 5. To change this number, run:

```
cumulus@switch:~$ sudo snapper set-config NUMBER_LIMIT=<NUM>
```

Also, you can prevent snapshots from being taken automatically before and running `apt-get upgrade|install|remove|dist-upgrade`. Edit `/etc/cumulus/apt-snapshot.conf` and set:

```
APT_SNAPSHOT_ENABLE=no
```



Rolling Back to Earlier Snapshots

If you need to restore Cumulus Linux to an earlier state, you can roll back to an older snapshot.

For a snapshot created with NCLU, you can revert to a specific snapshot listed in the output from `net show commit history`, or you can revert to the previous snapshot by specifying `last` when you run:

```
cumulus@switch:~$ net rollback SNAPSHOT_NUMBER|last
```

For any snapshot on the switch, you can use `snapper` to roll back to a specific snapshot. When running `snapper rollback`, you must reboot the switch for the rollback to complete:

```
cumulus@switch:~$ sudo snapper rollback SNAPSHOT_NUMBER  
cumulus@switch:~$ sudo reboot
```

You can also revert to an earlier version of a specific file instead of rolling back the whole file system:

```
cumulus@switch:~$ sudo snapper undochange 31..32 /etc/cumulus/acl  
/policy.d/50_nclu_acl.rules
```



You can also copy the file directly from the snapshot directory:

```
cumulus@switch:~$ cp /.snapshots/32/snapshot/etc/cumulus/acl  
/policy.d/50_nclu_acl.rules /etc/cumulus/acl/policy.d/
```

Configuring Automatic Time-based Snapshots

You can configure Cumulus Linux to take hourly snapshots. You need to enable `TIMELINE_CREATE` in the `snapper` configuration:

```
cumulus@switch:~$ sudo snapper set-config TIMELINE_CREATE=yes  
cumulus@switch:~$ sudo snapper get-  
config  
Key | Value  
-----+-----  
ALLOW_GROUPS |  
ALLOW_USERS |  
BACKGROUND_COMPARISON | yes  
EMPTY_PRE_POST_CLEANUP | yes  
EMPTY_PRE_POST_MIN_AGE | 1800  
FSTYPE | btrfs
```

NUMBER_CLEANUP	yes
NUMBER_LIMIT	5
NUMBER_LIMIT_IMPORTANT	10
NUMBER_MIN_AGE	1800
QGROUP	
SPACE_LIMIT	0.5
SUBVOLUME	/
SYNC_ACL	no
TIMELINE_CLEANUP	yes
TIMELINE_CREATE	yes
TIMELINE_LIMIT_DAILY	5
TIMELINE_LIMIT_HOURLY	5
TIMELINE_LIMIT_MONTHLY	5
TIMELINE_LIMIT_YEARLY	5
TIMELINE_MIN_AGE	1800

Caveats and Errata

root Partition Mounted Multiple Times

You may notice that the root partition gets mounted multiple times. This is due to the way the `btrfs` file system handles subvolumes, mounting the root partition once for each subvolume. `btrfs` keeps one subvolume for each snapshot taken, which stores the snapshot data. While all snapshots are subvolumes, not all subvolumes are snapshots.

Cumulus Linux excludes a number of directories when it takes a snapshot of the root file system (and from any rollbacks):

Directory	Reason
/home	Excluded to avoid user data loss on rollbacks.
/var/log, /var/support	Log file and Cumulus support location. Excluded from snapshots to allow post-rollback analysis.
/tmp, /var/tmp	No need to rollback temporary files.
/opt, /var/opt	Third-party software usually are installed in /opt. Exclude /opt to avoid re-installing these applications after rollbacks.
/srv	Contains data for HTTP and FTP servers. Excluded this directory to avoid server data loss on rollbacks.
/usr/local	This directory is used when installing locally built software. Exclude this directory to avoid re-installing these software after rollbacks.
/var/spool	Exclude this directory to avoid loss of mail after a rollback.
/var/lib/libvirt/images	



Directory	Reason
	This is the default directory for libvirt VM images. Exclude from the snapshot. Additionally disable Copy-On-Write (COW) for this subvolume as COW and VM image I/O access patterns do not play nice.
/boot/grub/i386-pc, /boot/grub/x86_64-efi, /boot/grub/arm-uboot	The GRUB kernel modules must stay in sync with the GRUB kernel installed in the master boot record or UEFI system partition.

Adding and Updating Packages

You use the Advanced Packaging Tool (`apt`) to manage additional applications (in the form of packages) and to install the latest updates.

Before running any `apt-get` commands or after changing the `/etc/apt/sources.list` file, you need to run `apt-get update`.

Contents

This chapter covers ...

- Updating the Package Cache (see page 62)
- Listing Available Packages (see page 64)
- Adding a Package (see page 65)
- Listing Installed Packages (see page 65)
- Upgrading to Newer Versions of Installed Packages (see page 66)
 - Upgrading a Single Package (see page 66)
 - Upgrading All Packages (see page 66)
- Adding Packages from Another Repository (see page 66)
- Related Information (see page 68)

Updating the Package Cache

To work properly, APT relies on a local cache of the available packages. You must populate the cache initially, and then periodically update it with `apt-get update`:

```
cumulus@switch:~$ sudo apt-get update
Get:1 http://repo3.cumulusnetworks.com CumulusLinux-3 InRelease
[7,624 B]
Get:2 http://repo3.cumulusnetworks.com CumulusLinux-3-security-
updates InRelease [7,555 B]
Get:3 http://repo3.cumulusnetworks.com CumulusLinux-3-updates
InRelease [7,660 B]
Get:4 http://repo3.cumulusnetworks.com CumulusLinux-3/cumulus Sources
[20 B]
```



```
Get:5 http://repo3.cumulusnetworks.com CumulusLinux-3/upstream  
Sources [20 B]  
Get:6 http://repo3.cumulusnetworks.com CumulusLinux-3/cumulus amd64  
Packages [38.4 kB]  
Get:7 http://repo3.cumulusnetworks.com CumulusLinux-3/upstream amd64  
Packages [445 kB]  
Get:8 http://repo3.cumulusnetworks.com CumulusLinux-3-security-updates  
/cumulus Sources [20 B]  
Get:9 http://repo3.cumulusnetworks.com CumulusLinux-3-security-updates  
/upstream Sources [11.8 kB]  
Get:10 http://repo3.cumulusnetworks.com CumulusLinux-3-security-  
updates/cumulus amd64 Packages [20 B]  
Get:11 http://repo3.cumulusnetworks.com CumulusLinux-3-security-  
updates/upstream amd64 Packages [8,941 B]  
Get:12 http://repo3.cumulusnetworks.com CumulusLinux-3-updates  
/cumulus Sources [20 B]  
Get:13 http://repo3.cumulusnetworks.com CumulusLinux-3-updates  
/upstream Sources [776 B]  
Get:14 http://repo3.cumulusnetworks.com CumulusLinux-3-updates  
/cumulus amd64 Packages [38.4 kB]  
Get:15 http://repo3.cumulusnetworks.com CumulusLinux-3-updates  
/upstream amd64 Packages [444 kB]  
Ign http://repo3.cumulusnetworks.com CumulusLinux-3/cumulus  
Translation-en_US  
Ign http://repo3.cumulusnetworks.com CumulusLinux-3/cumulus  
Translation-en  
Ign http://repo3.cumulusnetworks.com CumulusLinux-3/upstream  
Translation-en_US  
Ign http://repo3.cumulusnetworks.com CumulusLinux-3/upstream  
Translation-en  
Ign http://repo3.cumulusnetworks.com CumulusLinux-3-security-updates  
/cumulus Translation-en_US  
Ign http://repo3.cumulusnetworks.com CumulusLinux-3-security-updates  
/cumulus Translation-en  
Ign http://repo3.cumulusnetworks.com CumulusLinux-3-security-updates  
/upstream Translation-en_US  
Ign http://repo3.cumulusnetworks.com CumulusLinux-3-security-updates  
/upstream Translation-en  
Ign http://repo3.cumulusnetworks.com CumulusLinux-3-updates/cumulus  
Translation-en_US  
Ign http://repo3.cumulusnetworks.com CumulusLinux-3-updates/cumulus  
Translation-en  
Ign http://repo3.cumulusnetworks.com CumulusLinux-3-updates/upstream  
Translation-en_US  
Ign http://repo3.cumulusnetworks.com CumulusLinux-3-updates/upstream  
Translation-en  
Fetched 1,011 kB in 1s (797 kB/s)  
Reading package lists... Done
```

Listing Available Packages

Once the cache is populated, use `apt-cache` to search the cache to find the packages you are interested in or to get information about an available package. Here are examples of the `search` and `show` sub-commands:

```
cumulus@switch:~$ apt-cache search tcp
socat - multipurpose relay for bidirectional data transfer
quagga-doc - documentation files for quagga
fakeroot - tool for simulating superuser privileges
quagga - BGP/OSPF/RIP routing daemon
tcpdump - command-line network traffic analyzer
.openssh-server - secure shell (SSH) server, for secure access from
remote machines
.openssh-sftp-server - secure shell (SSH) sftp server module, for SFTP
access from remote machines
python-dpkt - Python packet creation / parsing module
libfakeroot - tool for simulating superuser privileges - shared
libraries
.openssh-client - secure shell (SSH) client, for secure access to
remote machines
rsyslog - reliable system and kernel logging daemon
libwrap0 - Wietse Venema's TCP wrappers library
netbase - Basic TCP/IP networking system
```

```
cumulus@switch:~$ apt-cache show tcpdump
Package: tcpdump
Status: install ok installed
Priority: optional
Section: net
Installed-Size: 1092
Maintainer: Romain Francoise <rfrancoise@debian.org>
Architecture: amd64
Multi-Arch: foreign
Version: 4.6.2-5+deb8u1
Depends: libc6 (>= 2.14), libpcap0.8 (>= 1.5.1), libssl1.0.0 (>=
1.0.0)
Description: command-line network traffic analyzer
  This program allows you to dump the traffic on a network. tcpdump
  is able to examine IPv4, ICMPv4, IPv6, ICMPv6, UDP, TCP, SNMP, AFS
  BGP, RIP, PIM, DVMRP, IGMP, SMB, OSPF, NFS and many other packet
  types.

  .
  It can be used to print out the headers of packets on a network
  interface, filter packets that match a certain expression. You can
  use this tool to track down network problems, to detect attacks
  or to monitor network activities.
```



```
Description-md5: f01841bfda357d116d7ff7b7a47e8782
Homepage: http://www.tcpdump.org/
cumulus@switch:~$
```



The search commands look for the search terms not only in the package name but in other parts of the package information. Consequently, it will match on more packages than you would expect.

Adding a Package

In order to add a new package, first ensure the package is not already installed in the system:

```
cumulus@switch:~$ dpkg -l | grep {name of package}
```

If the package is installed already, ensure it's the version you need. If it's an older version, then update the package from the Cumulus Linux repository:

```
cumulus@switch:~$ sudo apt-get update
```

If the package is not already on the system, add it by running `apt-get install`. This retrieves the package from the Cumulus Linux repository and installs it on your system together with any other packages that this package might depend on.

For example, the following adds the package `tcpreplay` to the system:

```
cumulus@switch:~$ sudo apt-get install tcpreplay
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following NEW packages will be installed:
tcpreplay
0 upgraded, 1 newly installed, 0 to remove and 1 not upgraded.
Need to get 436 kB of archives.
After this operation, 1008 kB of additional disk space will be used.
Get:1 https://repo.cumulusnetworks.com/ CumulusLinux-1.5/main
tcpreplay amd64 4.6.2-5+deb8u1 [436 kB]
Fetched 436 kB in 0s (1501 kB/s)
Selecting previously unselected package tcpreplay.
(Reading database ... 15930 files and directories currently
installed.)
Unpacking tcpreplay (from .../tcpreplay_4.6.2-5+deb8u1_amd64.deb) ...
Processing triggers for man-db ...
Setting up tcpreplay (4.6.2-5+deb8u1) ...
cumulus@switch:~$
```



Listing Installed Packages

The APT cache contains information about all the packages available on the repository. To see which packages are actually installed on your system, use `dpkg`. The following example lists all the packages on the system that have "tcp" in their package names:

```
cumulus@switch:~$ dpkg -l \*tcp\*
Desired=Unknown/Install/Remove/Purge/Hold
| Status=Not/Inst/Conf-files/Unpacked/half-conf/Half-inst/trig-aWait
/Trig-pend
| / Err?=(none)/Reinst-required (Status,Err: uppercase=bad)
|| / Name                           Version
Architecture      Description
+====+
=====
un  tcpd              <none>
<none>          (no description available)
ii  tcpdump           4.6.2-5+deb8u1
amd64            command-line network traffic analyzer
cumulus@switch:~$
```

Upgrading to Newer Versions of Installed Packages

Upgrading a Single Package

A single package can be upgraded by simply installing that package again with `apt-get install`. You should perform an update first so that the APT cache is populated with the latest information about the packages.

To see if a package needs to be upgraded, use `apt-cache show <pkgname>` to show the latest version number of the package. Use `dpkg -l <pkgname>` to show the version number of the installed package.

Upgrading All Packages

You can update all packages on the system by running `apt-get update`, then `apt-get upgrade`. This upgrades all installed versions with their latest versions but will not install any new packages.

Adding Packages from Another Repository

As shipped, Cumulus Linux searches the Cumulus Linux repository for available packages. You can add additional repositories to search by adding them to the list of sources that `apt-get` consults. See `man sources.list` for more information.



- For several packages, Cumulus Networks has added features or made bug fixes and these packages must not be replaced with versions from other repositories. Cumulus Linux has been configured to ensure that the packages from the Cumulus Linux repository are always preferred over packages from other repositories.



If you want to install packages that are not in the Cumulus Linux repository, the procedure is the same as above with one additional step.



Packages not part of the Cumulus Linux Repository have generally not been tested, and may not be supported by Cumulus Linux support.

Installing packages outside of the Cumulus Linux repository requires the use of `apt-get`, but, depending on the package, `easy-install` and other commands can also be used.

To install a new package, please complete the following steps:

1. First, ensure package is not already installed in the system. Use the `dpkg` command:

```
cumulus@switch:~$ dpkg -l | grep {name of package}
```

2. If the package is installed already, ensure it's the version you need. If it's an older version, then update the package from the Cumulus Linux repository:

```
cumulus@switch:~$ sudo apt-get update  
cumulus@switch:~$ sudo apt-get install {name of package}  
cumulus@switch:~$ sudo apt-get upgrade
```

3. If the package is not on the system, then most likely the package source location is also **not** in the `/etc/apt/sources.list` file. If the source for the new package is **not** in `sources.list`, please edit and add the appropriate source to the file. For example, add the following if you wanted a package from the Debian repository that is **not** in the Cumulus Linux repository:

```
deb http://http.us.debian.org/debian jessie main  
deb http://security.debian.org/ jessie/updates main
```

Otherwise, the repository may be listed in `/etc/apt/sources.list` but is commented out, as can be the case with the early-access repository:

```
#deb http://repo3.cumulusnetworks.com/repo CumulusLinux-3-early-access cumulus
```

To uncomment the repository, remove the # at the start of the line, then save the file:

```
deb http://repo3.cumulusnetworks.com/repo CumulusLinux-3-early-access cumulus
```

4. Run `apt-get update` then install the package and upgrade:



```
cumulus@switch:~$ sudo apt-get update  
cumulus@switch:~$ sudo apt-get install {name of package}  
cumulus@switch:~$ sudo apt-get upgrade
```

Related Information

- [Debian GNU/Linux FAQ, Ch 8 Package management tools](#)
- man pages for apt-get, dpkg, sources.list, apt_preferences

Zero Touch Provisioning - ZTP

Zero touch provisioning (ZTP) enables network devices to be quickly deployed in large-scale environments. On first boot, Cumulus Linux will invoke ZTP, which executes the provisioning automation used to deploy the device for its intended role in the network.

The provisioning framework allows for a one-time, user-provided script to be executed. You can develop this script using a variety of automation tools and scripting languages, providing ample flexibility for you to design the provisioning scheme to meet your needs. You can also use it to add the switch to a configuration management (CM) platform such as [Puppet](#), [Chef](#), [CFEngine](#) or possibly a custom, proprietary tool.

While developing and testing the provisioning logic, you can use the `ztp` command in Cumulus Linux to manually invoke your provisioning script on a device.

ZTP in Cumulus Linux can occur automatically in one of the following ways, in this order:

- Via a local file
- Using a USB drive inserted into the switch (ZTP-USB)
- Via DHCP

Each method is discussed in greater detail below.



The standard Cumulus Linux license requires you to page through the license file before accepting the terms, which can hinder an unattended installation like zero touch provisioning. To request a license without the EULA, email licensing@cumulusnetworks.com.

Contents

Click to expand...

- [Zero Touch Provisioning Using a Local File \(see page 69\)](#)
- [Zero Touch Provisioning Using USB \(ZTP-USB\) \(see page 69\)](#)
- [Zero Touch Provisioning over DHCP \(see page 70\)](#)
 - [Triggering ZTP over DHCP \(see page 70\)](#)
 - [Configuring The DHCP Server \(see page 70\)](#)
 - [Detailed Look at HTTP Headers \(see page 71\)](#)
- [Writing ZTP Scripts \(see page 71\)](#)
 - [Example ZTP Scripts \(see page 72\)](#)

- Testing and Debugging ZTP Scripts (see page 74)
- Manually Using the ztp Command (see page 78)
- Notes (see page 79)

Zero Touch Provisioning Using a Local File

ZTP only looks once for a ZTP script on the local file system when the switch boots. ZTP searches for an install script that matches an ONIE-style waterfall in `/var/lib/cumulus/ztp`, looking for the most specific name first, and ending at the most generic:

- `'cumulus-ztp-' + architecture + '-' + vendor + '_' + model + '-r' + revision`
- `'cumulus-ztp-' + architecture + '-' + vendor + '_' + model`
- `'cumulus-ztp-' + vendor + '_' + model`
- `'cumulus-ztp-' + architecture`
- `'cumulus-ztp'`

For example:

```
cumulus-ztp-amd64-cel_pebble-rUNKNOWN
cumulus-ztp-amd64-cel_pebble
cumulus-ztp-cel_pebble
cumulus-ztp-amd64
cumulus-ztp
```

You can also trigger the ZTP process manually by running the `ztp --run <URL>` command, where the URL is the path to the ZTP script.

Zero Touch Provisioning Using USB (ZTP-USB)



This feature has been tested only with "thumb" drives, not an actual external large USB hard drive.

If the `ztp` process did not discover a local script, it tries once to locate an inserted but unmounted USB drive. If it discovers one, it begins the ZTP process.

Cumulus Linux supports the use of a FAT32, FAT16, or VFAT-formatted USB drive as an installation source for ZTP scripts. You must plug in the USB stick **before** you power up the switch.

At minimum, the script should:

- Install the Cumulus Linux operating system and license.
- Copy over a basic configuration to the switch.
- Restart the switch or the relevant server to get `switchd` up and running with that configuration.

Follow these steps to perform zero touch provisioning using USB:

1. Copy the Cumulus Linux license and installation image to the USB stick.



2. The `ztp` process searches the root filesystem of the newly mounted device for filenames matching an [ONIE-style waterfall](#) (see the patterns and examples above), looking for the most specific name first, and ending at the most generic.
3. The script's contents are parsed to ensure it contains the `CUMULUS-AUTOPROVISIONING` flag (see [example scripts \(see page 72\)](#)).



The USB device is mounted to a temporary directory under `/tmp` (for example, `/tmp/pigGgjf/`). In order to reference files on the USB, use the environment variable `ZTP_USB_MOUNTPOINT` to refer to the USB root partition.

Zero Touch Provisioning over DHCP

If the `ztp` process did not discover a local/ONIE script or applicable USB drive, it checks DHCP every 10 seconds for up to 5 minutes for the presence of a ZTP URL specified in `/var/run/ztp.dhcp`. The URL can be any of HTTP, HTTPS, FTP or TFTP.

For ZTP using DHCP, provisioning initially takes place over the management network and is initiated via a DHCP hook. A DHCP option is used to specify a configuration script. This script is then requested from the Web server and executed locally on the switch.

The zero touch provisioning process over DHCP follows these steps:

1. The first time you boot Cumulus Linux, `eth0` is configured for DHCP and makes a DHCP request.
2. The DHCP server offers a lease to the switch.
3. If option 239 is present in the response, the zero touch provisioning process itself will start.
4. The zero touch provisioning process requests the contents of the script from the URL, sending additional [HTTP headers \(see page 71\)](#) containing details about the switch.
5. The script's contents are parsed to ensure it contains the `CUMULUS-AUTOPROVISIONING` flag (see [example scripts \(see page 72\)](#)).
6. If provisioning is necessary, then the script executes locally on the switch with root privileges.
7. The return code of the script gets examined. If it is 0, then the provisioning state is marked as complete in the autoprovisioning configuration file.

Triggering ZTP over DHCP

If provisioning has not already occurred, it is possible to trigger the zero touch provisioning process over DHCP when `eth0` is set to use DHCP and one of the following events occur:

- Booting the switch
- Plugging a cable into or unplugging it from the `eth0` port
- Disconnecting then reconnecting the switch's power cord

You can also run the `ztp --run <URL>` command, where the URL is the path to the ZTP script.

Configuring The DHCP Server

During the DHCP process over `eth0`, Cumulus Linux will request DHCP option 239. This option is used to specify the custom provisioning script.



For example, the `/etc/dhcp/dhcpd.conf` file for an ISC DHCP server would look like:

```
option cumulus-provision-url code 239 = text;

subnet 192.0.2.0 netmask 255.255.255.0 {
    range 192.0.2.100 192.168.0.200;
    option cumulus-provision-url "http://192.0.2.1/demo.sh";
}
```

Additionally, the hostname of the switch can be specified via the `host-name` option:

```
subnet 192.168.0.0 netmask 255.255.255.0 {
    range 192.168.0.100 192.168.0.200;
    option cumulus-provision-url "http://192.0.2.1/demo.sh";
    host dcl-tor-swl { hardware ethernet 44:38:39:00:1a:6b; fixed-
        address 192.168.0.101; option host-name "dcl-tor-swl"; }
}
```

Detailed Look at HTTP Headers

The following HTTP headers are sent in the request to the webserver to retrieve the provisioning script:

Header	Value	Example
User-Agent	-----	CumulusLinux-
AutoProvision/0.4		
CUMULUS-ARCH	CPU architecture	x86_64
CUMULUS-BUILD		3.0.0-5c6829a-
201309251712-final		
CUMULUS-LICENSE-INSTALLED	Either 0 or 1	1
CUMULUS-MANUFACTURER		odm
CUMULUS-PRODUCTNAME		switch_model
CUMULUS-SERIAL		XYZ123004
CUMULUS-VERSION		3.0.0
CUMULUS-PROV-COUNT		0
CUMULUS-PROV-MAX		32

Writing ZTP Scripts



Remember to include the following line in any of the supported scripts which are expected to be run via the autoprovisioning framework.

```
# CUMULUS-AUTOPROVISIONING
```



This line is required somewhere in the script file in order for execution to occur.

The script must contain the `CUMULUS-AUTOPROVISIONING` flag. This can be in a comment or remark and does not need to be echoed or written to `stdout`.

The script can be written in any language currently supported by Cumulus Linux, such as:

- Perl
- Python
- Ruby
- Shell

The script must return an exit code of 0 upon success, as this triggers the autoprovisioning process to be marked as complete in the autoprovisioning configuration file.

Example ZTP Scripts

The following script installs Cumulus Linux and its license from USB and applies a configuration:

```
#!/bin/bash
function error() {
    echo -e "\e[0;33mERROR: The Zero Touch Provisioning script failed
while running the command $BASH_COMMAND at line $BASH_LINENO.\e[0m"
>&2
    exit 1
}

# Log all output from this script
exec >/var/log/autoprovision 2>&1

trap error ERR

#Add Debian Repositories
echo "deb http://http.us.debian.org/debian jessie main" >> /etc/apt
/sources.list
echo "deb http://security.debian.org/ jessie/updates main" >> /etc/apt
/sources.list

#Update Package Cache
apt-get update -y

#Install netshow diagnostics commands
apt-get install -y netshow htop nmap

#Load interface config from usb
cp ${ZTP_USB_MOUNTPOINT}/interfaces /etc/network/interfaces

#Load port config from usb
# (if breakout cables are used for certain interfaces)
cp ${ZTP_USB_MOUNTPOINT}/ports.conf /etc/cumulus/ports.conf
```



```
#Install a License from usb and restart switchd
cl-license -i ${ZTP_USB_MOUNTPOINT}/license.txt && systemctl restart
switchd.service

#Reload interfaces to apply loaded config
ifreload -a

#Output state of interfaces
netshow interface

# CUMULUS-AUTOPROVISIONING
exit 0
```

Here is a simple script to install puppet:

```
#!/bin/bash
function error() {
    echo -e "\e[0;33mERROR: The Zero Touch Provisioning script failed
while running the command $BASH_COMMAND at line $BASH_lineno.\e[0m"
>&2
    exit 1
}
trap error ERR
apt-get update -y
apt-get upgrade -y
apt-get install puppet -y
sed -i /etc/default/puppet -e 's/START=no/START=yes/'
sed -i /etc/puppet/puppet.conf -e 's/\\[main\\]\\/\\[main\\]
\npluginsync=true/'
systemctl restart puppet.service
# CUMULUS-AUTOPROVISIONING
exit 0
```

This script illustrates how to specify an internal apt mirror and puppet master:

```
#!/bin/bash
function error() {
    echo -e "\e[0;33mERROR: The Zero Touch Provisioning script failed
while running the command $BASH_COMMAND at line $BASH_lineno.\e[0m"
>&2
    exit 1
}
trap error ERR
sed -i /etc/apt/sources.list -e 's/repo.cumulusnetworks.com/labrepo.
mycompany.com/'
apt-get update -y
apt-get upgrade -y
apt-get install puppet -y
```



```
sed -i /etc/default/puppet -e 's/START=no/START=yes/'  
sed -i /etc/puppet/puppet.conf -e 's/\\[main\\]/\\[main\\]  
\npluginsync=true/'  
sed -i /etc/puppet/puppet.conf -e 's/\\[main\\]/\\[main\\]  
\nserver=labpuppet.mycompany.com/'  
systemctl restart puppet.service  
# CUMULUS-AUTOPROVISIONING  
exit 0
```

Now puppet can take over management of the switch, configuration authentication, changing the default root password, and setting up interfaces and routing protocols.

Several ZTP example scripts are available in the [Cumulus GitHub repository](#).

Testing and Debugging ZTP Scripts

There are a few commands you can use to test and debug your ZTP scripts.

You can use verbose mode to debug your script and see where your script failed. Include the `-v` option when you run `ztp`:

```
cumulus@switch:~$ sudo ztp -v -r http://192.0.2.1/demo.sh  
Attempting to provision via ZTP Manual from http://192.0.2.1/demo.sh  
  
Broadcast message from root@dell-s6000-01 (ttyS0) (Tue May 10 22:44:  
17 2016):  
  
ZTP: Attempting to provision via ZTP Manual from http://192.0.2.1  
/demo.sh  
ZTP Manual: URL response code 200  
ZTP Manual: Found Marker CUMULUS-AUTOPROVISIONING  
ZTP Manual: Executing http://192.0.2.1/demo.sh  
error: ZTP Manual: Payload returned code 1  
error: Script returned failure
```

You can also run `ztp -s` to get more information about the current state of ZTP.

```
cumulus@switch:~$ ztp -s  
ZTP INFO:  
  
State          enabled  
Version        1.0  
Result         Script Failure  
Date          Tue May 10 22:42:09 2016 UTC  
Method        ZTP DHCP  
URL           http://192.0.2.1/demo.sh
```

If ZTP ran when the switch booted and not manually, you can run the `systemctl -l status ztp.service` then `journalctl -l -u ztp.service` to see if any failures occur:



```
cumulus@switch:~$ sudo systemctl -l status ztp.service
ztp.service - Cumulus Linux ZTP
   Loaded: loaded (/lib/systemd/system/ztp.service; enabled)
   Active: failed (Result: exit-code) since Wed 2016-05-11 16:38:45
             UTC; 1min 47s ago
     Docs: man:ztp(8)
   Process: 400 ExecStart=/usr/sbin/ztp -b (code=exited, status=1
/Failure)
      Main PID: 400 (code=exited, status=1/Failure)

May 11 16:37:45 cumulus ztp[400]: ztp [400]: ZTP USB: Device not found
May 11 16:38:45 dell-s6000-01 ztp[400]: ztp [400]: ZTP DHCP: Looking
for ZTP Script provided by DHCP
May 11 16:38:45 dell-s6000-01 ztp[400]: ztp [400]: Attempting to
provision via ZTP DHCP from http://192.0.2.1/demo.sh
May 11 16:38:45 dell-s6000-01 ztp[400]: ztp [400]: ZTP DHCP: URL
response code 200
May 11 16:38:45 dell-s6000-01 ztp[400]: ztp [400]: ZTP DHCP: Found
Marker CUMULUS-AUTOPROVISIONING
May 11 16:38:45 dell-s6000-01 ztp[400]: ztp [400]: ZTP DHCP:
Executing http://192.0.2.1/demo.sh
May 11 16:38:45 dell-s6000-01 ztp[400]: ztp [400]: ZTP DHCP: Payload
returned code 1
May 11 16:38:45 dell-s6000-01 ztp[400]: ztp [400]: Script returned
failure
May 11 16:38:45 dell-s6000-01 systemd[1]: ztp.service: main process
exited, code=exited, status=1/Failure
May 11 16:38:45 dell-s6000-01 systemd[1]: Unit ztp.service entered
failed state.
cumulus@switch:~$
cumulus@switch:~$ sudo journalctl -l -u ztp.service --no-pager
-- Logs begin at Wed 2016-05-11 16:37:42 UTC, end at Wed 2016-05-11
16:40:39 UTC. --
May 11 16:37:45 cumulus ztp[400]: ztp [400]: /var/lib/cumulus/ztp:
State Directory does not exist. Creating it...
May 11 16:37:45 cumulus ztp[400]: ztp [400]: /var/run/ztp.lock: Lock
File does not exist. Creating it...
May 11 16:37:45 cumulus ztp[400]: ztp [400]: /var/lib/cumulus/ztp
/ztp_state.log: State File does not exist. Creating it...
May 11 16:37:45 cumulus ztp[400]: ztp [400]: ZTP LOCAL: Looking for
ZTP local Script
May 11 16:37:45 cumulus ztp[400]: ztp [400]: ZTP LOCAL: Waterfall
search for /var/lib/cumulus/ztp/cumulus-ztp-x86_64-dell_s6000_s1220-
rUNKNOWN
May 11 16:37:45 cumulus ztp[400]: ztp [400]: ZTP LOCAL: Waterfall
search for /var/lib/cumulus/ztp/cumulus-ztp-x86_64-dell_s6000_s1220
May 11 16:37:45 cumulus ztp[400]: ztp [400]: ZTP LOCAL: Waterfall
search for /var/lib/cumulus/ztp/cumulus-ztp-x86_64-dell
May 11 16:37:45 cumulus ztp[400]: ztp [400]: ZTP LOCAL: Waterfall
search for /var/lib/cumulus/ztp/cumulus-ztp-x86_64
```

```

May 11 16:37:45 cumulus ztp[400]: ztp [400]: ZTP LOCAL: Waterfall
search for /var/lib/cumulus/ztp/cumulus-ztp
May 11 16:37:45 cumulus ztp[400]: ztp [400]: ZTP USB: Looking for
unmounted USB devices
May 11 16:37:45 cumulus ztp[400]: ztp [400]: ZTP USB: Parsing
partitions
May 11 16:37:45 cumulus ztp[400]: ztp [400]: ZTP USB: Device not found
May 11 16:38:45 dell-s6000-01 ztp[400]: ztp [400]: ZTP DHCP: Looking
for ZTP Script provided by DHCP
May 11 16:38:45 dell-s6000-01 ztp[400]: ztp [400]: Attempting to
provision via ZTP DHCP from http://192.0.2.1/demo.sh
May 11 16:38:45 dell-s6000-01 ztp[400]: ztp [400]: ZTP DHCP: URL
response code 200
May 11 16:38:45 dell-s6000-01 ztp[400]: ztp [400]: ZTP DHCP: Found
Marker CUMULUS-AUTOPROVISIONING
May 11 16:38:45 dell-s6000-01 ztp[400]: ztp [400]: ZTP DHCP:
Executing http://192.0.2.1/demo.sh
May 11 16:38:45 dell-s6000-01 ztp[400]: ztp [400]: ZTP DHCP: Payload
returned code 1
May 11 16:38:45 dell-s6000-01 ztp[400]: ztp [400]: Script returned
failure
May 11 16:38:45 dell-s6000-01 systemd[1]: ztp.service: main process
exited, code=exited, status=1/FAILURE
May 11 16:38:45 dell-s6000-01 systemd[1]: Unit ztp.service entered
failed state.

```

Instead of running journalctl, you can see the log history by running:

```

cumulus@switch:~$ cat /var/log/syslog | grep ztp
2016-05-11T16:37:45.132583+00:00 cumulus ztp [400]: /var/lib/cumulus
/ztp: State Directory does not exist. Creating it...
2016-05-11T16:37:45.134081+00:00 cumulus ztp [400]: /var/run/ztp.
lock: Lock File does not exist. Creating it...
2016-05-11T16:37:45.135360+00:00 cumulus ztp [400]: /var/lib/cumulus
/ztp/ztp_state.log: State File does not exist. Creating it...
2016-05-11T16:37:45.185598+00:00 cumulus ztp [400]: ZTP LOCAL:
Looking for ZTP local Script
2016-05-11T16:37:45.485084+00:00 cumulus ztp [400]: ZTP LOCAL:
Waterfall search for /var/lib/cumulus/ztp/cumulus-ztp-x86_64-
dell_s6000_s1220-rUNKNOWN
2016-05-11T16:37:45.486394+00:00 cumulus ztp [400]: ZTP LOCAL:
Waterfall search for /var/lib/cumulus/ztp/cumulus-ztp-x86_64-
dell_s6000_s1220
2016-05-11T16:37:45.488385+00:00 cumulus ztp [400]: ZTP LOCAL:
Waterfall search for /var/lib/cumulus/ztp/cumulus-ztp-x86_64-dell
2016-05-11T16:37:45.489665+00:00 cumulus ztp [400]: ZTP LOCAL:
Waterfall search for /var/lib/cumulus/ztp/cumulus-ztp-x86_64
2016-05-11T16:37:45.490854+00:00 cumulus ztp [400]: ZTP LOCAL:
Waterfall search for /var/lib/cumulus/ztp/cumulus-ztp

```



```
2016-05-11T16:37:45.492296+00:00 cumulus ztp [400]: ZTP USB: Looking  
for unmounted USB devices  
2016-05-11T16:37:45.493525+00:00 cumulus ztp [400]: ZTP USB: Parsing  
partitions  
2016-05-11T16:37:45.636422+00:00 cumulus ztp [400]: ZTP USB: Device  
not found  
2016-05-11T16:38:43.372857+00:00 cumulus ztp [1805]: Found ZTP DHCP  
Request  
2016-05-11T16:38:45.696562+00:00 cumulus ztp [400]: ZTP DHCP: Looking  
for ZTP Script provided by DHCP  
2016-05-11T16:38:45.698598+00:00 cumulus ztp [400]: Attempting to  
provision via ZTP DHCP from http://192.0.2.1/demo.sh  
2016-05-11T16:38:45.816275+00:00 cumulus ztp [400]: ZTP DHCP: URL  
response code 200  
2016-05-11T16:38:45.817446+00:00 cumulus ztp [400]: ZTP DHCP: Found  
Marker CUMULUS-AUTOPROVISIONING  
2016-05-11T16:38:45.818402+00:00 cumulus ztp [400]: ZTP DHCP:  
Executing http://192.0.2.1/demo.sh  
2016-05-11T16:38:45.834240+00:00 cumulus ztp [400]: ZTP DHCP: Payload  
returned code 1  
2016-05-11T16:38:45.835488+00:00 cumulus ztp [400]: Script returned  
failure  
2016-05-11T16:38:45.876334+00:00 cumulus systemd[1]: ztp.service:  
main process exited, code=exited, status=1/FAILURE  
2016-05-11T16:38:45.879410+00:00 cumulus systemd[1]: Unit ztp.service  
entered failed state.
```

If you see that the issue is a script failure, you can modify the script and then run `ztp` manually using `ztp -v -r <URL/path to that script>`, as above.

```
cumulus@switch:~$ sudo ztp -v -r http://192.0.2.1/demo.sh  
Attempting to provision via ZTP Manual from http://192.0.2.1/demo.sh  
  
Broadcast message from root@dell-s6000-01 (ttyS0) (Tue May 10 22:44:  
17 2016):  
  
ZTP: Attempting to provision via ZTP Manual from http://192.0.2.1  
/demo.sh  
ZTP Manual: URL response code 200  
ZTP Manual: Found Marker CUMULUS-AUTOPROVISIONING  
ZTP Manual: Executing http://192.0.2.1/demo.sh  
error: ZTP Manual: Payload returned code 1  
error: Script returned failure  
cumulus@switch:~$ sudo ztp -s  
State      enabled  
Version    1.0  
Result     Script Failure  
Date       Tue May 10 22:44:17 2016 UTC  
Method     ZTP Manual  
URL       http://192.0.2.1/demo.sh
```



Manually Using the ztp Command

To enable zero touch provisioning, use the `-e` option:

```
cumulus@switch:~$ sudo ztp -e
```



Enabling `ztp` means that `ztp` will try to occur the next time the switch boots. However, if ZTP already occurred on a previous boot up or if a manual configuration has been found, ZTP will just exit without trying to look for any script.

ZTP checks for these manual configurations during bootup:

- Password changes
- Users and groups changes
- Packages changes
- Interfaces changes
- The presence of an installed license

When the switch is booted for the very first time, ZTP records the state of some important files that are most likely going to be modified after that the switch is configured. If ZTP is still enabled after a reboot, ZTP will compare the recorded state to the current state of these files. If they do not match, ZTP considers that the switch has already been provisioned and exits. These files are only erased after a reset.

To reset `ztp` to its original state, use the `-R` option. This removes the `ztp` directory and `ztp` runs the next time the switch reboots.

```
cumulus@switch:~$ sudo ztp -R
```

To disable zero touch provisioning, use the `-d` option:

```
cumulus@switch:~$ sudo ztp -d
```

To force provisioning to occur and ignore the status listed in the configuration file use the `-r` option:

```
cumulus@switch:~$ sudo ztp -r cumulus-ztp.sh
```

To see the current `ztp` state, use the `-s` option:

```
cumulus@switch:~$ sudo ztp -s
ZTP INFO:
State disabled
```



```
Version 1.0
Result success
Date Thu May 5 16:49:33 2016 UTC
Method Switch manually configured
URL None
```

Notes

- During the development of a provisioning script, the switch may need to be rebooted.
- You can use the Cumulus Linux `onie-select -i` command to cause the switch to reprovision itself and install a network operating system again using ONIE.

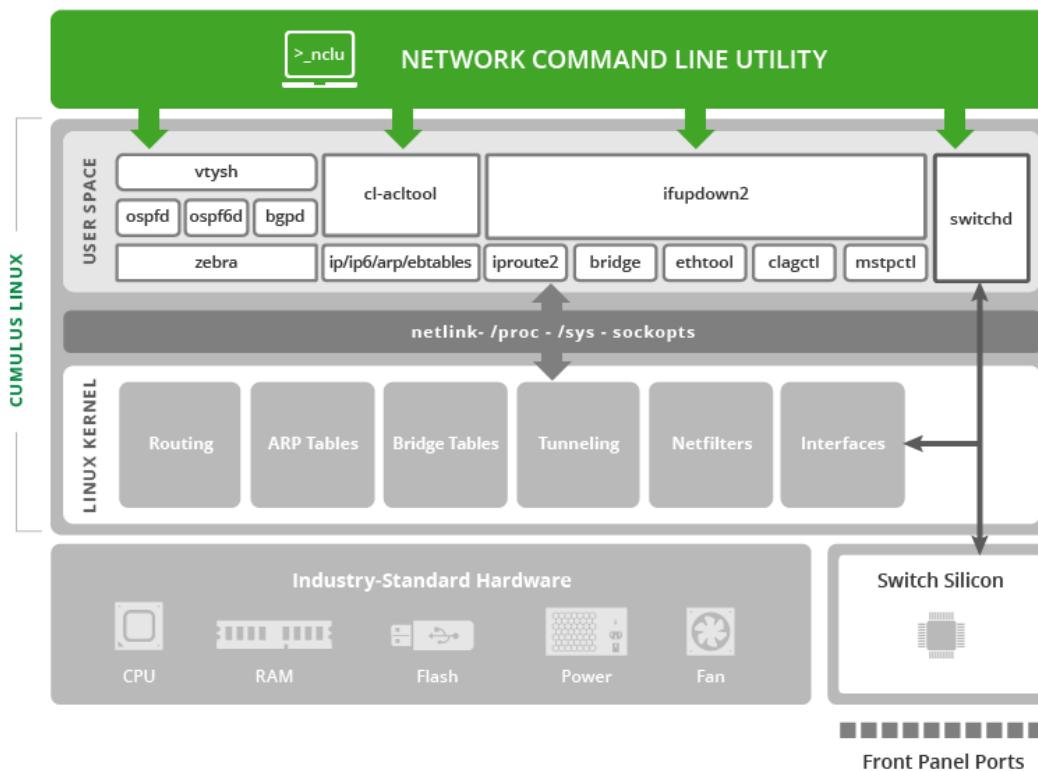
System Configuration

Network Command Line Utility

The Network Command Line Utility, or NCLU, is a command line interface for Cumulus Networks products, implemented in Cumulus Linux and VX 3.1 and later releases, with the goal of simplifying the networking configuration process for all users.

NCLU resides in the Linux user space, as seen below. It provides consistent access to networking commands directly via bash, thereby making configuration and troubleshooting simple and easy — no need to edit files or enter modes and sub-modes. In addition, NCLU does more than traditional command line interfaces by:

- Embedding help, examples and automatic command checking with suggestions in case you've entered a typo
- Running directly from and integrating with bash, while being interoperable with the regular way of accessing underlying configuration files and automation
- Automatically configuring dependent features so you don't have to



The NCLU wrapper utility is called `net`. `net` is capable of configuring L2 and L3 features of the networking stack, installing ACLs and VXLANs, rolling back and deleting snapshots, as well as providing monitoring and troubleshooting functionality for these features. `/etc/network/interfaces` and `/etc/quagga/Quagga.conf` can both be configured with `net`, in addition to running show and clear commands related to `ifupdown2` and Quagga.



Contents

This chapter covers ...

- Installing NCLU (see page 81)
- Getting Started (see page 81)
 - Tab Completion, Verification and Inline Help (see page 82)
 - Built-In Examples (see page 84)
- Adding More NCLU Users or Groups (see page 85)
- Restarting the netd Service (see page 86)
- Advanced Configuration (see page 86)

Installing NCLU

If you upgraded Cumulus Linux from a version earlier than 3.2 instead of performing a full binary install, you need to install the `nclu` package on your switch:

```
cumulus@switch:~$ sudo apt-get update  
cumulus@switch:~$ sudo apt-get install nclu  
cumulus@switch:~$ sudo apt-get upgrade
```



The `nclu` package installs a new bash completion script, and displays the following message when it is manually installed:

```
Setting up nclu (1.0-cl3u3) ...  
To enable the newly installed bash completion for nclu in  
this shell, execute...  
    source /etc/bash_completion
```

Getting Started

NCLU uses the following workflow for staging and committing changes to Cumulus Linux:

1. Use the `net add` and `net del` commands to stage/remove configuration changes.
2. Use the `net pending` command to review staged changes.
3. Use `net commit` and `net abort` to commit/delete staged changes.





⚠ `net commit` applies the changes to the relevant configuration files, such as `/etc/network/interfaces`, then runs necessary follow on commands to enable the configuration, such as `ifreload -a`.

Once you have a running configuration, you can review and update it using:

- `net show`: A series of commands for viewing various parts of the network configuration, such as `net show configuration`, `net show commit history` and `net show bgp` to view the complete network configuration, a history of commits using NCLU and BGP status, respectively.
- `net clear`: A way to clear netstat counters, BGP and OSPF neighbor content, and more.
- `net rollback`: Provides a mechanism to revert back to an earlier configuration.

Tab Completion, Verification and Inline Help

NCLU provides a number of features to assist users. In addition to tab completion and partial keyword commands identification, verification checks are set in place to ensure correct syntax is used. The examples below show the output for incorrect commands:

```
cumulus@switch:~$ net add bgp router-id 1.1.1.1/32
ERROR: Command not found

Did you mean one of the following?

    net add bgp router-id <ipv4>
        This command is looking for an IP address, not an IP/prefixlen

cumulus@switch:~$ net add bgp router-id 1.1.1.1
cumulus@switch:~$ net add int swp10 mtu <TAB>
    <552-9216> :
cumulus@switch:~$ net add int swp10 mtu 9300
ERROR: Command not found

Did you mean one of the following?
    net add interface <interface> mtu <552-9216>
```

NCLU has a comprehensive help system built in to assist usage. In addition to the net man page, you can use `?` and `help` to display available commands:

```
cumulus@switch:~$ net help

Usage:
    # net <COMMAND> [<ARGS>] [help]
    #
    # net is a command line utility for networking on Cumulus Linux
    switches.
    #
    # COMMANDS are listed below and have context specific arguments
    which can
```



```
# be explored by typing "<TAB>" or "help" anytime while using net.
#
# Use 'man net' for a more comprehensive overview.

net abort
net commit [verbose] [confirm] [description <wildcard>]
net commit delete (<number>|<number-range>)
net help [verbose]
net pending
net rollback (<number>|last)
net show commit (history|<number>|<number-range>|last)
net show rollback (<number>|last)
net show configuration
[commands|files|acl|bgp|ospf|ospf6|interface <interface>]
```

Options:

```
# Help commands
help      : context sensitive information; see section below
example   : detailed examples of common workflows
```

```
# Configuration commands
add       : add/modify configuration
del       : remove configuration
```

```
# Commit buffer commands
abort     : abandon changes in the commit buffer
commit    : apply the commit buffer to the system
pending   : show changes staged in the commit buffer
rollback  : revert to a previous configuration state
```

```
# Status commands
show      : show command output
clear     : clear counters, BGP neighbors, etc
```

```
cumulus@switch:~$ net help bestpath
The following commands contain keyword(s) 'bestpath'
```

```
net (add|del) bgp bestpath as-path multipath-relax [as-set|no-as-
set]
net (add|del) bgp bestpath compare-routerid
net (add|del) bgp bestpath med missing-as-worst
net (add|del) bgp vrf <text> bestpath as-path multipath-relax [as-
set|no-as-set]
net (add|del) bgp vrf <text> bestpath compare-routerid
net (add|del) bgp vrf <text> bestpath med missing-as-worst
```

```
net add bgp debug bestpath <ip/prefixlen>
net del bgp debug bestpath [<ip/prefixlen>]
net show bgp (<ipv4>|<ipv4/prefixlen>) [bestpath|multipath] [json]
net show bgp (<ipv6>|<ipv6/prefixlen>) [bestpath|multipath] [json]
net show bgp vrf <text> (<ipv4>|<ipv4/prefixlen>)
[bestpath|multipath] [json]
```



Multiple interfaces can be configured at once:

```
cumulus@switch:~$ net add int swp7-9,12,15-17,22 mtu 9216
```

Built-In Examples

The NCLU has a number of built in examples to guide users through basic configuration setup:

```
cumulus@switch:~$ net example
    acl          : access-list
    bgp          : Border Gateway Protocol
    bond         : Bond, port-channel, etc
    bridge       : interface
    clag         : Multi-Chassis Link Aggregation
    link-settings : Physical link parameters
    lnv          : Lightweight Network Virtualization
    management-vrf : Management VRF
    ospf         : Open Shortest Path First (OSPFv2)
    vlan-interfaces : IP interfaces for VLANs
```

```
cumulus@switch:~$ net example bridge
```

Scenario

=====

We are configuring switch1 and would like to configure the following

- configure switch1 as an L2 switch for host-11 and host-12
- enable vlans 10-20
- place host-11 in vlan 10
- place host-12 in vlan 20
- create an SVI interface for vlan 10
- create an SVI interface for vlan 20
- assign IP 10.0.0.1/24 to the SVI for vlan 10
- assign IP 20.0.0.1/24 to the SVI for vlan 20
- configure swp3 as a trunk for vlans 10, 11, 12 and 20
 - swp3
 - *switch1 ----- switch2
 - /\
 - swp1 / \ swp2



```
      /   \
      /     \
host-11    host-12

switch1 net commands
=====
- enable vlans 10-20
switch1# net add vlan 10-20
- place host-11 in vlan 10
- place host-12 in vlan 20
switch1# net add int swp1 bridge access 10
switch1# net add int swp2 bridge access 20
- create an SVI interface for vlan 10
- create an SVI interface for vlan 20
- assign IP 10.0.0.1/24 to the SVI for vlan 10
- assign IP 20.0.0.1/24 to the SVI for vlan 20
switch1# net add vlan 10 ip address 10.0.0.1/24
switch1# net add vlan 20 ip address 20.0.0.1/24
- configure swp3 as a trunk for vlans 10, 11, 12 and 20
switch1# net add int swp3 bridge trunk vlans 10-12,20
# Review and commit changes
switch1# net pending
switch1# net commit

Verification
=====
switch1# net show interface
switch1# net show bridge macs
```

Adding More NCLU Users or Groups

If you've created custom users or groups on your Cumulus Linux switches, you can configure those users to be able to run NCLU commands.

To do so, edit the `/etc/netd.conf` file, and add those users to the `users_with_edit` and `users_with_show` lines in the file, then save the file.

```
cumulus@switch:~$ sudo nano /etc/netd.conf

# Control which users/groups are allowed to run 'add', 'del',
# 'clear', 'net abort', 'net commit' and restart services (quagga,
etc)
# to apply those changes
users_with_edit = root, cumulus
groups_with_edit = root, cumulus

# Control which users/groups are allowed to run 'show' commands
users_with_show = root, cumulus
groups_with_show = root, cumulus
```



Similarly, to configure a new user group to use NCLU, add that group to the `groups_with_edit` and `groups_with_show` lines in the file.

Restarting the netd Service

Whenever you modify `netd.conf`, you must restart the `netd` service for the changes to take effect:

```
cumulus@switch:~$ sudo systemctl restart netd.service
```

Advanced Configuration

NCLU needs no initial configuration; it's ready to go in Cumulus Linux. However, if you need to modify its configuration, you must manually update the `/etc/netd.conf` file. This file can be configured to allow different permission levels for users to edit configurations and run show commands. It also contains a blacklist that hides less frequently used terms from the tabbed autocomplete.

Configuration Variable	Default Setting	Description
show_linux_command	False	When true, displays the Linux command running in the background.
enable_ifupdown2	True	Enables net wrapping of ifupdown2 commands.
enable_quagga	True	Enables net wrapping of Quagga commands.
users_with_edit	root, cumulus	Sets the Linux users with root edit privileges.
groups_with_edit	root, cumulus	Sets the Linux groups with root edit privileges.



Configuration Variable	Default Setting	Description
users_with_show	root, cumulus	Controls which users are allowed to run <code>show</code> commands.
groups_with_show	root, cumulus	Controls which groups are allowed to run <code>show</code> commands.
ifupdown_blacklist	address-purge, bond-ad-actor-sys-prio, bond-ad-actor-system, bond-mode, bond-num-grat-arp, bond-num-unsol-na, bond-use-carrier, bond-xmit-hash-policy, bridge-bridgeprio, bridge-fd, bridge-hashed, bridge-hashmax, bridge-hello, bridge-maxage, bridge-maxwait, bridge-mclmc, bridge-mclmi, bridge-mcmi, bridge-mcq, bridge-mcqpi, bridge-mcqri, bridge-mcrouter, bridge-mcsqc, bridge-mcsqi, bridge-pathcosts, bridge-port-pvids, bridge-port-vids, bridge-portprios, bridge-stp, bridge-waitport, broadcast, hwaddress, link-type, mstpcctl-ageing, mstpcctl-fdelay, mstpcctl-forcevers, mstpcctl-hello, mstpcctl-maxage, mstpcctl-maxhops, mstpcctl-portp2p, mstpcctl-portpathcost, mstpcctl-portrestrrole, mstpcctl-portrestrtcn, mstpcctl-treeportcost, mstpcctl-treeportprio, mstpcctl-txholdcount, netmask, preferred-lifetime, scope, vxlan-ageing, vxlan-learning, up, down, bridge-ageing, bridge-gcint, bridge-mcqifaddr, bridge-mcqf4src	Hides corner case command options from tab complete, to simplify and streamline output.

ⓘ Net Tab Complete Output

`net` provides an environment variable for setting where the `net` output is directed. To only use `stdout`, set the `NCLU_TAB_STDOUT` environment variable to `true`. The value is not case sensitive.

Setting Date and Time

Setting the time zone, date and time requires root privileges; use `sudo`.

Contents

This chapter covers ...

- [Setting the Time Zone \(see page 88\)](#)
 - Alternative: Use the Guided Wizard to Find and Apply a Time Zone (see page 88)
- [Setting the Date and Time \(see page 89\)](#)



- Setting Time Using NTP (see page 90)
- Specifying the NTP Source Interface (see page 90)
- NTP Default Configuration (see page 91)
- Related Information (see page 92)

Setting the Time Zone

To see the current time zone, list the contents of `/etc/timezone`:

```
cumulus@switch:~$ cat /etc/timezone
US/Eastern
```

Edit the file to add your desired time zone. A list of valid time zones can be found at the following [link](#).

Use the following command to apply the new time zone immediately.

```
cumulus@switch:~$ sudo dpkg-reconfigure --frontend noninteractive
tzdata
```

Alternative: Use the Guided Wizard to Find and Apply a Time Zone

To set the time zone, run `dpkg-reconfigure tzdata` as root:

```
cumulus@switch:~$ sudo dpkg-reconfigure tzdata
```

Then navigate the menus to enable the time zone you want. The following example selects the US/Pacific time zone:

```
cumulus@switch:~$ sudo dpkg-reconfigure tzdata

Configuring tzdata
-----

Please select the geographic area in which you live. Subsequent
configuration
questions will narrow this down by presenting a list of cities,
representing
the time zones in which they are located.

 1. Africa        4. Australia    7. Atlantic   10. Pacific   13. Etc
 2. America       5. Arctic       8. Europe     11. SystemV
 3. Antarctica    6. Asia        9. Indian     12. US

Geographic area: 12

Please select the city or region corresponding to your time zone.
```

```
1. Alaska      4. Central    7. Indiana-Starke 10. Pacific
 2. Aleutian    5. Eastern     8. Michigan       11. Pacific-New
 3. Arizona     6. Hawaii      9. Mountain      12. Samoa
Time zone: 10
```

```
Current default time zone: 'US/Pacific'
Local time is now:      Mon Jun 17 09:27:45 PDT 2013.
Universal Time is now:  Mon Jun 17 16:27:45 UTC 2013.
```

For more info see the Debian [System Administrator's Manual – Time](#).

Setting the Date and Time

The switch contains a battery backed hardware clock that maintains the time while the switch is powered off and in between reboots. When the switch is running, the Cumulus Linux operating system maintains its own software clock.

During boot up, the time from the hardware clock is copied into the operating system's software clock. The software clock is then used for all timekeeping responsibilities. During system shutdown the software clock is copied back to the battery backed hardware clock.

You can set the date and time on the software clock using the `date` command. First, determine your current time zone:

```
cumulus@switch$ date +%Z
```



If you need to reconfigure the current time zone, refer to the instructions above.

Then, to set the system clock according to the time zone configured:

```
cumulus@switch$ sudo date -s "Tue Jan 12 00:37:13 2016"
```

See `man date(1)` for if you need more information.

You can write the current value of the system (software) clock to the hardware clock using the `hwclock` command:

```
cumulus@switch$ sudo hwclock -w
```

See `man hwclock(8)` if you need more information.

You can find a good overview of the software and hardware clocks in the Debian [System Administrator's Manual – Time](#), specifically the section [Setting and showing hardware clock](#).



Setting Time Using NTP

The `ntpd` daemon running on the switch implements the NTP protocol. It synchronizes the system time with time servers listed in `/etc/ntp.conf`. It is started at boot by default. See `man ntpd(8)` for `ntpd` details.

By default, `/etc/ntp.conf` contains some default time servers. Edit `/etc/ntp.conf` to add or update time server information. See `man ntp.conf(5)` for details on configuring `ntpd` using `ntp.conf`.

To set the initial date and time via NTP before starting the `ntpd` daemon, use `ntpd -q` (This is same as `ntpdate`, which is to be retired and not available).



`ntpd -q` can hang if the time servers are not reachable.

To verify that `ntpd` is running on the system:

```
cumulus@switch:~$ ps -ef | grep ntp
ntp      4074      1  0 Jun20 ?          00:00:33 /usr/sbin/ntpd -p /var
/run/ntpd.pid -g -u 101:102
```

To check the NTP peer status:

```
cumulus@switch:~$ ntpq -p

      remote           refid      st t when poll reach   delay
offset jitter
=====
=====
*level1f.cs.unc. .PPS.          1 u    225 1024  377   92.505
-1.296  1.139
+ip.tcp.lv     193.11.166.8    2 u    29 1024  377   192.701
2.424  1.227
-host-86.3.217.2 131.107.13.100  2 u  1024 1024  367   240.622
11.250  7.785
+li290-38.member 128.138.141.172  2 u   553 1024  377   38.944
-0.810  1.139
```

Specifying the NTP Source Interface

You can change the source interface that NTP uses if you want to use something other than the default of `eth0`. Edit `ntp.conf` and edit the entry under the **# Specify interfaces** comment:

```
cumulus@switch:~$ sudo nano /etc/ntp.conf
...
```



```
# Specify interfaces
interface listen bridge10

...
```

NTP Default Configuration

The default NTP configuration comprises the following servers, which are listed in the `/etc/ntp.conf` file:

- server 0.cumulusnetworks.pool.ntp.org iburst
- server 1.cumulusnetworks.pool.ntp.org iburst
- server 2.cumulusnetworks.pool.ntp.org iburst
- server 3.cumulusnetworks.pool.ntp.org iburst

If you need to restore the default NTP configuration, its contents are listed below.

Default ntpd.conf file ...

```
# /etc/ntp.conf, configuration for ntpd; see ntp.conf(5) for help
driftfile /var/lib/ntp/ntp.drift
# Enable this if you want statistics to be logged.
#statsdir /var/log/ntpstats/
statistics loopstats peerstats clockstats
filegen loopstats file loopstats type day enable
filegen peerstats file peerstats type day enable
filegen clockstats file clockstats type day enable
# You do need to talk to an NTP server or two (or three).
#server ntp.your-provider.example
# pool.ntp.org maps to about 1000 low-stratum NTP servers. Your
server will
# pick a different set every time it starts up. Please consider
joining the
# pool: <http://www.pool.ntp.org/join.html>
server 0.cumulusnetworks.pool.ntp.org iburst
server 1.cumulusnetworks.pool.ntp.org iburst
server 2.cumulusnetworks.pool.ntp.org iburst
server 3.cumulusnetworks.pool.ntp.org iburst
# Access control configuration; see /usr/share/doc/ntp-doc/html
/accept.html for
# details. The web page <http://support.ntp.org/bin/view/Support
/AccessRestrictions>
# might also be helpful.
#
# Note that "restrict" applies to both servers and clients, so a
configuration
# that might be intended to block requests from certain clients could
also end
# up blocking replies from your own upstream servers.
```

```
# By default, exchange time with everybody, but don't allow
configuration.
restrict -4 default kod notrap nomodify nopeer noquery
restrict -6 default kod notrap nomodify nopeer noquery
# Local users may interrogate the ntp server more closely.
restrict 127.0.0.1
restrict ::1
# Clients from this (example!) subnet have unlimited access, but only
if
# cryptographically authenticated.
#restrict 192.168.123.0 mask 255.255.255.0 notrust
# If you want to provide time to your local subnet, change the next
line.
# (Again, the address is an example only.)
#broadcast 192.168.123.255
# If you want to listen to time broadcasts on your local subnet, de-
comment the
# next lines. Please do this only if you trust everybody on the
network!
#disable auth
#broadcastclient
# Specify interfaces, don't listen on switch ports
interface listen eth0
```

Related Information

- Debian System Administrator's Manual – Time
- www.ntp.org
- en.wikipedia.org/wiki/Network_Time_Protocol
- wiki.debian.org/NTP

Authentication, Authorization, and Accounting

- SSH for Remote Access (see page 92)
- User Accounts (see page 95)
- Using sudo to Delegate Privileges (see page 96)
- PAM and NSS (see page 102)
- TACACS+ (see page 111)

SSH for Remote Access

Authentication keys can be generated for securely accessing a Cumulus Linux switch with the ssh-keygen component of the Secure Shell (SSH) protocol. Cumulus Linux uses the OpenSSH package to provide this functionality. The section below covers how to generate a SSH key pair.



Contents

This chapter covers ...

- Generate an SSH Key Pair (see page 93)
- Related Information (see page 94)

Generate an SSH Key Pair

1. Run the `ssh-keygen` command, and follow the prompts, to generate the key pair:

ⓘ Configure a Passwordless System

To configure a completely passwordless system, do not enter a passphrase when prompted in the following step.

```
cumulus@leaf01:~$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/cumulus/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/cumulus/.ssh/id_rsa.
Your public key has been saved in /home/cumulus/.ssh/id_rsa.pub.
The key fingerprint is:
5a:b4:16:a0:f9:14:6b:51:f6:f6:c0:76:1a:35:2b:bb cumulus@leaf04
The key's randomart image is:
+---[RSA 2048]---+
|      .o   o |
|     o * o . o |
|    o + o O o |
|     + . = O |
|      . S o . |
|       + . |
|      . E |
+-----+
```

2. Run the `ssh-copy-id` command, and follow the prompts, to copy the generated public key to the desired location:

```
cumulus@leaf01:~$ ssh-copy-id -i /home/cumulus/.ssh/id_rsa.pub
cumulus@leaf02
The authenticity of host 'leaf02 (192.168.0.11)' can't be
established.
```

```
ECDSA key fingerprint is b1:ce:b7:6a:20:f4:06:3a:09:3c:d9:42:de:  
99:66:6e.  
Are you sure you want to continue connecting (yes/no)? yes  
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key  
(s), to filter out any that are already installed  
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed --  
if you are prompted now it is to install the new keys  
cumulus@leaf01's password:  
  
Number of key(s) added: 1
```



ssh-copy-id will not work if the username on the remote switch is different to the local switch. To work around this issue, use the scp command instead:

```
cumulus@leaf01:~$ scp .ssh/id_rsa.pub cumulus@leaf02:.ssh  
/authorized_keys  
Enter passphrase for key '/home/cumulus/.ssh/id_rsa':  
id_rsa.pub
```

3. Connect to the remote switch to confirm the authentication keys are in place:

```
cumulus@leaf04:~$ ssh cumulus@leaf02  
  
Welcome to Cumulus VX (TM)  
  
Cumulus VX (TM) is a community supported virtual appliance  
designed for  
experiencing, testing and prototyping Cumulus Networks' latest  
technology.  
For any questions or technical support, visit our community site  
at:  
http://community.cumulusnetworks.com  
  
The registered trademark Linux (R) is used pursuant to a  
sublicense from LMI,  
the exclusive licensee of Linus Torvalds, owner of the mark on a  
world-wide  
basis.  
Last login: Thu Sep 29 16:56:54 2016
```

Related Information

- Debian Documentation - Password-less logins with OpenSSH
- Wikipedia - Secure Shell (SSH)



User Accounts

By default, Cumulus Linux has two user accounts: *cumulus* and *root*.

The *cumulus* account:

- Default password is *CumulusLinux!*
- Is a user account in the *sudo* group with sudo privileges
- User can log in to the system via all the usual channels like console and SSH (see page 92)
- Along with the *cumulus* group, has both show and edit rights for *NCLU* (see page 80)

The *root* account:

- Default password is disabled by default
- Has the standard Linux root user access to everything on the switch
- Disabled password prohibits login to the switch by SSH, telnet, FTP, and so forth

For best security, you should change the default password (using the `passwd` command) before you configure Cumulus Linux on the switch.

You can add more user accounts as needed. Like the *cumulus* account, these accounts must use `sudo` to execute privileged commands (see page 96), so be sure to include them in the *sudo* group.

To access the switch without any password requires booting into a single shell/user mode (see page 632).

Enabling Remote Access for the root User

As mentioned above, the root user does not have a password set for it, and it cannot log in to a switch via SSH. This default account behavior is consistent with Debian. In order to connect to a switch using the root account, you can do one of two things for the account:

- Generate an SSH key
- Set a password

Generating an SSH Key for the root Account

1. First, in a terminal on your host system (not the switch), check to see if a key already exists:

```
root@host:~# ls -al ~/.ssh/
```

The key is named something like `id_dsa.pub`, `id_rsa.pub` or `id_ecdsa.pub`.

2. If a key doesn't exist, generate a new one by first creating the RSA key pair:

```
root@host:~# ssh-keygen -t rsa
```

3. A prompt appears, asking you to *Enter file in which to save the key (/root/.ssh/id_rsa)*. Press Enter to use the root user's home directory, or else provide a different destination.
4. You are prompted to *Enter passphrase (empty for no passphrase)*: This is optional but it does provide an extra layer of security.



5. The public key is now located in `/root/.ssh/id_rsa.pub`. The private key (identification) is now located in `/root/.ssh/id_rsa`.
6. Copy the public key to the switch. SSH to the switch as the `cumulus` user, then run:

```
cumulus@switch:~$ sudo mkdir -p /root/.ssh
cumulus@switch:~$ echo <SSH public key string> | sudo tee -a
/root/.ssh/authorized_keys
```

Setting the root User Password

1. Run:

```
cumulus@switch:~$ sudo passwd root
```

2. Change the `/etc/ssh/sshd_config` file's `PermitRootLogin` setting from `without-password` to `yes`

```
cumulus@switch:~$ sudo nano /etc/ssh/sshd_config
...
# Authentication:
LoginGraceTime 120
PermitRootLogin yes
StrictModes yes
...
```

3. Restart the `ssh` service:

```
cumulus@switch:~$ sudo systemctl reload ssh.service
```

Using sudo to Delegate Privileges

By default, Cumulus Linux has two user accounts: `root` and `cumulus`. The `cumulus` account is a normal user and is in the group `sudo`.

You can add more user accounts as needed. Like the `cumulus` account, these accounts must use `sudo` to execute privileged commands.

Contents

This chapter covers ...



- Using sudo (see page 97)
- sudoers Examples (see page 97)
- Related Information (see page 102)

Using sudo

`sudo` allows you to execute a command as superuser or another user as specified by the security policy. See `man sudo(8)` for details.

The default security policy is `sudoers`, which is configured using `/etc/sudoers`. Use `/etc/sudoers.d/` to add to the default `sudoers` policy. See `man sudoers(5)` for details.



Use `visudo` only to edit the `sudoers` file; do not use another editor like `vi` or `emacs`. See `man visudo(8)` for details.

Errors in the `sudoers` file can result in losing the ability to elevate privileges to root. You can fix this issue only by power cycling the switch and booting into single user mode. Before modifying `sudoers`, enable the root user by setting a password for the root user.

By default, users in the `sudo` group can use `sudo` to execute privileged commands. To add users to the `sudo` group, use the `useradd(8)` or `usermod(8)` command. To see which users belong to the `sudo` group, see `/etc/group` (`man group(5)`).

Any command can be run as `sudo`, including `su`. A password is required.

The example below shows how to use `sudo` as a non-privileged user `cumulus` to bring up an interface:

```
cumulus@switch:~$ ip link show dev swp1
3: swp1: <BROADCAST,MULTICAST> mtu 1500 qdisc pfifo_fast master br0
state DOWN mode DEFAULT qlen 500
link/ether 44:38:39:00:27:9f brd ff:ff:ff:ff:ff:ff

cumulus@switch:~$ ip link set dev swp1 up
RTNETLINK answers: Operation not permitted

cumulus@switch:~$ sudo ip link set dev swp1 up
Password:

cumulus@switch:~$ ip link show dev swp1
3: swp1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast
master br0 state UP mode DEFAULT qlen 500
link/ether 44:38:39:00:27:9f brd ff:ff:ff:ff:ff:ff
```

sudoers Examples

The following examples show how you grant as few privileges as necessary to a user or group of users to allow them to perform the required task. For each example, the system group `noc` is used; groups are prefixed with an `%`.

When executed by an unprivileged user, the example commands below must be prefixed with `sudo`.

Category	Privilege	Example Command	sudoers Entry
Monitoring	Switch port info	<code>ethtool -m swp1</code>	<code>%noc ALL=(ALL) NOPASSWD:/sbin/ethtool</code>
Monitoring	System diagnostics	<code>cl-support</code>	<code>%noc ALL=(ALL) NOPASSWD:/usr/cumulus/bin/cl-support</code>
Monitoring	Routing diagnostics	<code>cl-resource-query</code>	<code>%noc ALL=(ALL) NOPASSWD:/usr/cumulus/bin/cl-resource-query</code>
Image management	Install images	<code>onie-select http://lab/install.bin</code>	<code>%noc ALL=(ALL) NOPASSWD:/usr/cumulus/bin/onie-select</code>
Package management	Any apt-get command	<code>apt-get update or apt-get install</code>	<code>%noc ALL=(ALL) NOPASSWD:/usr/bin/apt-get</code>
Package management	Just apt-get update	<code>apt-get update</code>	<code>%noc ALL=(ALL) NOPASSWD:/usr/bin/apt-get update</code>
Package management	Install packages		



Category	Privilege	Example Command	sudoers Entry
		<code>apt-get install vim</code>	<code>%noc ALL=(ALL) NOPASSWD:/usr/bin/apt-get install *</code>
Package management	Upgrading	<code>apt-get upgrade</code>	<code>%noc ALL=(ALL) NOPASSWD:/usr/bin/apt-get upgrade</code>
Netfilter	Install ACL policies	<code>cl-acltool -i</code>	<code>%noc ALL=(ALL) NOPASSWD:/usr/cumulus/bin/cl-acltool</code>
Netfilter	List iptables rules	<code>iptables -L</code>	<code>%noc ALL=(ALL) NOPASSWD:/sbin/iptables</code>
L1 + 2 features	Any LLDP command	<code>lldpcli show neighbors / configure</code>	<code>%noc ALL=(ALL) NOPASSWD:/usr/sbin/lldpcli</code>
L1 + 2 features	Just show neighbors	<code>lldpcli show neighbors</code>	<code>%noc ALL=(ALL) NOPASSWD:/usr/sbin/lldpcli show neighbours*</code>
Interfaces	Modify any interface	<code>ip link set dev swp1 {up down}</code>	<code>%noc ALL=(ALL) NOPASSWD:/sbin/ip link set *</code>



Category	Privilege	Example Command	sudoers Entry
Interfaces	Up any interface	<pre>ifup swp1</pre>	<pre>%noc ALL=(ALL) NOPASSWD: /sbin/ifup</pre>
Interfaces	Down any interface	<pre>ifdown swp1</pre>	<pre>%noc ALL=(ALL) NOPASSWD: /sbin/ifdown</pre>
Interfaces	Up/down only swp2	<pre>ifup swp2 / ifdown swp2</pre>	<pre>%noc ALL=(ALL) NOPASSWD: /sbin/ifup swp2,/sbin /ifdown swp2</pre>
Interfaces	Any IP address chg	<pre>ip addr {add del} 192.0.2.1/30 dev swp1</pre>	<pre>%noc ALL=(ALL) NOPASSWD: /sbin/ip addr *</pre>
Interfaces	Only set IP address	<pre>ip addr add 192.0.2.1/30 dev swp1</pre>	<pre>%noc ALL=(ALL) NOPASSWD: /sbin/ip addr add *</pre>
Ethernet bridging	Any bridge command	<pre>brctl addbr br0 / brctl delif br0 swp1</pre>	<pre>%noc ALL=(ALL) NOPASSWD: /sbin/brctl</pre>



Category	Privilege	Example Command	sudoers Entry
Ethernet bridging	Add bridges and ints	<pre>brctl addbr br0 / brctl addif br0 swp1</pre>	<pre>%noc ALL=(ALL) NOPASSWD: /sbin/brctl addbr *,/sbin /brctl addif *</pre>
Spanning tree	Set STP properties	<pre>mstpcctl setmaxage br2 20</pre>	<pre>%noc ALL=(ALL) NOPASSWD: /sbin/mstpcctl</pre>
Troubleshooting	Restart switchd	<pre>systemctl restart switchd. service</pre>	<pre>%noc ALL=(ALL) NOPASSWD:/usr /sbin/service switchd *</pre>
Troubleshooting	Restart any service	<pre>systemctl cron switchd.service</pre>	<pre>%noc ALL=(ALL) NOPASSWD:/usr /sbin/service</pre>
Troubleshooting	Packet capture	<pre>tcpdump</pre>	<pre>%noc ALL=(ALL) NOPASSWD:/usr /sbin/tcpdump</pre>
L3	Add static routes	<pre>ip route add 10.2.0.0/16 via 10.0.0.1</pre>	<pre>%noc ALL=(ALL) NOPASSWD:/bin /ip route add *</pre>
L3			

Category	Privilege	Example Command	sudoers Entry
	Delete static routes	<pre>ip route del 10.2.0.0/16 via 10.0.0.1</pre>	<pre>%noc ALL=(ALL) NOPASSWD:/bin /ip route del *</pre>
L3	Any static route chg	<pre>ip route *</pre>	<pre>%noc ALL=(ALL) NOPASSWD:/bin /ip route *</pre>
L3	Any iproute command	<pre>ip *</pre>	<pre>%noc ALL=(ALL) NOPASSWD:/bin /ip</pre>
L3	Non-modal OSPF	<pre>cl-ospf area 0.0.0.1 range 10.0.0.0/24</pre>	<pre>%noc ALL=(ALL) NOPASSWD:/usr /bin/cl-ospf</pre>

Related Information

- [sudo](#)
- [Adding Yourself to sudoers](#)

LDAP Authentication and Authorization

Cumulus Linux uses Pluggable Authentication Modules (PAM) and Name Service Switch (NSS) for user authentication.

NSS specifies the order of information sources used to resolve names for each service. Using this with authentication and authorization, it provides the order and location used for user lookup and group mapping on the system. PAM handles the interaction between the user and the system, providing login handling, session setup, authentication of users and authorization of user actions.

NSS enables PAM to use LDAP for providing user authentication, group mapping and information for other services on the system.



Contents

This chapter covers ...

- Configuring LDAP Authentication (see page 103)
- Installing libnss-ldapd (see page 103)
- Configuring nslcd.conf (see page 104)
 - Connection (see page 104)
 - Search Function (see page 105)
 - Search Filters (see page 105)
 - Attribute Mapping (see page 105)
 - Example Configuration (see page 105)
- Troubleshooting (see page 106)
 - Using nslcd Debug Mode (see page 106)
 - Common Problems (see page 107)
- Configuring LDAP Authorization (see page 108)
- Active Directory Configuration (see page 109)
- LDAP Verification Tools (see page 109)
 - Identifying a User with the id Command (see page 109)
 - Using getent (see page 109)
 - Using LDAP search (see page 110)
 - LDAP Browsers (see page 111)
- Related Information (see page 111)

Configuring LDAP Authentication

There are 3 common ways of configuring LDAP authentication on Linux:

- libnss-ldap
- libnss-ldapd
- libnss-sss

This chapter covers using `libnss-ldapd` only. From internal testing, this library worked best with Cumulus Linux and was the easiest to configure, automate and troubleshoot.

Installing libnss-ldapd

The `libpam-ldapd` package depends on `nslcd`, so to install `libnss-ldapd`, `libpam-ldapd` and `ldap-utils`, you must run:

```
cumulus@switch:~$ sudo apt-get install libnss-ldapd libpam-ldapd ldap-utils nslcd
```

This brings up an interactive prompt asking questions about the LDAP URI, search base distinguished name (DN) and services that should have LDAP lookups enabled. This creates a very basic LDAP configuration, using anonymous bind, and initiating the search for a user under the base DN specified.



Alternatively, these parameters can be pre-seeded using the `debconf-utils`. To use this method, run `apt-get install debconf-utils` and create the pre-seeded parameters using `debconf-set-selections` with the appropriate answers. Run `debconf-show <pkg>` to check the settings. Here is an [example of how to preseed answers to the installer questions using debconf-set-selections](#).

Once the install is complete, the *name service LDAP caching daemon* (`nslcd`) will be running. This is the service that handles all of the LDAP protocol interactions, and caches the information returned from the LDAP server. In `/etc/nsswitch.conf`, `ldap` has been appended and is the secondary information source for `passwd`, `group` and `shadow`. The local files (`/etc/passwd`, `/etc/groups` and `/etc/shadow`) are used first, as specified by the `compat` source.

```
passwd: compat ldap
group: compat ldap
shadow: compat ldap
```



You are strongly advised to keep `compat` as the first source in NSS for `passwd`, `group` and `shadow`. This prevents you from getting locked out of the system.

Configuring `nslcd.conf`

You need to update the main configuration file (`/etc/nslcd.conf`) after installation to accommodate the expected LDAP server settings. The [nslcd.conf man page](#) details all the available configuration options. Some of the more important options are related to security and how the queries are handled.

Connection

The LDAP client starts a session by connecting to the LDAP server, by default, on TCP and UDP port 389, or on port 636 for LDAPS. Depending on the configuration, this connection may be unauthenticated (anonymous bind); otherwise, the client must provide a bind user and password. The variables used to define the connection to the LDAP server are the URI and bind credentials.

The URI is mandatory, and specifies the LDAP server location using the FQDN or IP address. It also designates whether to use `ldap://` for clear text transport, or `ldaps://` for SSL/TLS encrypted transport. Optionally, an alternate port may also be specified in the URI. Typically, in production environments, it is best to utilize the LDAPS protocol. Otherwise all communications are clear text and not secure.

After the connection to the server is complete, the BIND operation authenticates the session. The BIND credentials are optional, and if not specified, an anonymous bind is assumed. This is typically not allowed in most production environments. Configure authenticated (Simple) BIND by specifying the user (`binddn`) and password (`bindpw`) in the configuration. Another option is to use SASL (Simple Authentication and Security Layer) BIND, which provides authentication services using other mechanisms, like Kerberos. Contact your LDAP server administrator for this information since it depends on the configuration of the LDAP server and what credentials are created for the client device.

```
# The location at which the LDAP server(s) should be reachable.  
uri ldaps://ldap.example.com  
# The DN to bind with for normal lookups.  
binddn cn=CLswitch,ou=infra,dc=example,dc=com  
bindpw CuMuLus
```

Search Function

When an LDAP client requests information about a resource, it must connect and bind to the server. Then it performs one or more resource queries depending on what it is looking up. All search queries sent to the LDAP server are created using the configured search *base*, *filter*, and the desired entry (*uid=myuser*) being searched for. If the LDAP directory is large, this search may take a significant amount of time. It is a good idea to define a more specific search base for the common maps (*passwd* and *group*).

```
# The search base that will be used for all queries.  
base dc=example,dc=com  
# Mapped search bases to speed up common queries.  
base passwd ou=people,dc=example,dc=com  
base group ou=groups,dc=example,dc=com
```

Search Filters

It is also common to use search filters to specify criteria used when searching for objects within the directory. This is used to limit the search scope when authenticating users. The default filters applied are:

```
filter passwd (objectClass=posixAccount)  
filter group (objectClass=posixGroup)
```

Attribute Mapping

The *map* configuration allows for overriding the attributes pushed from LDAP. To override an attribute for a given *map**^{*}, specify the attribute name and the new value. One example of how this is useful is ensuring the shell is *bash* and the home directory is */home/cumulus*:

```
map      passwd homeDirectory "/home/cumulus"  
map      passwd shell "/bin/bash"
```



*In LDAP, the **map** refers to one of the supported maps specified in the manpage for *ns1cd.conf* (such as *passwd* or *group*).



Example Configuration

Here is an [example configuration](#) using Cumulus Linux.

Troubleshooting

Using nslcd Debug Mode

When setting up LDAP authentication for the first time, Cumulus Networks recommends you turn off this service using `sudo systemctl stop nslcd.service` and run it in debug mode. Debug mode works whether you are using LDAP over SSL (port 636) or an unencrypted LDAP connection (port 389).

```
cumulus@switch:~$ sudo systemctl stop nslcd.service
cumulus@switch:~$ sudo nslcd -d
```

Once you enable debug mode, run the following command to test LDAP queries:

```
cumulus@switch:~$ sudo getent myuser
```

If LDAP is configured correctly, the following messages appear after you run the `getent` command:

```
nslcd: DEBUG: accept() failed (ignored): Resource temporarily
unavailable
nslcd: [8elf29] DEBUG: connection from pid=11766 uid=0 gid=0
nslcd: [8elf29] <passwd(all)> DEBUG: myldap_search(base="dc=example,
dc=com", filter="(objectClass=posixAccount)")
nslcd: [8elf29] <passwd(all)> DEBUG: ldap_result(): uid=myuser,
ou=people,dc=example,dc=com
nslcd: [8elf29] <passwd(all)> DEBUG: ldap_result(): ... 152 more
results
nslcd: [8elf29] <passwd(all)> DEBUG: ldap_result(): end of results
(162 total)
```

In the output above, `<passwd(all)>` indicates that the entire directory structure was queried.

A specific user can be queried using the command:

```
cumulus@switch:~$ sudo getent passwd myuser
```

You can replace `myuser` with any username on the switch. The following debug output indicates that user `myuser` exists:

```
nslcd: DEBUG: add_uri(ldap://10.50.21.101)
nslcd: version 0.8.10 starting
```



```
nslcd: DEBUG: unlink() of /var/run/nslcd/socket failed (ignored): No such file or directory
nslcd: DEBUG: setgroups(0,NULL) done
nslcd: DEBUG: setgid(110) done
nslcd: DEBUG: setuid(107) done
nslcd: accepting connections
nslcd: DEBUG: accept() failed (ignored): Resource temporarily unavailable
nslcd: [8b4567] DEBUG: connection from pid=11369 uid=0 gid=0
nslcd: [8b4567] <passwd="myuser"> DEBUG: myldap_search(base="dc=cumulusnetworks,dc=com", filter="(&(objectClass=posixAccount)(uid=myuser))")
nslcd: [8b4567] <passwd="myuser"> DEBUG: ldap_initialize(ldap://<ip_address>)
nslcd: [8b4567] <passwd="myuser"> DEBUG: ldap_set_rebind_proc()
nslcd: [8b4567] <passwd="myuser"> DEBUG: ldap_set_option(LDAP_OPT_PROTOCOL_VERSION,3)
nslcd: [8b4567] <passwd="myuser"> DEBUG: ldap_set_option(LDAP_OPT_DEREF,0)
nslcd: [8b4567] <passwd="myuser"> DEBUG: ldap_set_option(LDAP_OPT_TIMELIMIT,0)
nslcd: [8b4567] <passwd="myuser"> DEBUG: ldap_set_option(LDAP_OPT_TIMEOUT,0)
nslcd: [8b4567] <passwd="myuser"> DEBUG: ldap_set_option(LDAP_OPT_NETWORK_TIMEOUT,0)
nslcd: [8b4567] <passwd="myuser"> DEBUG: ldap_set_option(LDAP_OPT_REFERRALS,LDAP_OPT_ON)
nslcd: [8b4567] <passwd="myuser"> DEBUG: ldap_set_option(LDAP_OPT_RESTART,LDAP_OPT_ON)
nslcd: [8b4567] <passwd="myuser"> DEBUG: ldap_simple_bind_s(NULL,NULL) (uri="ldap://<ip_address>")
nslcd: [8b4567] <passwd="myuser"> DEBUG: ldap_result(): end of results (0 total)
```

Notice how the <passwd="myuser"> shows that the specific *myuser* user was queried.

Common Problems

SSL/TLS

- The FQDN of the LDAP server URI does not match the FQDN in the CA-signed server certificate exactly.
- ns lcd cannot read the SSL certificate, and will report a "Permission denied" error in the debug during server connection negotiation. Check the permission on each directory in the path of the root SSL certificate. Ensure that it is readable by the ns lcd user.

NSCD

- If the ns cd cache daemon is also enabled and you make some changes to the user from LDAP, you may want to clear the cache using the commands:



```
nsqd --invalidate = passwd  
nsqd --invalidate = group
```

- The `nsqd` package works with `nslcd` to cache name entries returned from the LDAP server. This may cause authentication failures. To work around these issues:

1. Disable `nsqd` by running:

```
cumulus@switch:~$ sudo nsqd -K
```

2. Restart the `nslcd` service:

```
cumulus@switch:~$ sudo systemctl restart nslcd.service
```

3. Try the authentication again.

LDAP

- The search filter returns wrong results. Check for typos in the search filter. Use `ldapsearch` to test your filter.
- Optionally, configure the basic LDAP connection and search parameters in `/etc/ldap/ldap.conf`.

```
# ldapsearch -D 'cn=CLadmin' -w 'CuMuLuS' "(&  
(ObjectClass=inetOrgUser)(uid=myuser))"
```

- When a local username also exists in the LDAP database, the order of the information sources in `/etc/nsswitch` can be updated to query LDAP before the local user database. This is generally not recommended. For example, the configuration below ensures that LDAP is queried before the local database.

```
# /etc/nsswitch.conf  
passwd:      ldap  compat
```

Configuring LDAP Authorization

Linux uses the `sudo` command to allow non-administrator users — like the default `cumulus` user account — to perform privileged operations. To control the users authorized to use `sudo`, the `/etc/sudoers` file and files located in the `/etc/sudoers.d/` directory have a series of rules defined. Typically, the rules are based on groups, but can also be defined for specific users. Therefore, `sudo` rules can be added using the group names from LDAP. For example, if a group of users were associated with the group `netadmin`, a rule can be added to give those users `sudo` privileges. Refer to the `sudoers` manual (`man sudoers`) for a complete usage description. Here's an illustration of this in `/etc/sudoers`:



```
# The basic structure of a user specification is "who where =
(as_whom) what".
%sudo ALL=(ALL:ALL) ALL
%netadmin ALL=(ALL:ALL) ALL
```

Active Directory Configuration

Active Directory (AD) is a fully featured LDAP-based NIS server created by Microsoft. It offers unique features that classic OpenLDAP servers lack. Therefore, it can be more complicated to configure on the client and each version of AD is a little different in how it works with Linux-based LDAP clients. Some more advanced configuration examples, from testing LDAP clients on Cumulus Linux with Active Directory (AD/LDAP), are available in our [knowledge base](#).

LDAP Verification Tools

Typically, password and group information is retrieved from LDAP and cached by the LDAP client daemon. To test the LDAP interaction, these command line tools can be used to trigger an LDAP query from the device. This helps to create the best filters and verify the information sent back from the LDAP server.

Identifying a User with the id Command

The `id` command performs a username lookup by following the lookup information sources in NSS for the `passwd` service. This simply returns the user ID, group ID and the group list retrieved from the information source. In the following example, the user `cumulus` is locally defined in `/etc/passwd`, and `myuser` is on LDAP. The NSS configuration has the `passwd` map configured with the sources `compat ldap`:

```
cumulus@switch:~$ id cumulus
uid=1000(cumulus) gid=1000(cumulus) groups=1000(cumulus),24(cdrom),25
(floppy),27(sudo),29(audio),30(dip),44(video),46(plugdev)
cumulus@switch:~$ id myuser
uid=1230(myuser) gid=3000(Development) groups=3000(Development),500
(Employees),27(sudo)
```

Using getent

The `getent` command retrieves all records found via NSS for a given map. It can also get a specific entry under that map. Tests can be done with the `passwd`, `group`, `shadow` or any other map configured in `/etc/nsswitch.conf`. The output from this command is formatted according to the map requested. Thus, for the `passwd` service, the structure of the output is the same as the entries in `/etc/passwd`. The same can be said for the `group` map will output the same as `/etc/group`. In this example, looking up a specific user in the `passwd` map, the user `cumulus` is locally defined in `/etc/passwd`, and `myuser` is only in LDAP.

```
cumulus@switch:~$ getent passwd cumulus
cumulus:x:1000:1000::/home/cumulus:/bin/bash
cumulus@switch:~$ getent passwd myuser
myuser:x:1230:3000:My Test User:/home/myuser:/bin/bash
```

In the next example, looking up a specific group in the group service, the group *cumulus* is locally defined in `/etc/groups`, and *netadmin* is on LDAP.

```
cumulus@switch:~$ getent group cumulus
cumulus:x:1000:
cumulus@switch:~$ getent group netadmin
netadmin:*:502:larry,moe,curly,shemp
```

Running the command `getent passwd` or `getent group` without a specific request, returns **all** local and LDAP entries for the *passwd* and *group* maps, respectively.

Using LDAP search

The `ldapsearch` command performs LDAP operations directly on the LDAP server. This does not interact with NSS. This command helps display what the LDAP daemon process is receiving back from the server. The command has many options. The simplest uses anonymous bind to the host and specifies the search DN and what attribute to lookup.

```
cumulus@switch:~$ ldapsearch -H ldap://ldap.example.com -b dc=example,
dc=com -x uid=myuser
```

Click to expand the command output ...

```
# extended LDIF
#
# LDAPv3
# base <dc=example,dc=com> with scope subtree
# filter: uid=myuser
# requesting: ALL
#
# myuser, people, example.com
dn: uid=myuser,ou=people,dc=example,dc=com
cn: My User
displayName: My User
gecos: myuser
gidNumber: 3000
givenName: My
homeDirectory: /home/myuser
initials: MU
loginShell: /bin/bash
mail: myuser@example.com
objectClass: inetOrgPerson
objectClass: posixAccount
objectClass: shadowAccount
objectClass: top
shadowExpire: -1
shadowFlag: 0
shadowMax: 999999
```

```
shadowMin: 8
shadowWarning: 7
sn: User
uid: myuser
uidNumber: 1234
# search result
search: 2
result: 0 Success
# numResponses: 2
# numEntries: 1
```

LDAP Browsers

There are some GUI LDAP clients that help to work with LDAP servers. These are free tools to help graphically show the structure of the LDAP database.

- [Apache Directory Studio](#)
- [LDAPManager](#)

Related Information

- [Debian - configuring LDAP authentication](#)
- [Debian - configuring PAM to use LDAP](#)
- [GitHub - Arthur de Jong nslcd.conf file](#)
- [Debian backports](#)

TACACS Plus

Cumulus Linux implements TACACS+ client AAA (Accounting, Authentication, and Authorization) in a transparent way with minimal configuration. There is no need to create accounts or directories on the switch. Authentication is handled via PAM, and includes login, ssh, sudo, and su. Accounting records are sent to all configured TACACS+ servers by default. Use of per-command authorization requires additional setup on the switch.

By default, TACACS+ privilege 15 users are allowed to run any command with sudo via the `/etc/sudoers.d/tacplus` file that is installed by the libtacplus-map1 package.

Contents

This chapter covers ...

- [Installing the TACACS+ Client Packages \(see page 111\)](#)
- [Configuring the TACACS+ Client \(see page 112\)](#)
- [TACACS+ Authentication \(login\) \(see page 113\)](#)
- [TACACS+ Accounting \(see page 114\)](#)
- [TACACS+ Per-command Authorization \(see page 115\)
 - \[Command Options \\(see page 115\\)\]\(#\)
 - \[NSS Plugin \\(see page 116\\)\]\(#\)
 - \[TACACS Configuration Parameters \\(see page 117\\)\]\(#\)](#)



- [Removing the TACACS+ Client Packages \(see page \)](#)
- [Troubleshooting TACACS+ \(see page \)](#)
 - [Debugging Basic Server Connectivity or NSS Issues \(see page 118\)](#)
 - [Debugging Issues with Per-command Authorization \(see page 119\)](#)
 - [Debug Issues with Accounting Records \(see page 120\)](#)
 - [TACACS Component Software Descriptions \(see page 120\)](#)
- [Limitations \(see page 121\)](#)

Installing the TACACS+ Client Packages

TACACS+ requires the following packages to be installed on Cumulus Linux. They are not part of the base Cumulus Linux image installation. All required packages can be installed easily with these commands:

```
cumulus@switch:~$ sudo apt-get update
cumulus@switch:~$ sudo apt-get install tacplus-client
```

Configuring the TACACS+ Client

Post-installation TACACS+ configuration requires (at minimum) editing only one file, `/etc/tacplus_servers`. It is necessary add at least one server, and usually one shared secret (key). The `server` and `secret` parameters can be given in any order, and must not include any whitespace (spaces or tabs), and can be added anywhere in the file. For example, if your TACACS+ server IP address is `192.168.0.30`, and your shared secret is `tacacskey` then you would add these parameters to `/etc/tacplus_servers`:

```
secret=tacacskey
server=192.168.0.30
```

Up to 7 TACACS+ servers are supported. Connections are made in the order in which they are listed in this file. In most cases, no other parameters need to be changed. All parameters used by any of the packages can be added to this file, and will affect all the TACACS+ client software. It is also possible to configure some of the packages through individual configuration files. For example, the timeout value (see description below) is set to 5 seconds by default for NSS lookups in `/etc/tacplus_nss.conf`, while other packages use a value of 10 seconds, set in `/etc/tacplus_servers`.

When TACACS+ servers or secrets are added or removed, `auditd` must be restarted (with `systemctl restart auditd`) or a signal must be sent (with `killall -HUP audisp-tacplus`) before `audisp-tacplus` will reread the configuration to see the changed server list. Usually this is an issue only at first change to the configuration.

At this point, the Cumulus Linux switch should be able to query the TACACS server.

This is the complete list of the TACACS+ client configuration files, and their use. The full list of TACACS+ parameters is below at **TACACS Parameters** below.

Filename	Description
<code>/etc/tacplus_servers</code>	



Filename	Description
	This is the primary file that requires configuration post-installation, and is used by all packages via <code>include=/etc/tacplus_servers</code> parameters in the other configuration files, when installed. Since this is usually the file with the shared secrets, it should not be world readable (should be Linux file mode 600).
<code>/etc/nsswitch.conf</code>	When the <code>libnss_tacplus</code> package is installed, this file is configured to enable tacplus lookups via <code>libnss_tacplus</code> . If this file is replaced by automation or other means, you will need to add tacplus as the first lookup method for the <code>passwd</code> database line.
<code>/etc/tacplus_nss.conf</code>	This file sets the basic parameters for <code>libnss_tacplus</code> . It includes a debug variable for debugging NSS lookups separately from other client packages.
<code>/usr/share/pam-configs/tacplus</code>	Configuration file for <code>pam-auth-update</code> to generate the files in the next row. These configurations are used at <code>login</code> , by <code>su</code> , and by <code>ssh</code> .
<code>/etc/pam.d/common-*</code>	The <code>/etc/pam.d/common-*</code> files are updated for <code>tacplus</code> authentication. The files are updated with <code>pam-auth-update</code> , when <code>libpam-tacplus</code> is installed or removed.
<code>/etc/sudoers.d/tacplus</code>	This file allows TACACS+ privilege level 15 users to run commands with <code>sudo</code> . It also includes an example (commented out) showing how to enable privilege level 15 TACACS users to use <code>sudo</code> without having to enter a password, and an example of how to enable all TACACS users to run specific commands via <code>sudo</code> . It should only be edited with the command: <code>visudo -f /etc/sudoers.d/tacplus</code>
<code>audisp-tacplus.conf</code>	<code>audisp</code> plugin configuration file. In general, no modifications are required.
<code>/etc/audisp/audisp-tac_plus.conf</code>	TACACS+ server configuration file for accounting. In general, no modifications are required. It may be useful to use this configuration file when you only want to debug TACACS+ accounting issues, not all TACACS+ users.
<code>/etc/audit/rules.d/audisp-tacplus.rules</code>	The <code>audited</code> rules for TACACS+ accounting. The <code>augenrules</code> command uses all rules files to generate the file rules file (below)
<code>/etc/audit/audit.rules</code>	Audit rules file generated during <code>audited</code> installation.



The `/etc/pam.d/common-*` files can be edited manually. However, if `pam-auth-update` is run again after the changes are made, the update will fail. Configuration should be done in `/usr/share/pam-configs/tacplus` instead, followed by running `pam-auth-update`.

TACACS+ Authentication (login)

The initial authentication configuration is done through the PAM modules, and an updated version of the `libpam-tacplus` package. When the package is installed, the PAM configuration is updated in `/etc/pam.d` with the `pam-auth-update` command. If you have made changes to your PAM configuration, you may need to integrate these changes yourself. If you are also using LDAP with the `libpam-ldap` package, you will need to edit the PAM configuration to ensure the LDAP and TACACS ordering that you prefer. The `libpam-tacplus` are configured to skip over rules, and the values in the `success=2` may require adjustments to skip over LDAP rules.



TACACS+ users at privilege levels other than 15 are not allowed to run `sudo` commands by default, and are limited to commands that can be run with standard Linux user permissions.

TACACS+ Accounting

TACACS+ accounting is implemented with the `audisp` module, with an additional plugin for `auditd`/`audisp`. The plugin maps the `uid` in the accounting record to a TACACS login, based on the `uid` and `sessionid`. The `audisp` module requires `libnss_tacplus`, and uses the `libtacplus_map.so` library interfaces as part of the modified `lipam_tacplus` package.

Communication with the TACACS+ servers is done via the `libsimple-tacact1` library, through `dlopen()`. A maximum of 240 bytes of command name and arguments are sent in the accounting record, due to the TACACS+ field length limitation of 255 bytes.



All Linux commands result in an accounting record, including commands run as part of the login process or as a sub-processes of other commands. This can sometimes generate a large number of accounting records.

The IP address and encryption key of the server should be configured in the `/etc/tacplus_servers` file. Minimal configuration to `auditd` and `audisp` is necessary to enable the audit records necessary for accounting. These records are installed as part of the package.

`audisp-tacplus` installs the audit rules for command accounting. Modifying the configuration files is not usually necessary. However, when a [management VRF \(see page 593\)](#) is configured, the accounting configuration does need special modification, because the `auditd` service starts prior to networking. It is necessary add the `vrf` parameter, and to signal the `audisp-tacplus` process to reread the configuration. The example below shows that the management VRF is named `mgmt`. The `vrf` parameter can be placed in either `/etc/tacplus_servers` or in `/etc/audisp/audisp-tac_plus.conf`.

```
vrf=mgmt
```

After editing the configuration file, notify the accounting process to reread it by sending the **HUP** signal: `killall -HUP audisp-tacplus`.



All `sudo` commands run by TACACS+ users generate accounting records against the original TACACS+ login name.



For more information, refer to the `audisp.8` and `auditd.8` man pages.

TACACS+ Per-command Authorization

Per-command authorization is handled with the `tacplus-auth` command. To make this an enforced authorization, the TACACS+ login must be changed to use a restricted shell, with a very limited executable search path. Otherwise, the user can bypass the authorization. The `tacplus-restrict` utility simplifies the setup of the restricted environment. The example below initializes the environment for the `tacacs0` user account. This is the account used for TACACS+ users at privilege level 0.

```
tacuser0@switch:~$ sudo tacplus-restrict -i -u tacacs0 -a command1  
command2 ... commandN
```

Command Options

Option	Description
-i	Initializes the environment. It only needs to be issued once per username.
-a	The utility can be invoked with the <code>-a</code> option as many times as desired. For each command in the <code>-a</code> list, a symbolic link is created from <code>tacplus-auth</code> to the relative portion of the command name in the local <code>bin</code> subdirectory. These commands also need to be enabled on the TACACS+ server. See the server documentation for how to do that. It is common to have the server allow some options to a command, but not others.
-f	Re-initializes the environment. If you need to start over, issue the <code>-f</code> option with the <code>-i</code> to force the re-initialization; otherwise, repeated use of <code>-i</code> is ignored. As part of the initialization: <ul style="list-style-type: none">• The user's shell is changed to <code>/bin/rbash</code>.• Any existing dot files are saved.• A limited environment is set up that does not allow general command execution, but instead allows only commands from the user's local <code>bin</code> subdirectory.

As a full example, if you want to allow the user to be able to run the `net` and `ip` commands (potentially, if the TACACS+ server authorizes), use the command:

```
cumulus@switch:~$ sudo tacplus-restrict -i -u tacacs0 -a ip net
```

After running this command, examining the `tacacs0` directory should show something similar to the following:

```
cumulus@switch:~$ sudo ls -lR ~tacacs0  
total 12  
lrwxrwxrwx 1 root root 22 Nov 21 22:07 ip -> /usr/sbin/tacplus-auth  
lrwxrwxrwx 1 root root 22 Nov 21 22:07 net -> /usr/sbin/tacplus-auth
```

Other than shell built-ins, the only two commands the privilege level 0 TACACS users can run are the `ip` and `net` commands. If the user/command combination is not authorized by the TACACS+ server, a message similar to the following gets displayed:

```
tacuser0@switch:~$ net pending  
net not authorized by TACACS+ with given arguments, not executing
```



The `net` command has its own configuration file to enable users to run the `net` command. For the above command to enable TACACS+ privilege level 0 users to run the `net show` commands, edit the file `/etc/netd.conf` and add `tacacs0` to the `users_with_show` line. See the [NCLU \(see page 80\)](#) documentation for more details.

If you mistakenly add potential commands with the `-a` option, you can remove the commands that you don't want (the example below shows the `net` command):

```
cumulus@switch:~$ sudo rm ~tacacs0/bin/net
```

Or you can remove all the commands with:

```
cumulus@switch:~$ sudo rm ~tacacs0/bin/*
```

Use the `man` command on the switch for more information on `tacplus-auth` and `tacplus-restrict`.

```
cumulus@switch:~$ man tacplus-auth tacplus-restrict
```

NSS Plugin

When used with `pam_tacplus`, TACACS+ authenticated users are able to log in without a local account on the system via the NSS plugin that comes with the `tacplus_nss` package. The plugin uses the mapped `tacplus` information if the user is not found in the local password file, provides the `getpwnam()` and `getpwuid()` entry points and uses the TACACS+ authentication functions.

The plugin asks the TACACS+ server if the user is known, and then for relevant attributes to determine the user's privilege level. When the `libnss_tacplus` package is installed, `nsswitch.conf` is modified to set `tacplus` as the first lookup method for `passwd`. If the order is changed, lookups will return the local accounts such as `tacacs0`.

If the user is not found, a mapped lookup is performed using the `libtacplus_map.so` exported functions. The privilege level is appended to "tacacs", and the lookup searches for the name in the local password file. For example, privilege level 15 will search for the `tacacs15` user. If the user is found, the password structure is filled in with the user's information.



If it is not found, the privilege level is decremented and checked again, until privilege level 0 (user `tacacs0`) is reached. This allows use of only the two local users `tacacs0` and `tacacs15`, if minimal configuration is desired.

TACACS Configuration Parameters

The recognized configuration options are the same as the `libpam_tacplus` command line arguments; not all `pam_tacplus` options are supported, however. The table below describes the configuration options available:

Configuration Option	Description
<code>debug</code>	<p>Output debugging information via <code>syslog(3)</code>.</p> <div style="border: 2px solid #f0ad8e; padding: 10px; margin-top: 10px;">! Debugging is heavy, including passwords. Do not leave debugging enabled on a production switch once you have completed troubleshooting.</div>
<code>secret=STRING</code>	<p>Secret key used to encrypt/decrypt packets sent to/received from the server. Can be specified more than once, and can be in any order with respect to the <code>server</code>= parameter. When fewer <code>secret=</code> parameters are specified, the last secret given is used for the remaining servers. This parameter should only be put into files such as <code>/etc/tacplus_servers</code> that are not world readable.</p>
<code>server=HOSTNAME</code> <code>server=IP_ADDR</code>	<p>Adds a TACACS+ server to the servers list. Servers will be queried in turn until a match is found, or no servers remain in the list. Can be specified up to 7 times. When the <code>IP_ADDR</code> form is used, it can be optionally followed by a port number, preceded by a ":". The default port is 49.</p> <div style="border: 2px solid #f0ad8e; padding: 10px; margin-top: 10px;">! When sending accounting records, the record is sent to all servers in the list if <code>acct_all=1</code>, which is the default.</div>
<code>timeout=SECONDS</code>	TACACS+ server(s) communication timeout. The default value is 5 seconds.
<code>login=STRING</code>	TACACS+ authentication service (<code>pap</code> , <code>chap</code> , or <code>login</code>). The default value is <code>pap</code> .
<code>acct_all=1</code>	Configuration option for <code>audisp_tacplus</code> and <code>pam_tacplus</code> sending accounting records to all supplied servers (1), or the first server to respond (0). The default value is 1.
<code>timeout=SECS</code>	Sets the timeout in seconds for connections to each TACACS+ server. The default is 10 seconds for all lookups except that NSS lookups use a 5 second timeout.
<code>vrf=VRFNAME</code>	

Configuration Option	Description
	If the management network is in a VRF, set this variable to the VRF name. This would usually be "mgmt". When this variable is set, the connection to the TACACS+ accounting servers is made through the named VRF.
service	TACACS+ accounting and authorization service. Examples include shell, pap, raccess, ppp, and slip. The default value is <i>shell</i> .
protocol	TACACS+ protocol field. This option is use dependent. PAM uses the SSH protocol.

Removing the TACACS+ Client Packages

To remove all of the TACACS+ client packages, use the following commands:

```
cumulus@switch:~$ sudo apt-get remove tacplus-client
cumulus@switch:~$ sudo apt-get autoremove
```

To remove the TACACS+ client configuration files as well as the packages (recommended), use this command:

```
cumulus@switch:~$ sudo apt-get autoremove --purge
```

Troubleshooting TACACS+

Debugging Basic Server Connectivity or NSS Issues

The `getent` command can be used to determine whether TACACS+ is configured correctly, and the local password is stored in the configuration files. In the example commands below, the `cumulus` user represents the local user, while `cumulusTAC` represents the TACACS user.

To look up the username within all NSS methods:

```
cumulus@switch:~$ sudo getent password cumulusTAC
cumulusTAC:x:1016:1001:TACACS+ mapped user at privilege level 15,,,:
/home/tacacs15:/bin/bash
```

To look up the user within the local database only:

```
cumulus@switch:~$ sudo getent -s compat passwd cumulus
```



```
cumulus:x:1000:1000:cumulus,,,:/home/cumulus:/bin/bash
```

To look up the user within the TACACS+ database only:

```
cumulus@switch:~$ sudo getent -s tacplus passwd cumulusTAC
cumulusTAC:x:1016:1001:TACACS+ mapped user at privilege level 15,,,:
/home/tacacs15:/bin/bash
```

If TACACS does not appear to be working correctly, the following configuration files should be debugged by adding the `debug=1` parameter to one or more of these files:

- `/etc/tacplus_servers`
- `/etc/tacplus_nss.conf`



`debug=1` can also be added to individual `pam_tacplus` lines in `/etc/pam.d/common*`.

All log messages are stored in `/var/log/syslog`.

Debugging Issues with Per-command Authorization

To debug TACACS user command authorization, have the TACACS+ user enter the following command at a shell prompt, and then try the command again:

```
tacuser0@switch:~$ export TACACSAUTHDEBUG=1
```

When this debugging is enabled, additional information is shown for the command authorization conversation with the TACACS+ server:

```
tacuser0@switch:~$ net pending
tacplus-auth: found matching command (/usr/bin/net) request
authorization
tacplus-auth: error connecting to 10.0.3.195:49 to request
authorization for net: Transport endpoint is not connected
tacplus-auth: cmd not authorized (16)
tacplus-auth: net not authorized from 192.168.3.189:49
net not authorized by TACACS+ with given arguments, not executing

tacuser0@switch:~$ net show version
tacplus-auth: found matching command (/usr/bin/net) request
authorization
tacplus-auth: error connecting to 10.0.3.195:49 to request
authorization for net: Transport endpoint is not connected
tacplus-auth: 192.168.3.189:49 authorized command net
tacplus-auth: net authorized, executing
DISTRIB_ID="Cumulus Linux"
DISTRIB_RELEASE=3.2.1
```



```
DISTRIB_DESCRIPTION="Cumulus Linux 3.2.1"
```

To disable debugging:

```
tacuser0@switch:~$ export -n TACACSAUTHDEBUG
```

Debug Issues with Accounting Records

If TACACS+ servers have been added or deleted from the configuration files, make sure you notify the audisp plugin with this command:

```
cumulus@switch:~$ sudo killall -HUP audisp-tacplus
```

If accounting records still are not being sent, add `debug=1` to the file `/etc/audisp/audisp-tac_plus.conf`, and then issue the command above to notify the plugin. Then have the TACACS+ user run a command, and examine the end of `/var/log/syslog` for messages from the plugin. You can also check the auditing log file `/var/log/audit/audit.log` to be sure the auditing records are being written. If they are not, restart the audit daemon with:

```
cumulus@switch:~$ sudo systemctl restart auditd.service
```

TACACS Component Software Descriptions

These different pieces of software are involved with delivering TACACS. Provided below is a brief description of their functionalities.

Package Name	Description
audisp-tacplus_1.0.0-1-cl3u3	This package uses auditing data from <code>auditd</code> to send accounting records to the TACACS+ server and is started as part of <code>auditd</code> .
libtac2_1.4.0-cl3u2	Basic TACACS+ server utility and communications routines.
libnss-tacplus_1.0.1-cl3u3	Provides an interface between <code>libc</code> username lookups, the mapping functions, and the TACACS+ server.
tacplus-auth-1.0.0-cl3u1	This package provides the ability to do per-command TACACS+ authorization, and a setup utility <code>tacplus-restrict</code> to enable that. Per-command authorization is not done by default.



Package Name	Description
libpam-tacplus_1.4.0-cl3u2	A modified version of the standard Debian package.
libtacplus-map1_1.0.0-cl3u2	The mapping functionality between local and TACACS+ users on the server. Sets the immutable <code>sessionid</code> and auditing UID to ensure the original user can be tracked through multiple processes and privilege changes. Sets the auditing <code>loginuid</code> as immutable if supported. Creates and maintains a status database in <code>/run/tacacs_client_map</code> to manage and lookup mappings.
libsimple-tacacct1_1.0.0-cl3u2	Provides an interface for programs to send accounting records to the TACACS+ server. Used by <code>audisp-tacplus</code> .
libtac2-bin_1.4.0-cl3u2	Provides the “tacc” testing program and TACACS+ man page.

Limitations

If two or more TACACS+ users are logged in simultaneously, with the same privilege level, while the accounting records are maintained correctly, a lookup on either name will match both users, while a UID lookup will only return the user that logged in first.

This means that any processes run by either user will be attributed to both, and all files created by either user will be attributed to the first name matched. This is similar to adding two local users to the password file with the same UID and GID, and is an inherent limitation of using the UID for the base user from the password file.



The current algorithm returns the first name matching the UID from the mapping file; this could be the first or second user that logged in.

To work around this issue, the switch’s audit log or the TACACS server accounting logs can be used to determine which processes and files were created by each user.

- For commands that do not execute other commands (for example, changes to configurations in an editor, or actions with tools like `clagctl` and `vtysh`), no additional accounting is done.
- Per-command authorization is not implemented in this release except at the most basic level (commands are permitted or denied based on the standard Linux user permissions for the local TACACS users, and only privilege level 15 users can run `sudo` commands by default).

The Linux `auditd` system does not always generate audit events for processes when terminated with a signal (via the `kill` system call or internal errors such as `SIGSEGV`). As a result, processes that exit on a signal that isn’t caught and handled may not generate a STOP accounting record.

Netfilter - ACLs

Netfilter is the packet filtering framework in Cumulus Linux as well as most other Linux distributions. There are a number of tools available for configuring ACLs in Cumulus Linux, including:

- `iptables`, `ip6tables` and `ebtables` are Linux userspace tools to administer filtering rules for IPv4 packets, IPv6 packets and Ethernet frames (layer 2 using MAC addresses) respectively.
- [NCLU \(see page 80\)](#), a Cumulus Linux-specific userspace tool for configuring custom ACLs.
- `cl-acltool`, another Cumulus Linux-specific userspace tool to administer filtering rules and for configuring the default ACLs.

NCLU and `cl-acltool` operate on various configuration files, and use `iptables`, `ip6tables` and `ebtables` to install rules into the kernel. In addition to programming rules in the kernel, NCLU and `cl-acltool` program rules in hardware for interfaces involving switch port interfaces, which `iptables`, `ip6tables` and `ebtables` cannot do on their own.



In many instances, you can use NCLU to configure ACLs; however, `cl-acltool` must be used in some cases. The examples below show when to use which tool.



If you need help getting started setting up ACLs, run `net example acl` to see a basic configuration:

Click to see the example ...

```
cumulus@leaf01:~$ net example acl
Scenario
=====
We would like to use access-lists on 'switch' to
- Restrict inbound traffic on swp1 to traffic from 10.1.1.0/24
destined for 10.1.2.0/24
- Restrict outbound traffic on swp2 to http, https, or ssh
  *switch
    \
    swp1 /   \ swp2
      /     \
      /       \
      host-11  host-12
switch net commands
=====
Create an ACL that accepts traffic from 10.1.1.0/24 destined
for 10.1.2.0/24
and drops all other traffic
switch# net add acl ipv4 MYACL accept source-ip 10.1.1.0/24
dest-ip 10.1.2.0/24
switch# net add acl ipv4 MYACL drop source-ip any dest-ip any
Apply MYACL inbound on swp1
switch# net add interface swp1 acl ipv4 MYACL inbound
```



```
Create an ACL that accepts http, https, or ssh traffic and
drops all
other traffic.

switch# net add acl ipv4 WEB_OR_SSH accept tcp source-ip any
source-port any dest-ip any dest-port http
switch# net add acl ipv4 WEB_OR_SSH accept tcp source-ip any
source-port http dest-ip any dest-port any
switch# net add acl ipv4 WEB_OR_SSH accept tcp source-ip any
source-port any dest-ip any dest-port https
switch# net add acl ipv4 WEB_OR_SSH accept tcp source-ip any
source-port https dest-ip any dest-port any
switch# net add acl ipv4 WEB_OR_SSH accept tcp source-ip any
source-port any dest-ip any dest-port ssh
switch# net add acl ipv4 WEB_OR_SSH accept tcp source-ip any
source-port ssh dest-ip any dest-port any
switch# net add acl ipv4 WEB_OR_SSH drop source-ip any dest-ip
any
Apply WEB_OR_SSH outbound on swp2
switch# net add interface swp2 acl ipv4 WEB_OR_SSH outbound
commit the staged changes
switch# net commit
Verification
=====
switch# net show configuration acl
```

Contents

This chapter covers ...

- Understanding Traffic Rules In Cumulus Linux (see page 125)
 - Understanding Chains (see page 125)
 - Understanding Tables (see page 126)
 - Understanding Rules (see page 128)
 - How Rules Are Parsed and Applied (see page 129)
 - Rule Placement in Memory (see page 130)
 - Enabling Nonatomic Updates (see page 131)
 - Using iptables/ip6tables/ebtables Directly (see page 132)
 - Estimating the Number of Rules (see page 132)
- Installing and Managing ACL Rules with NCLU (see page 133)
- Installing and Managing ACL Rules with cl-acltool (see page 135)
- Installing Packet Filtering (ACL) Rules (see page 136)
- Specifying which Policy Files to Install (see page 138)
- Hardware Limitations on Number of Rules (see page 138)
 - Broadcom Tomahawk Limits (see page 139)
 - Broadcom Trident II+ Limits (see)



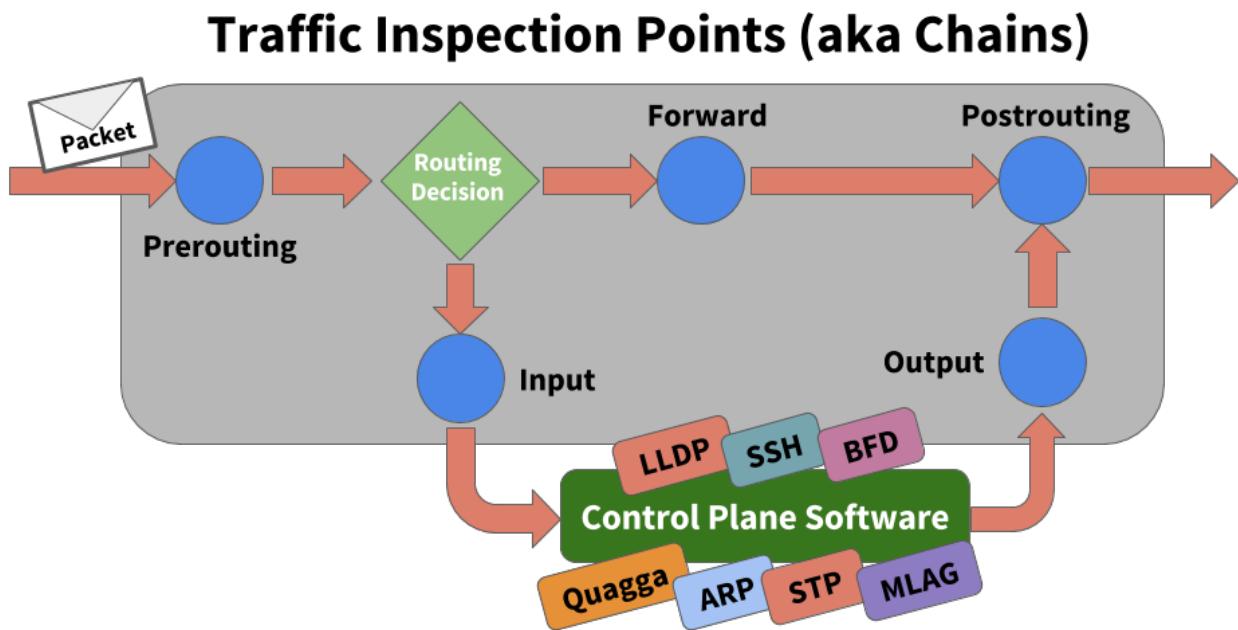
- Broadcom Trident II and Helix4 Limits (see page 139)
- Mellanox Spectrum Limits (see page 139)
- Supported Rule Types (see page 140)
 - iptables/ip6tables Rule Support (see page 141)
 - ebttables Rule Support (see page 141)
 - Other Unsupported Rules (see page 142)
- Common Examples (see page 142)
 - Policing Control Plane and Data Plane Traffic (see page 142)
 - Setting DSCP on Transit Traffic (see page 144)
 - Verifying DSCP Values on Transit Traffic (see page 144)
 - Checking the Packet and Byte Counters for ACL Rules (see page 144)
 - Filtering Specific TCP Flags (see page 147)
- Example Scenario (see page 147)
 - Switch 1 Configuration (see page 148)
 - Switch 2 Configuration (see page 149)
 - Egress Rule (see page 150)
 - Ingress Rule (see page 150)
 - Input Rule (see page 150)
 - Output Rule (see page 150)
 - Combined Rules (see page 150)
 - Layer 2-only Rules/ebtables (see page 151)
- Useful Links (see page 151)
- Caveats and Errata (see page 151)
 - Not All Rules Supported (see page 151)
 - Bridge Traffic Limitations (see page 151)
 - Log Actions Cannot Be Forwarded (see page 151)
 - Tomahawk Hardware Limitations (see page 151)
 - iptables Interactions with cl-acltool (see page 152)
 - Where to Assign Rules (see page 152)
 - Generic Error Message Displayed after ACL Rule Installation Failure (see page 153)
 - Dell S3048-ON Supports only 24K MAC Addresses (see page 153)

Understanding Traffic Rules In Cumulus Linux

Understanding Chains

Netfilter describes the mechanism for which packets are classified and controlled in the Linux kernel. Cumulus Linux uses the Netfilter framework to control the flow of traffic to, from and across the switch. Netfilter does not require a separate software daemon to run because it is part of the Linux kernel itself. Netfilter asserts policies at layers 2, 3 and 4 of the [OSI model](#) by inspecting packet and frame headers based on a list of rules. Rules are defined using syntax provided by the `iptables`, `ip6tables` and `ebtables` userspace applications.

The rules created by these programs inspect or operate on packets at several points in the life of the packet through the system. These five points are known as *chains* and are shown here:



The chains and their uses are:

- **PREROUTING:** Touches packets before they are routed
- **INPUT:** Touches packets once they are determined to be destined for the local system but before they are received by the control plane software
- **FORWARD:** Touches transit traffic as it moves through the box
- **OUTPUT:** Touches packets that are sourced by the control plane software before they are put on the wire
- **POSTROUTING:** Touches packets immediately before they are put on the wire but after the routing decision has been made



Understanding Tables

When building rules to affect the flow of traffic, the individual chains can be accessed by *tables*. Linux provides three tables by default:

- **Filter:** Classifies traffic or filters traffic
- **NAT:** Applies Network Address Translation rules

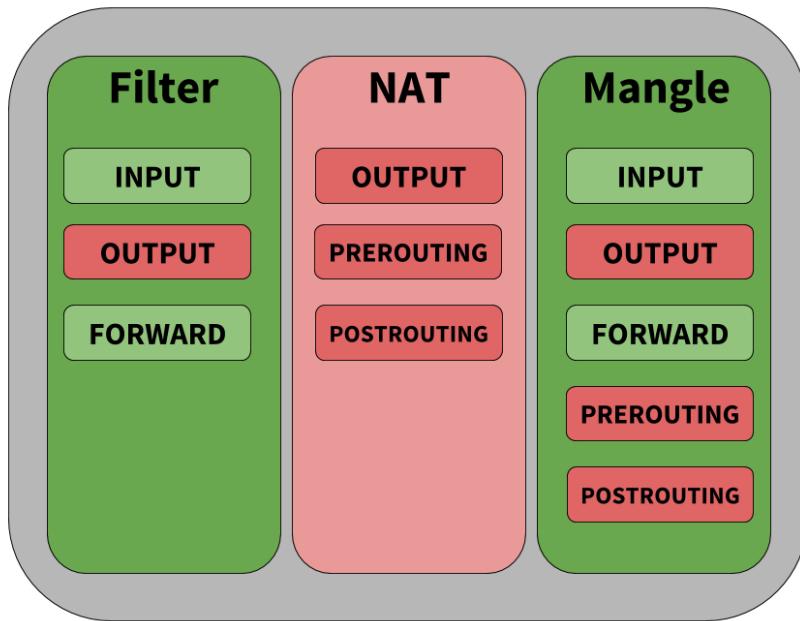


Cumulus Linux does not support NAT.

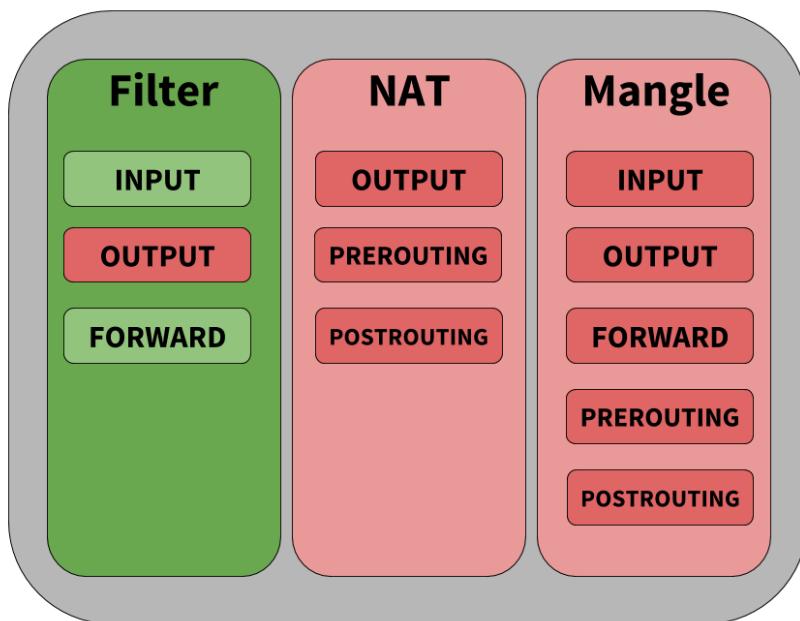
- **Mangle:** Alters packets as they move through the switch

Each table has a set of default chains that can be used to modify or inspect packets at different points of the path through the switch. Chains contain the individual rules to influence traffic. Each table and the default chains they support are shown below. Tables and chains in green are supported by Cumulus Linux, those in red are not supported (that is, they are not hardware accelerated) at this time.

IPtables/IP6tables Table Support



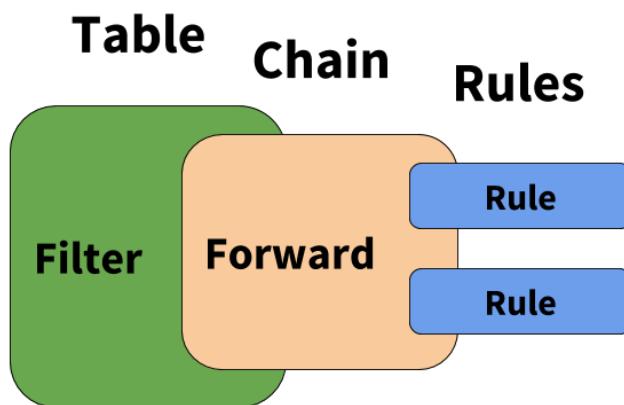
EBtables Table Support





Understanding Rules

Rules are the items that actually classify traffic to be acted upon. Rules are applied to chains, which are attached to tables, similar to the graphic below.



Rules have several different components; the examples below highlight those different components.

(Sets SSH as high priority traffic)

-t mangle	-A FORWARD	-p tcp --dport 22	-j	DSCP --set-dscp 46
-----------	------------	-------------------	----	--------------------

Table	Chain	Matches	Jump	Targets
	-A INPUT	-i swp1 -p tcp --dport bgp	-j	POLICE --set-mode pkt --set-rate 2000 --set-burst 2000 --set-class 7

(Police and Prioritize BGP Traffic)

- **Table:** The first argument is the *table*. Notice the second example does not specify a table, that is because the filter table is implied if a table is not specified.
- **Chain:** The second argument is the *chain*. Each table supports several different chains. See Understanding Tables above.

- **Matches:** The third argument(s) are called the *matches*. You can specify multiple matches in a single rule. However, the more matches you use in a rule, the more memory that rule consumes.
- **Jump:** The *jump* specifies the target of the rule; that is, what action to take if the packet matches the rule. If this option is omitted in a rule, then matching the rule will have no effect on the packet's fate, but the counters on the rule will be incremented.
- **Target(s):** The *target* can be a user-defined chain (other than the one this rule is in), one of the special built-in targets that decides the fate of the packet immediately (like `DROP`), or an extended target. See the [Supported Rule Types and Common Usages](#) (see page 140) section below for examples of different targets.

How Rules Are Parsed and Applied

All the rules from each chain are read from `iptables`, `ip6tables` and `ebtables` and entered in order into either the filter table or the mangle table. The rules are read from the kernel in the following order:

- IPv6 (`ip6tables`)
- IPv4 (`iptables`)
- `ebtables`

When rules are combined and put into one table, the order determines the relative priority of the rules; `iptables` and `ip6tables` have the highest precedence and `ebtables` has the lowest.

The Linux packet forwarding construct is an overlay for how the silicon underneath processes packets; to that end, here are some things to be aware of:

- The order of operations for how rules are processed is not perfectly maintained when you compare how `iptables` and the switch silicon process packets. The switch silicon reorders rules when `switchd` writes to the ASIC, whereas traditional `iptables` executes the list of rules in order.
- All rules are terminating. This means once a rule is matched, the action is carried out, and no more rules are processed. The exception to this is when a `SETCLASS` rule is placed immediately before another rule; this exists multiple times in the default ACL configuration.
In the example below, the `SETCLASS` action applied with the `--in-interface` option, creates the internal ASIC classification, and **continues to process** the next rule, which does the rate-limiting for the matched protocol:

```
-A $INGRESS_CHAIN --in-interface $INGRESS_INTF -p udp --dport
$BFD_ECHO_PORT -j SETCLASS --class 7
-A $INGRESS_CHAIN -p udp --dport $BFD_ECHO_PORT -j POLICE --set-
mode pkt --set-rate 2000 --set-burst 2000
```



If multiple contiguous rules with the same match criteria are applied to `--in-interface`, **only** the first rule gets processed and then terminates processing. This would also be a misconfiguration, because there is no reason to have duplicate rules with different actions.

- When processing traffic, rules affecting the FORWARD chain that specify an ingress interface are performed prior to rules that match on an egress interface. As a workaround, rules that only affect the egress interface can have an ingress interface wildcard (currently, only `swp+` and `bond+` are supported as wildcard names; see below) that matches any interface applied so that you can maintain order of operations with other input interface rules. Take the following rules, for example:

```
-A FORWARD -i $PORTA -j ACCEPT
-A FORWARD -o $PORTA -j ACCEPT    <-- This rule is performed LAST
(because of egress interface matching)
-A FORWARD -i $PORTB -j DROP
```

If you modify the rules like this, they are performed in order:

```
-A FORWARD -i $PORTA -j ACCEPT
-A FORWARD -i swp+ -o $PORTA -j ACCEPT    <-- These rules are
performed in order (because of wildcard match on ingress
interface)
-A FORWARD -i $PORTB -j DROP
```

- When using rules that do a mangle and a filter lookup for a packet, Cumulus Linux does them in parallel and combines the action.
- If a switch port is assigned to a bond, any egress rules must be assigned to the bond.
- When using the OUTPUT chain, rules must be assigned to the source. For example, if a rule is assigned to the switch port in the direction of traffic but the source is a bridge (VLAN), the traffic won't be affected by the rule and must be applied to the bridge.
- If all transit traffic needs to have a rule applied, use the FORWARD chain, not the OUTPUT chain.
- `eptables` rules are put into either the IPv4 or IPv6 memory space depending on whether the rule utilizes IPv4 or IPv6 to make a decision. Layer 2-only rules, which match the MAC address, are put into the IPv4 memory space.

Rule Placement in Memory

INPUT and ingress (FORWARD -i) rules occupy the same memory space. A rule counts as ingress if the -i option is set. If both input and output options (-i and -o) are set, the rule is considered as ingress and occupies that memory space. For example:

```
-A FORWARD -i swp1 -o swp2 -s 10.0.14.2 -d 10.0.15.8 -p tcp -j ACCEPT
```



If you set an output flag with the INPUT chain you will get an error. For example, running `cl-acltool -i` on the following rule:

```
-A FORWARD,INPUT -i swp1 -o swp2 -s 10.0.14.2 -d 10.0.15.8 -p
tcp -j ACCEPT
```



generates the following error:

```
error: line 2 : output interface specified with INPUT chain  
error processing rule '-A FORWARD,INPUT -i swp1 -o swp2 -s  
10.0.14.2 -d 10.0.15.8 -p tcp -j ACCEPT'
```

However, simply removing the `-o` option and interface would make it a valid rule.

Enabling Nonatomic Updates

You can enable nonatomic updates for `switchd`, which offer better scaling because all TCAM resources are used to actively impact traffic. With atomic updates, half of the hardware resources are on standby and do not actively impact traffic.

Nonatomic updates are now table based, so they don't interrupt network traffic when new rules are installed. The rules are mapped into the following tables and are updated in this order:

- mirror (ingress only)
- ipv4-mac (can be both ingress and egress)
- ipv6 (ingress only)

Updates are done incrementally, one table at a time without stopping traffic. Cumulus Linux checks if a table's rules have changed since the last time they were installed; if a table does not have any changes, then it is not reinstalled. If there are changes in a table, then the new rules are populated in new groups or slices in hardware, then that table is switched over to the new groups or slices. Finally, old resources for that table are freed. This process is repeated for each of the tables listed above. If a failure occurs during an update, the regular nonatomic mode is attempted, which interrupts network traffic. If that also fails, Cumulus Linux reverts back to the previous rules.

To always start `switchd` with nonatomic updates:

1. Edit `/etc/cumulus/switchd.conf`.
2. Add the following line to the file:

```
acl.non_atomic_update_mode = TRUE
```

3. Restart `switchd`:

```
cumulus@switch:~$ sudo systemctl restart switchd.service
```



During nonatomic updates, traffic is stopped first, and enabled after the new configuration is written into the hardware completely.



Using iptables/ip6tables/ebtables Directly

Using `iptables`/`ip6tables`/`ebtables` directly is not recommended because any rules installed in these cases only are applied to the Linux kernel and are not hardware accelerated via synchronization to the switch silicon. Also running `cl-acltool -i` (the installation command) resets all rules and deletes anything that is not stored in `/etc/cumulus/acl/policy.conf`.

For example, performing:

```
cumulus@switch:~$ sudo iptables -A INPUT -p icmp --icmp-type echo-request -j DROP
```

Appears to work, and the rule appears when you run `cl-acltool -L`:

```
cumulus@switch:~$ sudo cl-acltool -L ip
-----
Listing rules of type iptables:
-----

TABLE filter :
Chain INPUT (policy ACCEPT 72 packets, 5236 bytes)
 pkts bytes target prot opt in out source destination
 0 0 DROP  icmp -- any any anywhere anywhere icmp echo-request
```

However, the rule is not synced to hardware when applied in this fashion and running `cl-acltool -i` or `reboot` removes the rule without replacing it. To ensure all rules that can be in hardware are hardware accelerated, place them in `/etc/cumulus/acl/policy.conf` and install them by running `cl-acltool -i`.

Estimating the Number of Rules

To estimate the number of rules that could be created from an ACL entry, first determine if that entry is an ingress or an egress. Then determine if it is IPv4-mac or IPv6 type rule. This determines the slice to which the rule belongs. Then use the following to determine how many entries are used up for each type.

By default, each entry occupies one double wide entry, except if the entry is one of the following:

- An entry with multiple comma-separated input interfaces is split into one rule for each input interface (listed after `--in-interface` below). For example, this entry splits into two rules:

```
-A FORWARD --in-interface swp1s0,swp1s1 -p icmp -j ACCEPT
```

- An entry with multiple comma-separated output interfaces is split into one rule for each output interface (listed after `--out-interface` below). this entry splits into two rules:

```
-A FORWARD --in-interface swp+ --out-interface swp1s0,swp1s1 -p icmp -j ACCEPT
```



- An entry with both input and output comma-separated interfaces is split into one rule for each combination of input and output interface (listed after `--in-interface` and `--out-interface` below). For example, this entry splits into four rules:

```
-A FORWARD --in-interface swp1s0,swp1s1 --out-interface swp1s2,  
swp1s3 -p icmp -j ACCEPT
```

- An entry with multiple L4 port ranges is split into one rule for each range (listed after `--dports` below). For example, this entry splits into two rules:

```
-A FORWARD --in-interface swp+ -p tcp -m multiport --dports 1050:  
1051,1055:1056 -j ACCEPT
```

Installing and Managing ACL Rules with NCLU

NCLU provides an easy way to create custom ACLs in Cumulus Linux. The rules you create live in the `/var/lib/cumulus/nclu/nclu_acl.conf` file, which gets converted to a rules file, `/etc/cumulus/acl/policy.d/50_nclu_acl.rules`. This way, the rules you create with NCLU are independent of the two default files in `/etc/cumulus/acl/policy.d/00control_plane.rules` and `99control_plane_catch_all.rules`, as the content in these files may get updated after you upgrade Cumulus Linux.

Instead of crafting a rule by hand then installing it using `cl-acltool`, NCLU handles many of the options automatically. For example, consider the following `iptables` rule:

```
-A FORWARD -i swp1 -o swp2 -s 10.0.14.2 -d 10.0.15.8 -p tcp -j ACCEPT
```

You would create this rule, called `EXAMPLE1`, using NCLU like this:

```
cumulus@switch:~$ net add acl ipv4 EXAMPLE1 accept tcp source-ip  
10.0.14.2/32 source-port dest-ip 10.0.15.8/32 dest-port any  
cumulus@switch:~$ net pending  
cumulus@switch:~$ net commit
```

All options, such as the `-j` and `-p`, even `FORWARD` in the above rule, are added automatically when you apply the rule to the control plane; NCLU figures it all out for you.

You can also set a priority value, which specifies the order in which the rules get executed, and the order in which they appear in the rules file. Lower numbers are executed first. To add a new rule in the middle, first run `net show config acl`, which displays the priority numbers. Otherwise, new rules get appended to the end of the list of rules in the `nclu_acl.conf` and `50_nclu_acl.rules` files.



If you need to hand edit a rule, don't edit the `50_nclu_acl.rules` file. Instead, edit the `nclu_acl.conf` file.



After you add the rule, you need to apply it to an inbound or outbound interface using `net add int acl`; `swp1` is the inbound interface in our example:

```
cumulus@switch:~$ net add int swp1 acl ipv4 EXAMPLE1 inbound
cumulus@switch:~$ net pending
cumulus@switch:~$ net commit
```

After you commit your changes, you can verify the rule you created with NCLU by running `net show configuration acl`:

```
cumulus@switch:~$ net show configuration acl
acl ipv4 EXAMPLEv4 priority 10 accept tcp source-ip 10.0.14.2/32
source-port any dest-ip 10.0.15.8/32 dest-port any

interface swp1
  acl ipv4 EXAMPLE1 inbound
```

Or you can see all of the rules installed by running `cat` on the `50_nclu_acl.rules` file:

```
cumulus@switch:~$ cat /etc/cumulus/acl/policy.d/50_nclu_acl.rules
[iptables]
# swp1: acl ipv4 EXAMPLE1 inbound
-A FORWARD --in-interface swp1 --out-interface swp2 -j ACCEPT -p tcp -
s 10.0.14.2/32 -d 10.0.15.8/32 --dport 110
```

For INPUT and FORWARD rules, apply the rule to a control plane interface using `net add control-plane`:

```
cumulus@switch:~$ net add control-plane acl ipv4 EXAMPLE1 inbound
cumulus@switch:~$ net pending
cumulus@switch:~$ net commit
```

To remove a rule, use `net del acl ipv4|ipv6|mac RULENAME`:

```
cumulus@switch:~$ net del acl ipv4 EXAMPLE1
cumulus@switch:~$ net pending
cumulus@switch:~$ net commit
```

This deletes all rules from the `50_nclu_acl.rules` file with that name. It also deletes the interfaces referenced in the `nclu_acl.conf` file.



Installing and Managing ACL Rules with cl-acltool

You can manage Cumulus Linux ACLs with `cl-acltool`. Rules are first written to the `iptables` chains, as described above, and then synced to hardware via `switchd`.



Use `iptables`/`ip6tables`/`ebtables` and `cl-acltool` to manage rules in the default files, `00control_plane.rules` and `99control_plane_catch_all.rules`; they're not aware of rules created using NCLU.

To examine the current state of chains and list all installed rules, run:

```
cumulus@switch:~$ sudo cl-acltool -L all
-----
Listing rules of type iptables:
-----
TABLE filter :
Chain INPUT (policy ACCEPT 90 packets, 14456 bytes)
pkts bytes target prot opt in out source destination
0 0 DROP all -- swp+ any 240.0.0.0/5 anywhere
0 0 DROP all -- swp+ any loopback/8 anywhere
0 0 DROP all -- swp+ any base-address.mcast.net/8 anywhere
0 0 DROP all -- swp+ any 255.255.255.255 anywhere ...
```

To list installed rules using native `iptables`, `ip6tables` and `ebtables`, use the `-L` option with the respective commands:

```
cumulus@switch:~$ sudo iptables -L
cumulus@switch:~$ sudo ip6tables -L
cumulus@switch:~$ sudo ebtables -L
```

To flush all installed rules, run:

```
cumulus@switch:~$ sudo cl-acltool -F all
```

To flush only the IPv4 `iptables` rules, run:

```
cumulus@switch:~$ sudo cl-acltool -F ip
```

If the install fails, ACL rules in the kernel and hardware are rolled back to the previous state. Errors from programming rules in the kernel or ASIC are reported appropriately.

Installing Packet Filtering (ACL) Rules

`cl-acltool` takes access control list (ACL) rules input in files. Each ACL policy file contains `iptables`, `ip6tables` and `ebtables` categories under the tags `[iptables]`, `[ip6tables]` and `[ebtables]` respectively.

Each rule in an ACL policy must be assigned to one of the rule categories above.

See `man cl-acltool(5)` for ACL rule details. For `iptables` rule syntax, see `man iptables(8)`. For `ip6tables` rule syntax, see `man ip6tables(8)`. For `ebtables` rule syntax, see `man ebttables(8)`.

See `man cl-acltool(5)` and `man cl-acltool(8)` for further details on using `cl-acltool`; however, some examples are listed here, and more are listed [later in this chapter \(see page\)](#).



By default:

- ACL policy files are located in `/etc/cumulus/acl/policy.d/`.
- All `*.rules` files in this directory are included in `/etc/cumulus/acl/policy.conf`.
- All files included in this `policy.conf` file are installed when the switch boots up.
- The `policy.conf` file expects rules files to have a `.rules` suffix as part of the file name.

Here is an example ACL policy file:

```
[iptables]
-A INPUT --in-interface swp1 -p tcp --dport 80 -j ACCEPT
-A FORWARD --in-interface swp1 -p tcp --dport 80 -j ACCEPT

[ip6tables]
-A INPUT --in-interface swp1 -p tcp --dport 80 -j ACCEPT
-A FORWARD --in-interface swp1 -p tcp --dport 80 -j ACCEPT

[ebtables]
-A INPUT -p IPv4 -j ACCEPT
-A FORWARD -p IPv4 -j ACCEPT
```

You can use wildcards or variables to specify chain and interface lists to ease administration of rules.



Interface Wildcards

Currently only `swp+` and `bond+` are supported as wildcard names. There may be kernel restrictions in supporting more complex wildcards like `swp1+` etc.

```
INGRESS = swp+
INPUT_PORT_CHAIN = INPUT, FORWARD
```



```
[iptables]
-A $INPUT_CHAIN --in-interface $INGRESS -p tcp --dport 80 -j ACCEPT

[ip6tables]
-A $INPUT_CHAIN --in-interface $INGRESS -p tcp --dport 80 -j ACCEPT

[ebtables]
-A INPUT -p IPv4 -j ACCEPT
```

ACL rules for the system can be written into multiple files under the default /etc/cumulus/acl/policy.d/ directory. The ordering of rules during installation follows the sort order of the files based on their file names.

Use multiple files to stack rules. The example below shows two rules files separating rules for management and datapath traffic:

```
cumulus@switch:~$ ls /etc/cumulus/acl/policy.d/
00sample_mgmt.rules 01sample_datapath.rules
cumulus@switch:~$ cat /etc/cumulus/acl/policy.d/00sample_mgmt.rules

INGRESS_INTF = swp+
INGRESS_CHAIN = INPUT

[iptables]
# protect the switch management
-A $INGRESS_CHAIN --in-interface $INGRESS_INTF -s 10.0.14.2 -d
10.0.15.8 -p tcp -j ACCEPT
-A $INGRESS_CHAIN --in-interface $INGRESS_INTF -s 10.0.11.2 -d
10.0.12.8 -p tcp -j ACCEPT
-A $INGRESS_CHAIN --in-interface $INGRESS_INTF -d 10.0.16.8 -p udp -j
DROP

cumulus@switch:~$ cat /etc/cumulus/acl/policy.d/01sample_datapath.rules
INGRESS_INTF = swp+
INGRESS_CHAIN = INPUT, FORWARD

[iptables]
-A $INGRESS_CHAIN --in-interface $INGRESS_INTF -s 192.0.2.5 -p icmp -j ACCEPT
-A $INGRESS_CHAIN --in-interface $INGRESS_INTF -s 192.0.2.6 -d
192.0.2.4 -j DROP
-A $INGRESS_CHAIN --in-interface $INGRESS_INTF -s 192.0.2.2 -d
192.0.2.8 -j DROP
```



Install all ACL policies under a directory:

```
cumulus@switch:~$ sudo cl-acltool -i -P ./rules
Reading files under rules
Reading rule file ./rules/01_http_rules.txt ...
Processing rules in file ./rules/01_http_rules.txt ...
Installing acl policy ...
Done.
```

Install all rules and policies included in /etc/cumulus/acl/policy.conf:

```
cumulus@switch:~$ sudo cl-acltool -i
```

Specifying which Policy Files to Install

By default, any .rules file you configure in /etc/cumulus/acl/policy.d/ get installed by Cumulus Linux. To add other policy files to an ACL, you need to include them in /etc/cumulus/acl/policy.conf. For example, in order for Cumulus Linux to install a rule in a policy file called 01_new.datapathacl, you would add `include /etc/cumulus/acl/policy.d/01_new.rules` to `policy.conf`, as in this example:

```
cumulus@switch:~$ sudo nano /etc/cumulus/acl/policy.conf

#
# This file is a master file for acl policy file inclusion
#
# Note: This is not a file where you list acl rules.
#
# This file can contain:
# - include lines with acl policy files
#   example:
#     include <filepath>
#
# see manpage cl-acltool(5) and cl-acltool(8) for how to write policy
files
#
include /etc/cumulus/acl/policy.d/01_new.datapathacl
```

Hardware Limitations on Number of Rules

The maximum number of rules that can be handled in hardware is a function of the platform type (switch silicon, like Tomahawk) and a mix of IPv4 and/or IPv6. See the [HCL](#) to determine which platform type applies to a particular switch.



If the maximum number of rules for a particular table is exceeded, `cl-acltool -i` generates the following error:

```
error: hw sync failed (sync_acl hardware installation failed) Rolling
back .. failed.
```

The nonatomic mode rules are used when nonatomic updates are enabled ([see above \(see page 131\)](#)).

Broadcom Tomahawk Limits

Direction	Atomic Mode IPv4 Rules	Atomic Mode IPv6 Rules	Nonatomic Mode IPv4 Rules	Nonatomic Mode IPv6 Rules
Ingress	512	512	1024	1024
Egress	512	256	1024	512

Broadcom Trident II+ Limits

Direction	Atomic Mode IPv4 Rules	Atomic Mode IPv6 Rules	Nonatomic Mode IPv4 Rules	Nonatomic Mode IPv6 Rules
Ingress	4096	4096	8192	8192
Egress	512	256	1024	512

Broadcom Trident II and Helix4 Limits

Direction	Atomic Mode IPv4 Rules	Atomic Mode IPv6 Rules	Nonatomic Mode IPv4 Rules	Nonatomic Mode IPv6 Rules
Ingress	1024	1024	2048	2048
Egress	512	256	1024	512

Mellanox Spectrum Limits

The Mellanox Spectrum ASIC has one common [TCAM](#) for both ingress and egress, and it may be used for other non-ACL-related resources. However, the number of supported rules varies with the [TCAM profile \(see page 481\)](#) specified for the switch.

Profile	Atomic Mode IPv4 Rules	Atomic Mode IPv6 Rules	Nonatomic Mode IPv4 Rules	Nonatomic Mode IPv6 Rules
default	1750	750	3500	1500
ipmc-heavy	200	80	400	160
acl-heavy	3000	2000	6000	2500

Supported Rule Types

The `iptables`/`ip6tables`/`ebtables` construct tries to layer the Linux implementation on top of the underlying hardware but they are not always directly compatible. Here are the supported rules for chains in `iptables`, `ip6tables` and `ebtables`.



To learn more about any of the options shown in the tables below, run `iptables -h [name of option]`. The same help syntax works for options for `ip6tables` and `ebtables`.

[Click to see an example of help syntax for an ebttables target](#)

```
root@leaf1# ebttables -h tricolorpolice
<...snip...
tricolorpolice option:
  --set-color-mode STRING setting the mode in blind or aware
  --set-cir INT setting committed information rate in kbits per
second
  --set-cbs INT setting committed burst size in kbyte
  --set-pir INT setting peak information rate in kbits per
second
  --set-ebs INT setting excess burst size in kbyte
  --set-conform-action-dscp INT setting dscp value if the
action is accept for conforming packets
  --set-exceed-action-dscp INT setting dscp value if the action
is accept for exceeding packets
  --set-violate-action STRING setting the action (accept/drop)
for violating packets
  --set-violate-action-dscp INT setting dscp value if the
action is accept for violating packets
Supported chains for the filter table:
INPUT FORWARD OUTPUT
```



iptables/ip6tables Rule Support

Rule Element	Supported	Unsupported
Matches	<ul style="list-style-type: none">Src/Dst, IP protocolIn/out interfaceIPv4: icmp, ttl,IPv6: icmp6, frag, hl,IP common: tcp (with flags (see page 147)), udp, multiport, TOS, DSCP, addrtype	<ul style="list-style-type: none">Rules with input/output Ethernet interfaces are ignoredInverse matches
Standard Targets	<ul style="list-style-type: none">ACCEPT, DROP	<ul style="list-style-type: none">RETURN, QUEUE, STOP, Fall Thru, Jump
Extended Targets	<ul style="list-style-type: none">LOG (IPv4/IPv6); UID is not supported for LOGTCP SEQ, TCP options or IP optionsULOGSETQOSDSCP <p><i>Unique to Cumulus Linux:</i></p> <ul style="list-style-type: none">SPANERSPAN (IPv4/IPv6)POLICETRICOLORPOLICESETCLASS	

ebtables Rule Support

Rule Element	Supported	Unsupported
Matches	<ul style="list-style-type: none">ether typeinput interface/wildcardoutput interface/wildcardsrc/dst MACIP: src, dest, tos, proto, sport, dportIPv6: tcclass, icmp6: type, icmp6: code range, src /dst addr, sport, dport	<ul style="list-style-type: none">Inverse matchesProto lengthVLAN802.1p (CoS)



Standard Targets	<ul style="list-style-type: none">ACCEPT, DROP	<ul style="list-style-type: none">Return, Continue, Jump, Fall Thru
Extended Targets	<ul style="list-style-type: none">Uloglog <p><i>Unique to Cumulus Linux:</i></p> <ul style="list-style-type: none">spanerspanpolicetricolorpolicesetclass	

Other Unsupported Rules

- Rules that have no matches and accept all packets in a chain are currently ignored. This probably has side effects in the sense that the rules below them do get hit, when normally they wouldn't.
- Chain default rules (which are ACCEPT) are also ignored.

Common Examples

Policing Control Plane and Data Plane Traffic

You can configure quality of service for traffic on both the control plane and the data plane. By using QoS policers, you can rate limit traffic so incoming packets get dropped if they exceed specified thresholds.



Counters on POLICE ACL rules in `iptables` do not currently show the packets that are dropped due to those rules.

Use the `POLICE` target with `iptables`. `POLICE` takes these arguments:

- `--set-class value`: Sets the system internal class of service queue configuration to *value*.
- `--set-rate value`: Specifies the maximum rate in kilobytes (KB) or packets.
- `--set-burst value`: Specifies the number of packets or kilobytes (KB) allowed to arrive sequentially.
- `--set-mode string`: Sets the mode in KB (kilobytes) or pkt (packets) for rate and burst size.

For example, to rate limit the incoming traffic on `swp1` to 400 packets/second with a burst of 100 packets /second and set the class of the queue for the policed traffic as 0, set this rule in your appropriate `.rules` file:



```
-A INPUT --in-interface swp1 -j POLICE --set-mode pkt --set-rate 400  
--set-burst 100 --set-class 0
```

Here is another example of control plane ACL rules to lock down the switch. You specify them in /etc/cumulus/acl/policy.d/00control_plane.rules:

View the contents of the file ...

```
INGRESS_INTF = swp+
INGRESS_CHAIN = INPUT
INNFWD_CHAIN = INPUT,FORWARD
MARTIAN_SOURCES_4 = "240.0.0.0/5,127.0.0.0/8,224.0.0.0/8,
255.255.255.255/32"
MARTIAN_SOURCES_6 = "ff00::/8,::128,::ffff:0.0.0.0/96,::1/128"
# Custom Policy Section
SSH_SOURCES_4 = "192.168.0.0/24"
NTP_SERVERS_4 = "192.168.0.1/32,192.168.0.4/32"
DNS_SERVERS_4 = "192.168.0.1/32,192.168.0.4/32"
SNMP_SERVERS_4 = "192.168.0.1/32"
[iptables]
-A $INNFWD_CHAIN --in-interface $INGRESS_INTF -s $MARTIAN_SOURCES_4 -
j DROP
-A $INGRESS_CHAIN --in-interface $INGRESS_INTF -p ospf -j POLICE --
set-mode pkt --set-rate 2000 --set-burst 2000 --set-class 7
-A $INGRESS_CHAIN --in-interface $INGRESS_INTF -p tcp --dport bgp -j
POLICE --set-mode pkt --set-rate 2000 --set-burst 2000 --set-class 7
-A $INGRESS_CHAIN --in-interface $INGRESS_INTF -p tcp --sport bgp -j
POLICE --set-mode pkt --set-rate 2000 --set-burst 2000 --set-class 7
-A $INGRESS_CHAIN --in-interface $INGRESS_INTF -p icmp -j POLICE --
set-mode pkt --set-rate 100 --set-burst 40 --set-class 2
-A $INGRESS_CHAIN --in-interface $INGRESS_INTF -p udp --dport bootps:
bootpc -j POLICE --set-mode pkt --set-rate 100 --set-burst 100 --set-
class 2
-A $INGRESS_CHAIN --in-interface $INGRESS_INTF -p tcp --dport bootps:
bootpc -j POLICE --set-mode pkt --set-rate 100 --set-burst 100 --set-
class 2
-A $INGRESS_CHAIN --in-interface $INGRESS_INTF -p igmp -j POLICE --
set-mode pkt --set-rate 300 --set-burst 100 --set-class 6
# Custom policy
-A $INGRESS_CHAIN --in-interface $INGRESS_INTF -p tcp --dport 22 -s
$SSH_SOURCES_4 -j ACCEPT
-A $INGRESS_CHAIN --in-interface $INGRESS_INTF -p udp --sport 123 -s
$NTP_SERVERS_4 -j ACCEPT
-A $INGRESS_CHAIN --in-interface $INGRESS_INTF -p udp --sport 53 -s
$DNS_SERVERS_4 -j ACCEPT
-A $INGRESS_CHAIN --in-interface $INGRESS_INTF -p udp --dport 161 -s
$SNMP_SERVERS_4 -j ACCEPT
# Allow UDP traceroute when we are the current TTL expired hop
-A $INGRESS_CHAIN --in-interface $INGRESS_INTF -p udp --dport 1024:
65535 -m ttl --ttl-eq 1 -j ACCEPT
-A $INGRESS_CHAIN --in-interface $INGRESS_INTF -j DROP
```

Setting DSCP on Transit Traffic

The examples here use the *mangle* table to modify the packet as it transits the switch. DSCP is expressed in **decimal notation** in the examples below.

```
[iptables]

#Set SSH as high priority traffic.
-t mangle -A FORWARD -p tcp --dport 22 -j DSCP --set-dscp 46

#Set everything coming in SWP1 as AF13
-t mangle -A FORWARD --in-interface swp1 -j DSCP --set-dscp 14

#Set Packets destined for 10.0.100.27 as best effort
-t mangle -A FORWARD -d 10.0.100.27/32 -j DSCP --set-dscp 0

#Example using a range of ports for TCP traffic
-t mangle -A FORWARD -p tcp -s 10.0.0.17/32 --sport 10000:20000 -d
10.0.100.27/32 --dport 10000:20000 -j DSCP --set-dscp 34
```

Verifying DSCP Values on Transit Traffic

The examples here use the DSCP match criteria in combination with other IP, TCP and interface matches to identify traffic and count the number of packets.

```
[iptables]

#Match and count the packets that match SSH traffic with DSCP EF
-A FORWARD -p tcp --dport 22 -m dscp --dscp 46 -j ACCEPT

#Match and count the packets coming in SWP1 as AF13
-A FORWARD --in-interface swp1 -m dscp --dscp 14 -j ACCEPT
#Match and count the packets with a destination 10.0.0.17 marked best
effort
-A FORWARD -d 10.0.100.27/32 -m dscp --dscp 0 -j ACCEPT

#Match and count the packets in a port range with DSCP AF41
-A FORWARD -p tcp -s 10.0.0.17/32 --sport 10000:20000 -d 10.0.100.27
/32 --dport 10000:20000 -m dscp --dscp 34 -j ACCEPT
```

Checking the Packet and Byte Counters for ACL Rules

To verify the counters, using the above example rules, first send test traffic matching the patterns through the network. The following example generates traffic with *mz*, which can be installed on host servers or even on Cumulus Linux switches. Once traffic is sent to validate the counters, they are matched on switch1 use *cl-acltool*.



```
# Send 100 TCP packets on host1 with a DSCP value of EF with a
# destination of host2 TCP port 22:

cumulus@host1$ mz eth1 -A 10.0.0.17 -B 10.0.100.27 -c 100 -v -t tcp
"dp=22,dscp=46"
  IP:  ver=4, len=40, tos=184, id=0, frag=0, ttl=255, proto=6, sum=0,
  SA=10.0.0.17, DA=10.0.100.27,
    payload=[see next layer]
  TCP:  sp=0, dp=22, S=42, A=42, flags=0, win=10000, len=20, sum=0,
    payload=

# Verify the 100 packets are matched on switch1

cumulus@switch1$ sudo cl-acltool -L ip
-----
Listing rules of type iptables:
-----
TABLE filter :
Chain INPUT (policy ACCEPT 9314 packets, 753K bytes)
  pkts bytes target      prot opt in     out      source
destination
Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
  pkts bytes target      prot opt in     out      source
destination
    100   6400 ACCEPT      tcp   --  any      any      anywhere
anywhere          tcp  dpt:ssh DSCP match 0x2e
      0     0 ACCEPT      all   --  swp1    any      anywhere
anywhere          DSCP match 0x0e
      0     0 ACCEPT      all   --  any      any      10.0.0.17
anywhere          DSCP match 0x00
      0     0 ACCEPT      tcp   --  any      any      10.0.0.17
10.0.100.27        tcp  spts:webmin:20000 dpts:webmin:2002
```

```
# Send 100 packets with a small payload on host1 with a DSCP value of
# AF13 with a destination of host2:

cumulus@host1$ mz eth1 -A 10.0.0.17 -B 10.0.100.27 -c 100 -v -t ip
  IP:  ver=4, len=20, tos=0, id=0, frag=0, ttl=255, proto=0, sum=0,
  SA=10.0.0.17, DA=10.0.100.27,
    payload=

# Verify the 100 packets are matched on switch1

cumulus@switch1$ sudo cl-acltool -L ip
-----
Listing rules of type iptables:
-----
TABLE filter :
```



```
Chain INPUT (policy ACCEPT 9314 packets, 753K bytes)
  pkts bytes target     prot opt in      out      source
destination
Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
  pkts bytes target     prot opt in      out      source
destination
  100  6400 ACCEPT     tcp   --  any    any    anywhere
anywhere          tcp dpt:ssh DSCP match 0x2e
  100  7000 ACCEPT     all   --  swp3   any    anywhere
anywhere          DSCP match 0x0e
  100  6400 ACCEPT     all   --  any    any    10.0.0.17
anywhere          DSCP match 0x00
      0    0 ACCEPT     tcp   --  any    any    10.0.0.17
10.0.100.27        tcp spts:webmin:20000 dpts:webmin:2002
```

```
# Send 100 packets on host1 with a destination of host2:
```

```
cumulus@host1$ mz eth1 -A 10.0.0.17 -B 10.0.100.27 -c 100 -v -t ip
  IP: ver=4, len=20, tos=56, id=0, frag=0, ttl=255, proto=0, sum=0,
SA=10.0.0.17, DA=10.0.100.27,
  payload=
```

```
# Verify the 100 packets are matched on switch1
```

```
cumulus@switch1$ sudo cl-acltool -L ip
-----
Listing rules of type iptables:
-----
TABLE filter :
Chain INPUT (policy ACCEPT 9314 packets, 753K bytes)
  pkts bytes target     prot opt in      out      source
destination
Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
  pkts bytes target     prot opt in      out      source
destination
  100  6400 ACCEPT     tcp   --  any    any    anywhere
anywhere          tcp dpt:ssh DSCP match 0x2e
  100  7000 ACCEPT     all   --  swp3   any    anywhere
anywhere          DSCP match 0x0e
      0    0 ACCEPT     all   --  any    any    10.0.0.17
anywhere          DSCP match 0x00
      0    0 ACCEPT     tcp   --  any    any    10.0.0.17
10.0.100.27        tcp spts:webmin:20000 dpts:webmin:2002Still
working
```



Filtering Specific TCP Flags

The example solution below creates rules on the INPUT and FORWARD chains to drop ingress IPv4 and IPv6 TCP packets when the SYN bit is set and the RST, ACK and FIN bits are reset. The default for the INPUT and FORWARD chains allows all other packets. The ACL is applied to ports swp20 and swp21. After configuring this ACL, new TCP sessions that originate from ingress ports swp20 and swp21 will not be allowed. TCP sessions that originate from any other port are allowed.

```
INGRESS_INTF = swp20,swp21

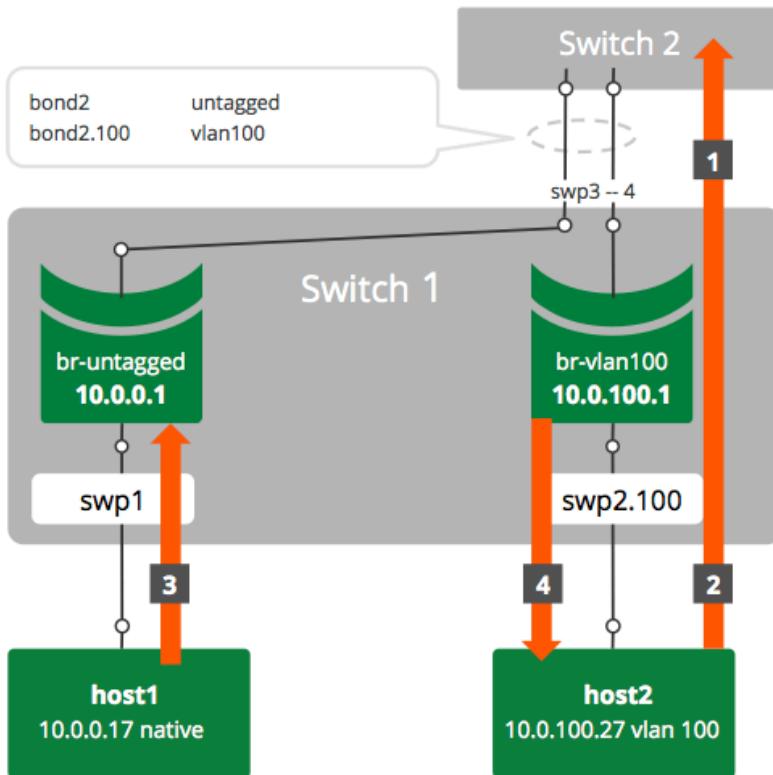
[iptables]
-A INPUT, FORWARD --in-interface $INGRESS_INTF -p tcp --syn -j DROP
[ip6tables]
-A INPUT, FORWARD --in-interface $INGRESS_INTF -p tcp --syn -j DROP
```

The `--syn` flag in the above rule matches packets with the SYN bit set and the ACK, RST and FIN bits are cleared. It is equivalent to using `-tcp-flags SYN,RST,ACK,FIN SYN`. For example, the above rule could be re-written as:

```
-A INPUT, FORWARD --in-interface $INGRESS_INTF -p tcp --tcp-flags SYN,
RST,ACK,FIN SYN -j DROP
```

Example Scenario

The following example scenario demonstrates where several different rules are applied to show what is possible.



Hardware Accelerated

- 1 egress rule**
(Forward Chain with `-o` flag)
- 2 ingress rule**
(Forward Chain with `-i` flag)
- 3 input rule**
(Input Chain with or w/o `-i` flag)

In Software

- 4 output rule**
(Output Chain with or w/o `-o` flag)

Following are the configurations for the two switches used in these examples. The configuration for each switch appears in `/etc/network/interfaces` on that switch.

Switch 1 Configuration

```
cumulus@switch1:~$ net show configuration files
...
/etc/network/interfaces
=====

auto swp1
iface swp1

auto swp2
iface swp2

auto swp3
iface swp3

auto swp4
iface swp4

auto bond2
```



```
iface bond2
    bond-slaves swp3 swp4

auto br-untagged
iface br-untagged
    address 10.0.0.1/24
    bridge_ports swp1 bond2
    bridge_stp on

auto br-tag100
iface br-tag100
    address 10.0.100.1/24
    bridge_ports swp2.100 bond2.100
    bridge_stp on

...
```

Switch 2 Configuration

```
cumulus@switch2:~$ net show configuration files

...
/etc/network/interfaces
=====

auto swp3
iface swp3

auto swp4
iface swp4

auto br-untagged
iface br-untagged
    address 10.0.0.2/24
    bridge_ports bond2
    bridge_stp on

auto br-tag100
iface br-tag100
    address 10.0.100.2/24
    bridge_ports bond2.100
    bridge_stp on

auto bond2
iface bond2
    bond-slaves swp3 swp4

...
```



Egress Rule

The following rule blocks any TCP with destination port 200 traffic going from host1 or host2 through the switch (corresponding to rule 1 in the diagram above).

```
[iptables] -A FORWARD -o bond2 -p tcp --dport 200 -j DROP
```

Ingress Rule

The following rule blocks any UDP traffic with source port 200 going from host1 through the switch (corresponding to rule 2 in the diagram above).

```
[iptables] -A FORWARD -i swp2 -p udp --sport 200 -j DROP
```

Input Rule

The following rule blocks any UDP traffic with source port 200 and destination port 50 going from host1 to the switch (corresponding to rule 3 in the diagram above).

```
[iptables] -A INPUT -i swp1 -p udp --sport 200 --dport 50 -j DROP
```

Output Rule

The following rule blocks any TCP traffic with source port 123 and destination port 123 going from Switch 1 to host2 (corresponding to rule 4 in the diagram above).

```
[iptables] -A OUTPUT -o br-tag100 -p tcp --sport 123 --dport 123 -j DROP
```

Combined Rules

The following rule blocks any TCP traffic with source port 123 and destination port 123 going from any switch port egress or generated from Switch 1 to host1 or host2 (corresponding to rules 1 and 4 in the diagram above).

```
[iptables] -A OUTPUT, FORWARD -o swp+ -p tcp --sport 123 --dport 123 -j DROP
```

This also becomes 2 ACLs, and is effectively the same as:



```
[iptables]
-A FORWARD -o swp+ -p tcp --sport 123 --dport 123 -j DROP
-A OUTPUT -o swp+ -p tcp --sport 123 --dport 123 -j DROP
```

Layer 2-only Rules/ebtables

The following rule blocks any traffic with source MAC address 00:00:00:00:00:12 and destination MAC address 08:9e:01:ce:e2:04 going from any switch port egress/ingress.

```
[ebtables] -A FORWARD -s 00:00:00:00:00:12 -d 08:9e:01:ce:e2:04 -j
DROP
```

Useful Links

- www.netfilter.org
- Netfilter.org packet filtering how-to

Caveats and Errata

Not All Rules Supported

As mentioned in the [Supported Rules section](#) (see page 140) above, not all `iptables`, `ip6tables` or `ebtables` rules are supported. Refer to that section for specific rule support.

Bridge Traffic Limitations

Bridge traffic that matches LOG ACTION rules are not logged in syslog, as the kernel and hardware identify packets using different information.

Log Actions Cannot Be Forwarded

Logged packets cannot be forwarded. The hardware cannot both forward a packet and send the packet to the control plane (or kernel) for logging. To emphasize this, a log action must also have a drop action.

Tomahawk Hardware Limitations

Rate Limiting per Pipeline, Not Global

On Tomahawk switches, the field processor (FP) polices on a per-pipeline basis instead of globally, as with a Trident II switch. If packets come in to different switch ports that are on different pipelines on the ASIC, they may be rate limited differently.

For example, your switch is set so BFD is rate limited to 2000 packets per second. When the BFD packets are received on port1/pipe1 and port2/pipe2, they would each be rate limited at 2000 pps, thus the switch is rate limiting at 4000 pps overall. Since there are four pipelines on a Tomahawk switch, you could see a 4x increase of your configured rate limits.



Ping-pong Mode Enabled by Default

In Cumulus Linux, the ping-pong atomic policy update mode is enabled by default. If you have Tomahawk switches and plan to use SPAN and/or mangle rules, you must disable ping-pong mode.

To do so, set the value for `acl.non_atomic_update_mode` to true in `/etc/cumulus/switchd.conf`:

```
acl.non_atomic_update_mode = TRUE
```

Then `restart switchd`.

Packets Undercounted during ACL Updates

On Tomahawk switches, when updating egress FP rules, some packets do not get counted. This results in an underreporting of counts during ping-pong/incremental switchover.

iptables Interactions with *cl-acltool*

Since Cumulus Linux is a Linux operating system, the `iptables` commands can be used directly and does work. However, you should consider using `cl-acltool` instead because:

- Without using `cl-acltool`, rules are not installed into hardware.
- Running `cl-acltool -i` (the installation command) resets all rules and deletes anything that is not stored in `/etc/cumulus/acl/policy.conf`.

For example, running the following command works:

```
cumulus@switch:~$ sudo iptables -A INPUT -p icmp --icmp-type echo-request -j DROP
```

And the rules appear when you run `cl-acltool -L`:

```
cumulus@switch:~$ sudo cl-acltool -L ip
-----
Listing rules of type iptables:
-----
TABLE filter :
Chain INPUT (policy ACCEPT 72 packets, 5236 bytes)
  pkts bytes target     prot opt in      out
source          destination
               icmp --  any      any
anywhere        anywhere           icmp echo-request
```

However, running `cl-acltool -i` or `reboot` removes them. To ensure all rules that can be in hardware are hardware accelerated, place them in `/etc/cumulus/acl/policy.conf` and run `cl-acltool -i`.



Where to Assign Rules

- If a switch port is assigned to a bond, any egress rules must be assigned to the bond.
- When using the OUTPUT chain, rules must be assigned to the source. For example, if a rule is assigned to the switch port in the direction of traffic but the source is a bridge (VLAN), the traffic won't be affected by the rule and must be applied to the bridge.
- If all transit traffic needs to have a rule applied, use the FORWARD chain, not the OUTPUT chain.

Generic Error Message Displayed after ACL Rule Installation Failure

After an ACL rule installation failure, a generic error message like the following is displayed:

```
cumulus@switch:$ sudo cl-acltool -i -p 00control_plane.rules
Using user provided rule file 00control_plane.rules
Reading rule file 00control_plane.rules ...
Processing rules in file 00control_plane.rules ...
error: hw sync failed (sync_acl hardware installation failed)
Installing acl policy... Rolling back ..
failed.
```

Dell S3048-ON Supports only 24K MAC Addresses

The Dell S3048-ON has a limit of 24576 MAC address entries, instead of 32K for other 1G switches.

Default Cumulus Linux ACL Configuration

The Cumulus Linux default ACL configuration is split into three parts, as outlined in the [netfilter ACL documentation](#): IP tables, IPv6 tables, and EB tables. The sections below cover the default configurations for each part, while the default file can be seen by clicking the Default ACL Configuration link:

Default ACL Configuration

```
cumulus@switch:~$ sudo cl-acltool -L all
-----
Listing rules of type iptables:
-----
TABLE filter :
Chain INPUT (policy ACCEPT 167 packets, 16481 bytes)
  pkts bytes target      prot opt in     out      source
destination
      0     0  DROP       all   --  swp+    any     240.0.0.0/5
anywhere
      0     0  DROP       all   --  swp+    any     loopback/8
anywhere
      0     0  DROP       all   --  swp+    any     base-address.mcast.net
/8  anywhere
      0     0  DROP       all   --  swp+    any     255.255.255.255
anywhere
```

```

      0      0 SETCLASS    udp  --  swp+   any     anywhere
anywhere          udp dpt:3785 SETCLASS  class:7
      0      0 POLICE     udp  --  any     any     anywhere
anywhere          udp dpt:3785 POLICE  mode:pkt rate:2000 burst:
2000
      0      0 SETCLASS    udp  --  swp+   any     anywhere
anywhere          udp dpt:3784 SETCLASS  class:7
      0      0 POLICE     udp  --  any     any     anywhere
anywhere          udp dpt:3784 POLICE  mode:pkt rate:2000 burst:
2000
      0      0 SETCLASS    udp  --  swp+   any     anywhere
anywhere          udp dpt:4784 SETCLASS  class:7
      0      0 POLICE     udp  --  any     any     anywhere
anywhere          udp dpt:4784 POLICE  mode:pkt rate:2000 burst:
2000
      0      0 SETCLASS    ospf --  swp+   any     anywhere
anywhere          SETCLASS  class:7
      0      0 POLICE     ospf --  any     any     anywhere
anywhere          POLICE  mode:pkt rate:2000 burst:2000
      0      0 SETCLASS    tcp   --  swp+   any     anywhere
anywhere          tcp dpt:bgp SETCLASS  class:7
      0      0 POLICE     tcp   --  any     any     anywhere
anywhere          tcp dpt:bgp POLICE  mode:pkt rate:2000 burst:2000
      0      0 SETCLASS    tcp   --  swp+   any     anywhere
anywhere          tcp spt:bgp SETCLASS  class:7
      0      0 POLICE     tcp   --  any     any     anywhere
anywhere          tcp spt:bgp POLICE  mode:pkt rate:2000 burst:2000
      0      0 SETCLASS    tcp   --  swp+   any     anywhere
anywhere          tcp dpt:5342 SETCLASS  class:7
      0      0 POLICE     tcp   --  any     any     anywhere
anywhere          tcp dpt:5342 POLICE  mode:pkt rate:2000 burst:
2000
      0      0 SETCLASS    tcp   --  swp+   any     anywhere
anywhere          tcp spt:5342 SETCLASS  class:7
      0      0 POLICE     tcp   --  any     any     anywhere
anywhere          tcp spt:5342 POLICE  mode:pkt rate:2000 burst:
2000
      0      0 SETCLASS    icmp --  swp+   any     anywhere
anywhere          SETCLASS  class:2
      1      84 POLICE    icmp --  any     any     anywhere
anywhere          POLICE  mode:pkt rate:100 burst:40
      0      0 SETCLASS    udp  --  swp+   any     anywhere
anywhere          udp dpts:bootps:bootpc SETCLASS  class:2
      0      0 POLICE     udp  --  any     any     anywhere
anywhere          udp dpt:bootps POLICE  mode:pkt rate:100 burst:
100
      0      0 POLICE     udp  --  any     any     anywhere
anywhere          udp dpt:bootpc POLICE  mode:pkt rate:100 burst:
100
      0      0 SETCLASS    tcp  --  swp+   any     anywhere
anywhere          tcp dpts:bootps:bootpc SETCLASS  class:2

```

```

      0      0 POLICE      tcp  --  any    any    anywhere
anywhere                                tcp dpt:bootps POLICE mode:pkt rate:100 burst:
100
      0      0 POLICE      tcp  --  any    any    anywhere
anywhere                                tcp dpt:bootpc POLICE mode:pkt rate:100 burst:
100
      0      0 SETCLASS    udp  --  swp+   any    anywhere
anywhere                                udp dpt:10001 SETCLASS class:3
      0      0 POLICE      udp  --  any    any    anywhere
anywhere                                udp dpt:10001 POLICE mode:pkt rate:2000 burst:
2000
      0      0 SETCLASS    igmp --  swp+   any    anywhere
anywhere                                SETCLASS class:6
      1      32 POLICE     igmp --  any    any    anywhere
anywhere                                POLICE mode:pkt rate:300 burst:100
      0      0 POLICE     all   --  swp+   any    anywhere
anywhere                                ADDRTYPE match dst-type LOCAL POLICE mode:pkt
rate:1000 burst:1000 class:0
      0      0 POLICE     all   --  swp+   any    anywhere
anywhere                                ADDRTYPE match dst-type IPROUTER POLICE mode:
pkt rate:400 burst:100 class:0
      0      0 SETCLASS    all   --  swp+   any    anywhere
anywhere                                SETCLASS class:0
Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source
destination
      0      0 DROP        all   --  swp+   any    240.0.0.0/5
anywhere
      0      0 DROP        all   --  swp+   any    loopback/8
anywhere
      0      0 DROP        all   --  swp+   any    base-address.mcast.net
/8 anywhere
      0      0 DROP        all   --  swp+   any    255.255.255.255
anywhere
Chain OUTPUT (policy ACCEPT 107 packets, 12590 bytes)
 pkts bytes target     prot opt in     out     source
destination
TABLE mangle :
Chain PREROUTING (policy ACCEPT 172 packets, 17871 bytes)
 pkts bytes target     prot opt in     out     source
destination
Chain INPUT (policy ACCEPT 172 packets, 17871 bytes)
 pkts bytes target     prot opt in     out     source
destination
Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source
destination
Chain OUTPUT (policy ACCEPT 111 packets, 18134 bytes)
 pkts bytes target     prot opt in     out     source
destination
Chain POSTROUTING (policy ACCEPT 111 packets, 18134 bytes)

```

```

pkts bytes target      prot opt in     out      source
destination
TABLE raw :
Chain PREROUTING (policy ACCEPT 173 packets, 17923 bytes)
  pkts bytes target      prot opt in     out      source
destination
Chain OUTPUT (policy ACCEPT 112 packets, 18978 bytes)
  pkts bytes target      prot opt in     out      source
destination
-----
Listing rules of type ip6tables:
-----
TABLE filter :
Chain INPUT (policy ACCEPT 0 packets, 0 bytes)
  pkts bytes target      prot opt in     out      source
destination
    0      0 DROP        all      swp+   any    ip6-mcastprefix/8
anywhere
    0      0 DROP        all      swp+   any    ::/128
anywhere
    0      0 DROP        all      swp+   any    ::ffff:0.0.0.0/96
anywhere
    0      0 DROP        all      swp+   any    localhost/128
anywhere
    0      0 POLICE     udp     swp+   any    anywhere
anywhere          udp dpt:3785 POLICE mode:pkt rate:2000 burst:
2000 class:7
    0      0 POLICE     udp     swp+   any    anywhere
anywhere          udp dpt:3784 POLICE mode:pkt rate:2000 burst:
2000 class:7
    0      0 POLICE     udp     swp+   any    anywhere
anywhere          udp dpt:4784 POLICE mode:pkt rate:2000 burst:
2000 class:7
    0      0 POLICE     ospf    swp+   any    anywhere
anywhere          POLICE mode:pkt rate:2000 burst:2000 class:7
    0      0 POLICE     tcp     swp+   any    anywhere
anywhere          tcp dpt:bgp POLICE mode:pkt rate:2000 burst:
2000 class:7
    0      0 POLICE     tcp     swp+   any    anywhere
anywhere          tcp spt:bgp POLICE mode:pkt rate:2000 burst:
2000 class:7
    0      0 POLICE     ipv6-icmp swp+   any    anywhere
anywhere          ipv6-icmp router-
solicitation POLICE mode:pkt rate:100 burst:100 class:2
    0      0 POLICE     ipv6-icmp swp+   any    anywhere
anywhere          ipv6-icmp router-
advertisement POLICE mode:pkt rate:500 burst:500 class:2
    0      0 POLICE     ipv6-icmp swp+   any    anywhere
anywhere          ipv6-icmp neighbour-
solicitation POLICE mode:pkt rate:400 burst:400 class:2

```



```
0      0 POLICE      ipv6-icmp      swp+    any
anywhere          anywhere          ipv6-icmp neighbour-
advertisement POLICE mode:pkt rate:400 burst:400 class:2
0      0 POLICE      ipv6-icmp      swp+    any
anywhere          anywhere          ipv6-icmptype 130 POLICE
mode:pkt rate:200 burst:100 class:6
0      0 POLICE      ipv6-icmp      swp+    any
anywhere          anywhere          ipv6-icmptype 131 POLICE
mode:pkt rate:200 burst:100 class:6
0      0 POLICE      ipv6-icmp      swp+    any
anywhere          anywhere          ipv6-icmptype 132 POLICE
mode:pkt rate:200 burst:100 class:6
0      0 POLICE      ipv6-icmp      swp+    any
anywhere          anywhere          ipv6-icmptype 143 POLICE
mode:pkt rate:200 burst:100 class:6
0      0 POLICE      ipv6-icmp      swp+    any
anywhere          anywhere          POLICE mode:pkt rate:64
burst:40 class:2
0      0 POLICE      udp       swp+    any     anywhere
anywhere          udp dpts:dhcpv6-client:dhcpv6-server POLICE
mode:pkt rate:100 burst:100 class:2
0      0 POLICE      tcp       swp+    any     anywhere
anywhere          tcp dpts:dhcpv6-client:dhcpv6-server POLICE
mode:pkt rate:100 burst:100 class:2
0      0 POLICE      all      swp+    any     anywhere
anywhere          ADDRTYPE match dst-type LOCAL POLICE mode:pkt
rate:1000 burst:1000 class:0
0      0 POLICE      all      swp+    any     anywhere
anywhere          ADDRTYPE match dst-type IPROUTER POLICE mode:
pkt rate:400 burst:100 class:0
0      0 SETCLASS    all      swp+    any     anywhere
anywhere          SETCLASS class:0
Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target      prot opt in      out      source
destination
0      0 DROP        all      swp+    any     ip6-mcastprefix/8
anywhere
0      0 DROP        all      swp+    any     ::/128
anywhere
0      0 DROP        all      swp+    any     ::ffff:0.0.0.0/96
anywhere
0      0 DROP        all      swp+    any     localhost/128
anywhere
Chain OUTPUT (policy ACCEPT 5 packets, 408 bytes)
 pkts bytes target      prot opt in      out      source
destination
TABLE mangle :
Chain PREROUTING (policy ACCEPT 7 packets, 718 bytes)
 pkts bytes target      prot opt in      out      source
destination
Chain INPUT (policy ACCEPT 0 packets, 0 bytes)
```

```

pkts bytes target      prot opt in      out      source
destination
Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
  pkts bytes target      prot opt in      out      source
destination
Chain OUTPUT (policy ACCEPT 0 packets, 0 bytes)
  pkts bytes target      prot opt in      out      source
destination
Chain POSTROUTING (policy ACCEPT 0 packets, 0 bytes)
  pkts bytes target      prot opt in      out      source
destination
TABLE raw :
Chain PREROUTING (policy ACCEPT 7 packets, 718 bytes)
  pkts bytes target      prot opt in      out      source
destination
Chain OUTPUT (policy ACCEPT 0 packets, 0 bytes)
  pkts bytes target      prot opt in      out      source
destination
-----
Listing rules of type ebtables:
-----
TABLE filter :
Bridge table: filter
Bridge chain: INPUT, entries: 16, policy: ACCEPT
-d BGA -i swp+ -j setclass --class 7 , pcnt = 0 -- bcnt = 0
-d BGA -j police --set-mode pkt --set-rate 2000 --set-burst 2000 ,
pcnt = 0 -- bcnt = 0
-d 1:80:c2:0:0:2 -i swp+ -j setclass --class 7 , pcnt = 0 -- bcnt = 0
-d 1:80:c2:0:0:2 -j police --set-mode pkt --set-rate 2000 --set-burst
2000 , pcnt = 0 -- bcnt = 0
-d 1:80:c2:0:0:e -i swp+ -j setclass --class 6 , pcnt = 0 -- bcnt = 0
-d 1:80:c2:0:0:e -j police --set-mode pkt --set-rate 200 --set-burst
200 , pcnt = 0 -- bcnt = 0
-d 1:0:c:cc:cc:cc -i swp+ -j setclass --class 6 , pcnt = 0 -- bcnt = 0
-d 1:0:c:cc:cc:cc -j police --set-mode pkt --set-rate 200 --set-burst
200 , pcnt = 0 -- bcnt = 0
-p ARP -i swp+ -j setclass --class 2 , pcnt = 0 -- bcnt = 0
-p ARP -j police --set-mode pkt --set-rate 400 --set-burst 100 , pcnt
= 0 -- bcnt = 0
-d 1:0:c:cc:cc:cd -i swp+ -j setclass --class 7 , pcnt = 0 -- bcnt = 0
-d 1:0:c:cc:cc:cd -j police --set-mode pkt --set-rate 2000 --set-
burst 2000 , pcnt = 0 -- bcnt = 0
-p IPv4 -i swp+ -j ACCEPT , pcnt = 0 -- bcnt = 0
-p IPv6 -i swp+ -j ACCEPT , pcnt = 0 -- bcnt = 0
-i swp+ -j setclass --class 0 , pcnt = 0 -- bcnt = 0
-j police --set-mode pkt --set-rate 100 --set-burst 100 , pcnt = 0 --
bcnt = 0
Bridge chain: FORWARD, entries: 0, policy: ACCEPT
Bridge chain: OUTPUT, entries: 0, policy: ACCEPT

```



IP Tables

Action/Value	Protocol/IP Address
Drop Destination IP: Any	Source IPv4: <ul style="list-style-type: none">• 240.0.0.0/5• loopback/8• 224.0.0.0/4• 255.255.255.255
Set class: 7 Police: Packet rate 2000 burst 2000 Source IP: Any Destination IP: Any	Protocol: <ul style="list-style-type: none">• UDP/BFD Echo• UDP/BFD Control• UDP BFD Multihop Control• OSPF• TCP/BGP (spt dpt 179)• TCP/MLAG (spt dpt 5342)
Set Class: 6 Police: Rate 300 burst 100 Source IP: Any Destination IP: Any	Protocol: <ul style="list-style-type: none">• IGMP
Set class: 2 Police: Rate 100 burst 40 Source IP : Any Destination IP: Any	Protocol: <ul style="list-style-type: none">• ICMP
Set class: 2 Police: Rate 100 burst 100 Source IP: Any Destination IP: Any	Protocol: <ul style="list-style-type: none">• UDP/bootpc, bootps
Set class: 3 Police: Rate 2000 burst:2000	Protocol: <ul style="list-style-type: none">• UDP/LNV

Action/Value	Protocol/IP Address
Source IP: Any Destination IP: Any	
Set class: 0 Police: Rate 1000 burst 1000 Source IP: Any Destination IP: Any	ADDRTYPE match dst-type LOCAL <div style="border: 2px solid #fbc02d; padding: 10px; margin-top: 10px;"> ⚠️ LOCAL is any local address -> Receiving a packet with a destination matching a local IP address on the switch will go to the CPU. </div>
Set class: 0 Police: Rate 400 burst 100 Source IP: Any Destination IP: Any	ADDRTYPE match dst-type IPROUTER <div style="border: 2px solid #fbc02d; padding: 10px; margin-top: 10px;"> ⚠️ IPROUTER is any unresolved address -> On a l2/l3 boundary receiving a packet from L3 and needs to go to CPU in order to ARP for the destination. </div>
Set class 0	All

⚠️ Set class is internal to the switch - it does not set any precedence bits.

IPv6 Tables

Action/Value	Protocol/IP Address
Drop	Source IPv6: <ul style="list-style-type: none"> • ff00::/8 • :: • ::ffff:0.0.0.0/96 • localhost
Set class: 7 Police: Packet rate 2000 burst 2000 Source IPv6: Any Destination IPv6: Any	Protocol: <ul style="list-style-type: none"> • UDP/BFD Echo • UDP/BFD Control • UDP BFD Multihop Control • OSPF • TCP/BGP (spt dpt 179)



Action/Value	Protocol/IP Address
Set class: 6 Police: Packet Rte: 200 burst 100 Source IPv6: Any Destination IPv6: Any	Protocol: <ul style="list-style-type: none">• Multicast Listener Query (MLD)• Multicast Listener Report (MLD)• Multicast Listener Done (MLD)• Multicast Listener Report V2
Set class: 2 Police: Packet rate: 100 burst 100 Source IPv6: Any Destination IPv6: Any	Protocol: <ul style="list-style-type: none">• ipv6-icmp router-solicitation
Set class: 2 Police: Packet rate: 500 burst 500 Source IPv6: Any Destination IPv6: Any	Protocol: <ul style="list-style-type: none">• ipv6-icmp router-advertisement POLICE
Set class: 2 Police: Packet rate: 400 burst 400 Source IPv6: Any Destination IPv6: Any	Protocol: <ul style="list-style-type: none">• ipv6-icmp neighbour-solicitation• ipv6-icmp neighbour-advertisement
Set class: 2 Police: Packet rate: 64 burst: 40 Source IPv6: Any Destination IPv6: Any	Protocol: <ul style="list-style-type: none">• Ipv6 icmp
Set class: 2 Police: Packet rate: 100 burst: 100	Protocol: UDP/dhcpv6-client:dhcpv6-server (Spts & dpts)

Action/Value	Protocol/IP Address
Source IPv6: Any Destination IPv6: Any	
Police: Packet rate: 1000 burst 1000 Source IPv6: Any Destination IPv6: Any	ADDRTYPE match dst-type LOCAL ⚠️ LOCAL is any local address -> Receiving a packet with a destination matching a local IPv6 address on the switch will go to the CPU.
Set class: 0 Police: Packet rate: 400 burst 100	ADDRTYPE match dst-type IPROUTER ⚠️ IPROUTER is an unresolved address -> On a l2/l3 boundary receiving a packet from L3 and needs to go to CPU in order to ARP for the destination.
Set class 0	All

⚠️ Set class is internal to the switch - it does not set any precedence bits.

EB Tables

Action/Value	Protocol/MAC Address
Set Class: 7 Police: packet rate: 2000 burst rate:2000 Any switchport input interface	BDPU LACP Cisco PVST
Set Class: 6 Police: packet rate: 200 burst rate: 200 Any switchport input interface	LLDP CDP
Set Class: 2 Police: packet rate: 400 burst rate: 100 Any switchport input interface	ARP
Catch All: Allow all traffic	IPv4 IPv6



Action/Value	Protocol/MAC Address
Any switchport input interface	
Catch All (applied at end): Set class: 0 Police: packet rate 100 burst rate 100 Any switchport	ALL OTHER



Set class is internal to the switch. It does not set any precedence bits.

Managing Application Daemons

You manage application daemons (services) in Cumulus Linux in the following ways:

- Identifying active listener ports
- Identifying daemons currently active or stopped
- Identifying boot time state of a specific daemon
- Disabling or enabling a specific daemon

Contents

This chapter covers ...

- Using `systemd` and the `systemctl` Command (see page 163)
 - Understanding the `systemctl` Subcommands (see page 164)
 - Ensuring a Service Starts after Multiple Restarts (see page 164)
 - Keeping `systemd` Services from Hanging after Starting (see page 165)
- Identifying Active Listener Ports for IPv4 and IPv6 (see page 165)
- Identifying Daemons Currently Active or Stopped (see page 166)
- Identifying Essential Services (see page 170)

Using `systemd` and the `systemctl` Command

In general, you manage services using `systemd` via the `systemctl` command. You use it with any service on the switch to start/stop/restart/reload/enable/disable/reenable or get the status of the service.

```
cumulus@switch:~$ sudo systemctl start | stop | restart | status |  
reload | enable | disable | reenable SERVICENAME.service
```

For example to restart networking, run the command:



```
cumulus@switch:~$ sudo systemctl restart networking.service
```



Unlike the `service` command in Debian Wheezy, the service name is written **after** the `systemctl` subcommand, not before it.

Understanding the `systemctl` Subcommands

`systemctl` has a number of subcommands that perform a specific operation on a given daemon.

- **status**: Returns the status of the specified daemon.
- **start**: Starts the daemon.
- **stop**: Stops the daemon.
- **restart**: Stops, then starts the daemon, all the while maintaining state. So if there are dependent services or services that mark the restarted service as *Required*, the other services also get restarted. For example, running `systemctl restart quagga.service` restarts any of the routing protocol daemons that are enabled and running, such as `bgpd` or `ospfd`.
- **reload**: Reloads a daemon's configuration.
- **enable**: Enables the daemon to start when the system boots, but does not start it unless you use the `systemctl start SERVICENAME.service` command or reboot the switch.
- **disable**: Disables the daemon, but does not stop it unless you use the `systemctl stop SERVICENAME.service` command or reboot the switch. A disabled daemon can still be started or stopped.
- **reenable**: Disables, then enables a daemon. You might need to do this so that any new *Wants* or *WantedBy* lines create the symlinks necessary for ordering. This has no side effects on other daemons.

Ensuring a Service Starts after Multiple Restarts

By default, `systemd` is configured to try to restart a particular service only a certain number of times within a given interval before the service fails to start at all. The settings for this are stored in the service script. The settings are *StartLimitInterval* (which defaults to 10 seconds) and *StartBurstLimit* (which defaults to 5 attempts), but many services override these defaults, sometimes with much longer times. `switchd.service`, for example, sets *StartLimitInterval=10m* and *StartBurstLimit=3*, which means if you restart `switchd` more than 3 times in 10 minutes, it will not start.

When the restart fails for this reason, a message similar to the following appears:

```
Job for switchd.service failed. See 'systemctl status switchd.service' and 'journalctl -xn' for details.
```

And `systemctl status switchd.service` shows output similar to:

```
Active: failed (Result: start-limit) since Thu 2016-04-07 21:55:14  
UTC; 15s ago
```



To clear this error, run `sudo systemctl reset-failed switchd.service`. If you know you are going to restart frequently (multiple times within the `StartLimitInterval`), you can run the same command before you issue the restart request. This also applies to stop followed by start.

Keeping systemd Services from Hanging after Starting

If you start, restart or reload any `systemd` service that could be started from another `systemd` service, you must use the `--no-block` option with `sudo systemctl`. Otherwise, that service or even the switch itself may hang after starting or restarting.

Identifying Active Listener Ports for IPv4 and IPv6

You can identify the active listener ports under both IPv4 and IPv6 using the `netstat` command:

```
cumulus@switch:~$ sudo netstat -nlp --inet --inet6
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address     State       PID/Program name
tcp        0      0 0.0.0.0:53              0.0.0.0:*          LISTEN      444/dnsmasq
tcp        0      0 0.0.0.0:22              0.0.0.0:*          LISTEN      874/sshd
tcp6       0      0 ::::53                 ::::*               LISTEN      444/dnsmasq
tcp6       0      0 ::::22                 ::::*               LISTEN      874/sshd
udp        0      0 0.0.0.0:28450            0.0.0.0:             udp         *          839/dhclient
udp        0      0 0.0.0.0:53              0.0.0.0:             udp         *          444/dnsmasq
udp        0      0 0.0.0.0:68              0.0.0.0:             udp         *          839/dhclient
udp        0      0 192.168.0.42:123          0.0.0.0:             udp         *          907/ntp
udp        0      0 127.0.0.1:123            0.0.0.0:             udp         *          907/ntp
udp        0      0 0.0.0.0:123              0.0.0.0:             udp         *          907/ntp
udp        0      0 0.0.0.0:4784            0.0.0.0:             udp         *          909/ptmd
udp        0      0 0.0.0.0:3784            0.0.0.0:             udp         *          909/ptmd
udp        0      0 0.0.0.0:3785            0.0.0.0:             udp         *          909/ptmd
udp6       0      0 ::::58352                ::::               udp         *          839/dhclient
udp6       0      0 ::::53                 ::::               udp         *          444/dnsmasq
udp6       0      0 fe80::a200:ff:fe00::123  ::::               udp         *          907/ntp
```

```
udp6      0      0  ::1:123          :::::907/ntpd
*          0      0  ::::123          :::::907/ntpd
udp6      0      0  ::::4784         :::::909/ptmd
*          0      0  ::::3784         :::::909/ptmd
```

Identifying Daemons Currently Active or Stopped

To determine which daemons are currently active or stopped, run `cl-service-summary`:

```
cumulus@switch:~$ cl-service-summary
Service cron           enabled   active
Service ssh            enabled   active
Service syslog          enabled   active
Service arp_refresh    enabled   active
Service clagd           enabled   active
Service lldpd           enabled   active
Service mstpd           enabled   active
Service poed            inactive
Service portwd          inactive
Service ptmd            enabled   active
Service pwmd            enabled   active
Service smond           enabled   active
Service switchd          enabled   active
Service vxrd             disabled  inactive
Service vxsnd            disabled  inactive
Service bgpd             disabled  inactive
Service isisd             disabled  inactive
Service ospf6d            disabled  inactive
Service ospfd             disabled  inactive
Service rdnbrd           disabled  inactive
Service ripd              disabled  inactive
Service ripngd           disabled  inactive
Service zebra             disabled  inactive
```

You can also run `systemctl list-unit-files --type service` to list all services on the switch and see which ones are enabled:

[Click here to see output of this command ...](#)

```
cumulus@switch:~$ systemctl list-unit-files --type service
UNIT FILE                                STATE
aclinit.service                            enabled
acltool.service                            enabled
acpid.service                             disabled
arp_refresh.service                         enabled
```



auditd.service	enabled
autovt@.service	disabled
bootlog.service	enabled
bootlogd.service	masked
bootlogs.service	masked
bootmisc.service	masked
checkfs.service	masked
checkroot-bootclean.service	masked
checkroot.service	masked
clagd.service	enabled
clcmd.service	enabled
console-getty.service	disabled
console-shell.service	disabled
container-getty@.service	static
cron.service	enabled
cryptdisks-early.service	masked
cryptdisks.service	masked
cumulus-aclcheck.service	static
cumulus-core.service	static
cumulus-fastfailover.service	enabled
cumulus-firstboot.service	disabled
cumulus-platform.service	enabled
cumulus-support.service	static
dbus-org.freedesktop.hostname1.service	static
dbus-org.freedesktop.locale1.service	static
dbus-org.freedesktop.login1.service	static
dbus-org.freedesktop.machine1.service	static
dbus-org.freedesktop.timedate1.service	static
dbus.service	static
debian-fixup.service	static
debug-shell.service	disabled
decode-syseeprom.service	static
dhcpd.service	disabled
dhcpd6.service	disabled
dhcpd6@.service	disabled
dhcpd@.service	disabled
dhcrelay.service	enabled
dhcrelay6.service	disabled
dhcrelay6@.service	disabled
dhcrelay@.service	disabled
dm-event.service	disabled
dns-watcher.service	disabled
dnsmasq.service	enabled
emergency.service	static
fuse.service	masked
getty-static.service	static
getty@.service	enabled
halt-local.service	static
halt.service	masked
heartbeat-failed@.service	static
hostname.service	masked
hsflowd.service	enabled

hsflowd.service	enabled
hwclock-save.service	enabled
hwclock.service	masked
hwclockfirst.service	masked
ifup@.service	static
initrd-cleanup.service	static
initrd-parse-etc.service	static
initrd-switch-root.service	static
initrd-udevadm-cleanup-db.service	static
killprocs.service	masked
kmod-static-nodes.service	static
kmod.service	static
ledmgrd.service	enabled
lldpd.service	enabled
lm-sensors.service	enabled
lvm2-activation-early.service	enabled
lvm2-activation.service	enabled
lvm2-lvmetad.service	static
lvm2-monitor.service	enabled
lvm2-pvscan@.service	static
lvm2.service	disabled
module-init-tools.service	static
motd.service	masked
mountall-bootclean.service	masked
mountall.service	masked
mountdevsubfs.service	masked
mountkernfs.service	masked
mountnfs-bootclean.service	masked
mountnfs.service	masked
mstpd.service	enabled
netd.service	enabled
netq-agent.service	disabled
networking.service	enabled
ntp.service	enabled
ntp@.service	disabled
openvswitch-vtep.service	disabled
phy-ucode-update.service	enabled
portwd.service	enabled
procps.service	static
ptmd.service	enabled
pwmd.service	enabled
quagga.service	enabled
quotaon.service	static
rc-local.service	static
rc.local.service	static
rdnbrd.service	disabled
reboot.service	masked
rescue.service	static
rmnlogin.service	masked
rsyslog.service	enabled
screen-cleanup.service	masked
sendsigs.service	masked



serial-getty@.service	disabled
single.service	masked
smond.service	enabled
snmpd.service	disabled
snmpd@.service	disabled
snmptrapd.service	disabled
snmptrapd@.service	disabled
ssh.service	enabled
ssh@.service	disabled
sshd.service	enabled
stop-bootlogd-single.service	masked
stop-bootlogd.service	masked
stopssh.service	enabled
sudo.service	disabled
switchd-diag.service	static
switchd.service	enabled
syslog.service	enabled
sysmonitor.service	static
systemd-ask-password-console.service	static
systemd-ask-password-wall.service	static
systemd-backlight@.service	static
systemd-binfmt.service	static
systemd-fsck-root.service	static
systemd-fsck@.service	static
systemd-halt.service	static
systemd-hibernate.service	static
systemd-hostnamed.service	static
systemd-hybrid-sleep.service	static
systemd-initctl.service	static
systemd-journal-flush.service	static
systemd-journald.service	static
systemd-kexec.service	static
systemd-located.service	static
systemd-logind.service	static
systemd-machined.service	static
systemd-modules-load.service	static
systemd-networkd-wait-online.service	disabled
systemd-networkd.service	disabled
systemd-nspawn@.service	disabled
systemd-poweroff.service	static
systemd-quotacheck.service	static
systemd-random-seed.service	static
systemd-readahead-collect.service	disabled
systemd-readahead-done.service	static
systemd-readahead-drop.service	disabled
systemd-readahead-replay.service	disabled
systemd-reboot.service	static
systemd-remount-fs.service	static
systemd-resolved.service	disabled
systemd-rfkill@.service	static
systemd-setup-dgram-qlen.service	static
systemd-shutdownd.service	static

```

systemd-suspend.service           static
systemd-sysctl.service           static
systemd-timedated.service        static
systemd-timesyncd.service       disabled
systemd-tmpfiles-clean.service   static
systemd-tmpfiles-setup-dev.service static
systemd-tmpfiles-setup.service   static
systemd-udev-settle.service     static
systemd-udev-trigger.service    static
systemd-udevd.service          static
systemd-update-utmp-runlevel.service static
systemd-update-utmp.service     static
systemd-user-sessions.service   static
udev-finish.service             static
udev.service                     static
umountfs.service                masked
umountnfs.service               masked
umountroot.service              masked
update-ports.service            enabled
urandom.service                 static
user@.service                   static
uuidd.service                   static
vboxadd-service.service         enabled
vboxadd-x11.service            enabled
vboxadd.service                 enabled
vxrd.service                    disabled
vxsnd.service                   disabled
wd_keepalive.service           enabled
x11-common.service             masked
ztp-init.service                enabled
ztp.service                     disabled
191 unit files listed.
lines 147-194/194 (END)

```

Identifying Essential Services

If you need to know which services are required to run when the switch boots, run:

```
cumulus@switch:~$ sudo systemctl list-dependencies --before basic.target
```

To see which services are needed for networking, run:

```
cumulus@switch:~$ sudo systemctl list-dependencies --after network.target
network.target
```



```
networking.service  
switchd.service  
wd_keepalive.service  
network-pre.target
```

To identify the services needed for a multi-user environment, run:

```
cumulus@leaf01:~$ sudo systemctl list-dependencies --before multi-user.  
target  
multi-user.target
```

```
bootlog.service  
systemd-readahead-done.service  
systemd-readahead-done.timer  
systemd-update-utmp-runlevel.service  
graphical.target  
systemd-update-utmp-runlevel.service
```

Configuring switchd

switchd is the daemon at the heart of Cumulus Linux. It communicates between the switch and Cumulus Linux, and all the applications running on Cumulus Linux.

The switchd configuration is stored in `/etc/cumulus/switchd.conf`.



Versions of Cumulus Linux prior to 2.1 stored the `switchd` configuration at `/etc/default/switchd`.

Contents

This chapter covers ...

- [The switchd File System \(see page 171\)](#)
- [Configuring switchd Parameters \(see page 173\)](#)
- [Restarting switchd \(see page 173\)](#)

The switchd File System

switchd also exports a file system, mounted on `/cumulus/switchd`, that presents all the switchd configuration options as a series of files arranged in a tree structure. You can see the contents by parsing the `switchd` tree; run `tree /cumulus/switchd`. The output below is for a switch with one switch port configured:

```
cumulus@switch:~$ sudo tree /cumulus/switchd/
/cumulus/switchd/
|-- config
|   |-- acl
|   |   |-- non_atomic_update_mode
|   |   `-- optimize_hw
|   |-- arp
|   |   `-- next_hops
|   |-- buf_util
|   |   |-- measure_interval
|   |   `-- poll_interval
|   |-- coalesce
|   |   |-- reducer
|   |   `-- timeout
|   |-- disable_internal_restart
|   |-- ignore_non_swps
|   |-- interface
|   |   |-- swp1
|   |   |   `-- storm_control
|   |   |       |-- broadcast
|   |   |       |-- multicast
|   |   |       `-- unknown_unicast
|   |-- logging
|   |-- route
|   |   |-- host_max_percent
|   |   `-- table
|   '-- stats
|       `-- poll_interval
|-- ctrl
|   |-- acl
|   |-- hal
|   |   `-- resync
|   |-- logger
|   |-- netlink
|   |   `-- resync
|   |-- resync
|   '-- sample
|       `-- ulog_channel
|-- run
|   '-- route_info
|       |-- ecmp_nh
|       |   |-- count
|       |   |-- max
|       |   `-- max_per_route
|       '-- host
|           |-- count
|           |-- count_v4
|           |-- count_v6
|           `-- max
|       '-- mac
|           |-- count
|           `-- max
```

```
    '-- route
        '-- count_0
        '-- count_1
        '-- count_total
        '-- count_v4
        '-- count_v6
        '-- mask_limit
        '-- max_0
        '-- max_1
        '-- max_total
-- version
```

Configuring switchd Parameters

You can use `cl-cfg` to configure many `switchd` parameters at runtime (like ACLs, interfaces, and route table utilization), which minimizes disruption to your running switch. However, some options are read only and cannot be configured at runtime.

For example, to see data related to routes, run:

```
cumulus@switch:~$ sudo cl-cfg -a switchd | grep route
route.table = 254
route.host_max_percent = 50
cumulus@cumulus:~$
```

To modify the configuration, run `cl-cfg -w`. For example, to set the buffer utilization measurement interval to 1 minute, run:

```
cumulus@switch:~$ sudo cl-cfg -w switchd buf_util.measure_interval=1
```

To verify that the value changed, use `grep`:

```
cumulus@switch:~$ cl-cfg -a switchd | grep buf
buf_util.poll_interval = 0
buf_util.measure_interval = 1
```



You can get some of this information by running `cl-resource-query`; though you cannot update the `switchd` configuration with it.

Restarting switchd

Whenever you modify any `switchd` hardware configuration file (typically changing any `*.conf` file that requires making a change to the switching hardware, like `/etc/cumulus/datapath/traffic.conf`), you must restart `switchd` for the change to take effect:



```
cumulus@switch:~$ sudo systemctl restart switchd.service
```



You do not have to restart the `switchd` service when you update a network interface configuration (that is, edit `/etc/network/interfaces`).



Restarting `switchd` causes all network ports to reset in addition to resetting the switch hardware configuration.

Power over Ethernet - PoE

Cumulus Linux supports Power over Ethernet (PoE) and PoE+, so certain Cumulus Linux switches can supply power from Ethernet switch ports to enabled devices over the Ethernet cables that connect them. Power over Ethernet (PoE) is capable of powering devices up to 15W, while PoE+ can power devices up to 30W.

The [currently supported platform](#) is the Edge-Core AS4610-54P, which supports PoE and PoE+ and configuration over Ethernet layer 2 LLDP for power negotiation.

Contents

This chapter covers ...

- [How It Works](#) (see page 174)
 - [About Link State and PoE State](#) (see page 175)
- [Configuring PoE](#) (see page 175)
 - [poectl Arguments](#) (see page 178)
- [Troubleshooting PoE and PoE+](#) (see page)
 - [Verify the Link Is Up](#) (see page 180)
 - [View LLDP Information Using llpdcli](#) (see page 180)
 - [View LLDP Information Using tcpdump](#) (see page 181)
 - [Logging poed Events in syslog](#) (see page 182)

How It Works

PoE functionality is provided by the `cumulus-poe` package. When a powered device is connected to the switch via an Ethernet cable:

- If the available power is greater than the power required by the connected device, power is supplied to the switch port, and the device powers on
- If available power is less than the power required by the connected device and the switch port's priority is less than the port priority set on all powered ports, power is **not** supplied to the port



- If available power is less than the power required by the connected device and the switch port's priority is greater than the priority of a currently powered port, power is removed from lower priority port(s) and power is supplied to the port
- If the total consumed power exceeds the configured power limit of the power source, low priority ports are turned off. In the case of a tie, the port with the lower port number gets priority

Power is available as follows:

PSU 1	PSU 2	PoE Power Budget
920W	X	750W
X	920W	750W
920W	920W	1650W

The AS4610-54P has an LED on the front panel to indicate PoE status:

- Green: The `poed` daemon is running and no errors are detected
- Yellow: One or more errors are detected or the `poed` daemon is not running

About Link State and PoE State

Link state and PoE state are completely independent of each other. When a link is brought down on a particular port using `ip link <port> down`, power on that port is not turned off; however, LLDP negotiation is not possible.

Configuring PoE

You use the `poectl` command utility to configure PoE on a [switch that supports](#) the feature. You can:

- Enable or disable PoE for a given switch port
- Set a switch port's PoE priority to one of three values: *low*, *high* or *critical*

The PoE configuration resides in `/etc/cumulus/poe.conf`. The file lists all the switch ports, whether PoE is enabled for those ports and the priority for each port.

Sample `poe.conf` file ...

```
[enable]
swp1 = enable
swp2 = enable
swp3 = enable
swp4 = enable
swp5 = enable
swp6 = enable
swp7 = enable
swp8 = enable
swp9 = enable
swp10 = enable
swp11 = enable
```



```
swp12 = enable
swp13 = enable
swp14 = enable
swp15 = enable
swp16 = enable
swp17 = enable
swp18 = enable
swp19 = enable
swp20 = enable
swp21 = enable
swp22 = enable
swp23 = enable
swp24 = enable
swp25 = enable
swp26 = enable
swp27 = enable
swp28 = enable
swp29 = enable
swp30 = enable
swp31 = enable
swp32 = enable
swp33 = enable
swp34 = enable
swp35 = enable
swp36 = enable
swp37 = enable
swp38 = enable
swp39 = enable
swp40 = enable
swp41 = enable
swp42 = enable
swp43 = enable
swp44 = enable
swp45 = enable
swp46 = enable
swp47 = enable
swp48 = enable
[priority]
swp1 = low
swp2 = low
swp3 = low
swp4 = low
swp5 = low
swp6 = low
swp7 = low
swp8 = low
swp9 = low
swp10 = low
swp11 = low
swp12 = low
swp13 = low
swp14 = low
```

```
swp15 = low
swp16 = low
swp17 = low
swp18 = low
swp19 = low
swp20 = low
swp21 = low
swp22 = low
swp23 = low
swp24 = low
swp25 = low
swp26 = low
swp27 = low
swp28 = low
swp29 = low
swp30 = low
swp31 = low
swp32 = low
swp33 = low
swp34 = low
swp35 = low
swp36 = low
swp37 = low
swp38 = low
swp39 = low
swp40 = low
swp41 = low
swp42 = low
swp43 = low
swp44 = low
swp45 = low
swp46 = low
swp47 = low
swp48 = low
```

By default, PoE and PoE+ are enabled on all Ethernet/1G switch ports, and these ports are set with a low priority. Switch ports can have low, high or critical priority.

There is no additional configuration for PoE+.

To change the priority for one or more switch ports, run `poectl -p swp# [low|high|critical]`. For example:

```
cumulus@switch:~$ sudo poectl -p swp1-swp5,swp7 high
```

To disable PoE for one or more ports, run `poectl -d [port_numbers]`:

```
cumulus@switch:~$ sudo poectl -d swp1-swp5,swp7
```

To display PoE information for a set of switch ports, run `poectl -i [port_numbers]`:



```
cumulus@switch:~$ sudo poectl -i swp10-swp13
Port          Status      Allocated   Priority  PD type
PD class     Voltage    Current    Power
-----  -----  -----  -----
-----  -----  -----  -----
swp10    connected    negotiating low      IEEE802.3at
4        53.5 V     25 mA     3.9 W
swp11    searching    n/a        low      IEEE802.3at
none     0.0 V       0 mA      0.0 W
swp12    connected    n/a        low      IEEE802.3at
2        53.5 V     25 mA     1.4 W
swp13    connected    51.0 W    low      IEEE802.3at
4        53.6 V     72 mA     3.8 W
```

The **Status** can be one of the following:

- **searching:** PoE is enabled but no device has been detected.
- **disabled:** The PoE port has been configured as disabled.
- **connected:** A powered device is connected and receiving power.
- **power-denied:** There is insufficient PoE power available to enable the connected device.

The **Allocated** column displays how much PoE power has been allocated to the port, which can be one of the following:

- **n/a:** No device is connected or the connected device does not support LLDP negotiation.
- **negotiating:** An LLDP-capable device is connected and is negotiating for PoE power.
- **XX.X W:** An LLDP-capable device has negotiated for XX.X watts of power (for example, 51.0 watts for swp13 above).

To see all the PoE information for a switch, run `poectl -s`:

```
cumulus@switch:~$ poectl -s
System power:
  Total:      730.0 W
  Used:       11.0 W
  Available:  719.0 W
Connected ports:
  swp11, swp24, swp27, swp48
```

The set commands (priority, enable, disable) either succeed silently or display an error message if the command fails.

poectl Arguments

The `poectl` command takes the following arguments:



Argument	Description
-h, --help	Show this help message and exit
-i, --port-info PORT_LIST	Returns detailed information for the specified ports. You can specify a range of ports. For example: <code>-i swp1-swp5,swp10</code> <div style="border: 2px solid yellow; padding: 10px; margin-top: 10px;"><p>⚠ On an Edge-Core AS4610-54P switch, the voltage reported by the <code>poectl -i</code> command and measured through a power meter connected to the device varies by 5V. The current and power readings are correct and no difference is seen for them.</p></div>
-a, --all	Returns PoE status and detailed information for all ports.
-p, --priority PORT_LIST PRIORITY	Sets priority for the specified ports: low, high, critical.
-d, --disable-ports PORT_LIST	Disables PoE operation on the specified ports.
-e, --enable-ports PORT_LIST	Enables PoE operation on the specified ports.
-s, --system	Returns PoE status for the entire switch.
-r, --reset PORT_LIST	Performs a hardware reset on the specified ports. Use this if one or more ports are stuck in an error state. This does not reset any configuration settings for the specified ports.
-v, --version	Displays version information.
-j, --json	Displays output in JSON format.
--save	Saves the current configuration. The saved configuration is automatically loaded on system boot.
--load	Loads and applies the saved configuration.



Troubleshooting PoE and PoE+

You can troubleshoot PoE and PoE+ using the following utilities and files:

- `poectl -s`, as described above.
- The Cumulus Linux `cl-support` script, which includes PoE-related output from `poed.conf`, `syslog`, `poectl --diag-info` and `lldpctl`.
- `lldpcli show neighbors ports <swp> protocol lldp hidden details`
- `tcpdump -v -v -i <swp> ether proto 0x88cc`
- The contents of the PoE/PoE+ `/etc/lldpd.d/poed.conf` configuration file, as described above.

Verify the Link Is Up

LLDP requires network connectivity, so verify that the link is up.

```
cumulus@switch:~$ net show interface swp20
  Name      MAC                Speed      MTU    Mode
--  -----  -----
UP   swp20   44:38:39:00:00:04  1G        1500  Access/L2
```

View LLDP Information Using `lldpcli`

You can run `lldpcli` to view the LLDP information that has been received on a switch port. For example:

```
cumulus@switch:~$ sudo lldpcli show neighbors ports swp20 protocol
lldp hidden details
-----
-----
LLDP neighbors:
-----
-----
Interface:      swp20, via: LLDP, RID: 2, Time: 0 day, 00:03:34
  Chassis:
    ChassisID:      mac 68:c9:0b:25:54:7c
    SysName:        ihm-ubuntu
    SysDescr:       Ubuntu 14.04.2 LTS Linux 3.14.4+ #1 SMP Thu Jun 26
00:54:44 UTC 2014 armv7l
    MgmtIP:         fe80::6ac9:bff:fe25:547c
    Capability:    Bridge, off
    Capability:    Router, off
    Capability:    Wlan, off
    Capability:    Station, on
  Port:
    PortID:        mac 68:c9:0b:25:54:7c
    PortDescr:     eth0
    PMD autoneg:   supported: yes, enabled: yes
    Adv:           10Base-T, HD: yes, FD: yes
```



```
Adv: 100Base-TX, HD: yes, FD: yes
MAU oper type: 100BaseTXFD - 2 pair category 5 UTP, full duplex
mode
MDI Power: supported: yes, enabled: yes, pair control: no
Device type: PD
Power pairs: spare
Class: class 4
Power type: 2
Power Source: Primary power source
Power Priority: low
PD requested power Value: 51000
PSE allocated power Value: 51000
UnknownTLVs:
TLV: OUI: 00,01,42, SubType: 1, Len: 1 05
TLV: OUI: 00,01,42, SubType: 1, Len: 1 0D
```

View LLDP Information Using tcpdump

You can use `tcpdump` to view the LLDP frames being transmitted and received. For example:

```
cumulus@switch:~$ sudo tcpdump -v -v -i swp20 ether proto 0x88cc
tcpdump: listening on swp20, link-type EN10MB (Ethernet), capture
size 262144 bytes
18:41:47.559022 LLDP, length 211
    Chassis ID TLV (1), length 7
        Subtype MAC address (4): 00:30:ab:f2:d7:a5 (oui Unknown)
        0x0000: 0400 30ab f2d7 a5
    Port ID TLV (2), length 6
        Subtype Interface Name (5): swp20
        0x0000: 0573 7770 3230
    Time to Live TLV (3), length 2: TTL 120s
        0x0000: 0078
    System Name TLV (5), length 13: dni-3048up-09
        0x0000: 646e 692d 3330 3438 7570 2d30 39
    System Description TLV (6), length 68
        Cumulus Linux version 3.0.1~1466303042.2265c10 running on dni
3048up
        0x0000: 4375 6d75 6c75 7320 4c69 6e75 7820 7665
        0x0010: 7273 696f 6e20 332e 302e 317e 3134 3636
        0x0020: 3330 3330 3432 2e32 3236 3563 3130 2072
        0x0030: 756e 6e69 6e67 206f 6e20 646e 6920 3330
        0x0040: 3438 7570
    System Capabilities TLV (7), length 4
        System Capabilities [Bridge, Router] (0x0014)
        Enabled Capabilities [Router] (0x0010)
        0x0000: 0014 0010
    Management Address TLV (8), length 12
```

```

Management Address length 5, AFI IPv4 (1): 10.0.3.190
Interface Index Interface Numbering (2): 2
0x0000: 0501 0a00 03be 0200 0000 0200
Management Address TLV (8), length 24
Management Address length 17, AFI IPv6 (2): fe80::230:abff:fef2:
d7a5
Interface Index Interface Numbering (2): 2
0x0000: 1102 fe80 0000 0000 0000 0230 abff fef2
0x0010: d7a5 0200 0000 0200
Port Description TLV (4), length 5: swp20
0x0000: 7377 7032 30
Organization specific TLV (127), length 9: OUI IEEE 802.3 Private
(0x00120f)
Link aggregation Subtype (3)
    aggregation status [supported], aggregation port ID 0
    0x0000: 0012 0f03 0100 0000 00
Organization specific TLV (127), length 9: OUI IEEE 802.3 Private
(0x00120f)
MAC/PHY configuration/status Subtype (1)
    autonegotiation [supported, enabled] (0x03)
    PMD autoneg capability [10BASE-T fdx, 100BASE-TX fdx,
1000BASE-T fdx] (0x2401)
    MAU type 100BASEFX fdx (0x0012)
    0x0000: 0012 0f01 0324 0100 12
Organization specific TLV (127), length 12: OUI IEEE 802.3
Private (0x00120f)
Power via MDI Subtype (2)
    MDI power support [PSE, supported, enabled], power pair
spare, power class class4
    0x0000: 0012 0f02 0702 0513 01fe 01fe
Organization specific TLV (127), length 5: OUI Unknown (0x000142)
    0x0000: 0001 4201 0d
Organization specific TLV (127), length 5: OUI Unknown (0x000142)
    0x0000: 0001 4201 01
End TLV (0), length 0

```

Logging poed Events in syslog

The poed service logs the following events to syslog:

- When a switch provides power to a powered device
- When a device that was receiving power is removed
- When the power available to the switch changes
- Errors



Configuring a Global Proxy

You configure [global HTTP and HTTPS proxies](#) in the `/etc/profile.d/` directory of Cumulus Linux. To do so, set the `http_proxy` and `https_proxy` variables, which tells the switch the address of the proxy server to use to fetch URLs on the command line. This is useful for programs such as `apt/apt-get`, `curl` and `wget`, which can all use this proxy.

1. In a terminal, create a new file in the `/etc/profile.d/` directory. In the code example below, the file is called `proxy.sh`, and is created using the text editor `nano`.

```
cumulus@switch:~$ sudo nano /etc/profile.d/proxy.sh
```

2. Add a line to the file to configure either an HTTP or an HTTPS proxy, or both:

- HTTP proxy:

```
http_proxy=http://myproxy.domain.com:8080
export http_proxy
```

- HTTPS proxy:

```
https_proxy=https://myproxy.domain.com:8080
export https_proxy
```

3. Create a file in the `/etc/apt/apt.conf.d/` directory and add the following lines to the file for acquiring the HTTP and HTTPS proxies; the example below uses `http_proxy.sh` as the file name:

```
cumulus@switch:~$ sudo nano /etc/apt/apt.conf.d/http_proxy.sh
Acquire::http::Proxy "http://myproxy.domain.com:8080";
Acquire::https::Proxy "https://myproxy.domain.com:8080";
```

4. Add the proxy addresses to `/etc/wgetrc`; you may have to uncomment the `http_proxy` and `https_proxy` lines:

```
cumulus@switch:~$ sudo nano /etc/wgetrc
...
https_proxy = https://myproxy.domain.com:8080
http_proxy = http://myproxy.domain.com:8080
...
```

5. Run the `source` command, to execute the file in the current environment:



```
cumulus@switch:~$ source /etc/profile.d/proxy.sh
```

The proxy is now configured. The echo command can be used to confirm a proxy is set up correctly:

- HTTP proxy:

```
cumulus@switch:~$ echo $http_proxy  
http://myproxy.domain.com:8080
```

- HTTPS proxy:

```
cumulus@switch:~$ echo $https_proxy  
https://myproxy.domain.com:8080
```

Related Information

- [Setting up an apt package cache](#)

Interface Configuration and Management

`ifupdown` is the network interface manager for Cumulus Linux. Cumulus Linux 2.1 and later uses an updated version of this tool, `ifupdown2`.

For more information on network interfaces, see [Layer 1 and Switch Port Attributes \(see page 201\)](#).



By default, `ifupdown` is quiet; use the verbose option `-v` when you want to know what is going on when bringing an interface down or up.

Contents

This chapter covers ...

- [Basic Commands \(see page 185\)](#)
- [ifupdown2 Interface Classes \(see page 186\)](#)
 - [Bringing All auto Interfaces Up or Down \(see page 187\)](#)
 - [Configuring a Loopback Interface \(see page 188\)](#)
 - [ifupdown Behavior with Child Interfaces \(see page 188\)](#)
 - [ifupdown2 Interface Dependencies \(see page 189\)](#)
 - [ifup Handling of Upper \(Parent\) Interfaces \(see page 192\)](#)
 - [Configuring IP Addresses \(see page 193\)](#)
 - [Specifying IP Address Scope \(see page 194\)](#)
 - [Purging Existing IP Addresses on an Interface \(see page 195\)](#)
 - [Specifying User Commands \(see page 195\)](#)
 - [Sourcing Interface File Snippets \(see page 196\)](#)
 - [Using Globs for Port Lists \(see page 197\)](#)
 - [Using Templates \(see page 198\)](#)
 - [Commenting out Mako Templates \(see page 199\)](#)
 - [Adding Descriptions to Interfaces \(see page 199\)](#)
 - [Caveats and Errata \(see page 200\)](#)
 - [Related Information \(see page 200\)](#)

Basic Commands

To bring up an interface or apply changes to an existing interface, run:



```
cumulus@switch:~$ sudo ifup <ifname>
```

To bring down a single interface, run:

```
cumulus@switch:~$ sudo ifdown <ifname>
```



`ifdown` always deletes logical interfaces after bringing them down. Use the `--admin-state` option if you only want to administratively bring the interface up or down.

To see the link and administrative state, use the `ip link show` command:

```
cumulus@switch:~$ ip link show dev swp1
3: swp1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast
state UP mode DEFAULT qlen 500
    link/ether 44:38:39:00:03:c1 brd ff:ff:ff:ff:ff:ff
```

In this example, `swp1` is administratively UP and the physical link is UP (`LOWER_UP` flag). More information on interface administrative state and physical state can be found in [this knowledge base article](#).

ifupdown2 Interface Classes

`ifupdown2` provides for the grouping of interfaces into separate classes, where a class is simply a user-defined label used to group interfaces that share a common function (like uplink, downlink or compute). You specify classes in `/etc/network/interfaces`.

The most common class users are familiar with is `auto`, which you configure like this:

```
auto swp1
iface swp1
```

You can add other classes using the `allow` prefix. For example, if you have multiple interfaces used for uplinks, you can make up a class called `uplinks`:

```
auto swp1
allow-uplink swp1
iface swp1 inet static
    address 10.1.1.1/31

auto swp2
allow-uplink swp2
iface swp2 inet static
    address 10.1.1.3/31
```



This allows you to perform operations on only these interfaces using the --allow-uplinks option, or still use the -a options since these interfaces are also in the auto class:

```
cumulus@switch:~$ sudo ifup --allow=uplinks
cumulus@switch:~$ sudo ifreload -a
```

Another example where this feature is useful is if you're using [Management VRF \(see page 593\)](#), you can use the special interface class called *mgmt*, and put the management interface into that class:

```
allow-mgmt eth0
iface eth0 inet dhcp
    vrf mgmt

allow-mgmt mgmt
iface mgmt
    address 127.0.0.1/8
    vrf-table auto
```

All *ifupdown2* commands (*ifup*, *ifdown*, *ifquery*, *ifreload*) can take a class. Include the --allow=<class> option when you run the command. For example, to reload the configuration for the management interface described above, run:

```
cumulus@switch:~$ sudo ifreload --allow=mgmt
```

Bringing All auto Interfaces Up or Down

You can easily bring up or down all interfaces marked with the common *auto* class in */etc/network/interfaces*. Use the -a option. For further details, see individual man pages for *ifup(8)*, *ifdown(8)*, *ifreload(8)*.

To administratively bring up all interfaces marked *auto*, run:

```
cumulus@switch:~$ sudo ifup -a
```

To administratively bring down all interfaces marked *auto*, run:

```
cumulus@switch:~$ sudo ifdown -a
```

To reload all network interfaces marked *auto*, use the *ifreload* command, which is equivalent to running *ifdown* then *ifup*, the one difference being that *ifreload* skips any configurations that didn't change):

```
cumulus@switch:~$ sudo ifreload -a
```



Configuring a Loopback Interface

Cumulus Linux has a loopback preconfigured in `/etc/network/interfaces`. When the switch boots up, it has a loopback interface, called `lo`, which is up and assigned an IP address of 127.0.0.1.

- ✓ The loopback interface `lo` must always be specified in `/etc/network/interfaces` and must always be up.

ifupdown Behavior with Child Interfaces

By default, `ifupdown` recognizes and uses any interface present on the system — whether a VLAN, bond or physical interface — that is listed as a dependent of an interface. You are not required to list them in the `interfaces` file unless they need a specific configuration, for [MTU, link speed, and so forth \(see page 201\)](#). And if you need to delete a child interface, you should delete all references to that interface from the `interfaces` file.

For this example, `swp1` and `swp2` below do not need an entry in the `interfaces` file. The following stanzas defined in `/etc/network/interfaces` provide the exact same configuration:

With Child Interfaces Defined

```
auto swp1
iface swp1

auto swp2
iface swp2

auto bridge
iface bridge
    bridge-vlan-aware yes
    bridge-ports swp1
swp2
    bridge-vids 1-100
    bridge-pvid 1
    bridge-stp on
```

Without Child Interfaces Defined

```
auto bridge
iface bridge
    bridge-vlan-aware yes
    bridge-ports swp1
swp2
    bridge-vids 1-100
    bridge-pvid 1
    bridge-stp on
```

Bridge in Traditional Mode - Example

For this example, `swp1.100` and `swp2.100` below do not need an entry in the `interfaces` file. The following stanzas defined in `/etc/network/interfaces` provide the exact same configuration:

With Child Interfaces Defined

```
auto swp1.100
```

Without Child Interfaces Defined

```
auto br-100
```

```

iface swp1.100
auto swp2.100
iface swp2.100
auto br-100
iface br-100
    address 10.0.12.2/24
    address 2001:dad:beef::3/64
    bridge-ports swp1.100 swp2.
100
    bridge-stp on

```

```

iface br-100
address 10.0.12.2/24
address 2001:dad:beef::3/64
bridge-ports swp1.100 swp2.
100
bridge-stp on

```

For more information on the bridge in traditional mode vs the bridge in VLAN-aware mode, please read [this knowledge base article](#).

ifupdown2 Interface Dependencies

`ifupdown2` understands interface dependency relationships. When `ifup` and `ifdown` are run with all interfaces, they always run with all interfaces in dependency order. When run with the interface list on the command line, the default behavior is to not run with dependents. But if there are any built-in dependents, they will be brought up or down.

To run with dependents when you specify the interface list, use the `--with-dependents` option. `--with-dependents` walks through all dependents in the dependency tree rooted at the interface you specify. Consider the following example configuration:

```

auto bond1
iface bond1
    address 100.0.0.2/16
    bond-slaves swp29 swp30

auto bond2
iface bond2
    address 100.0.0.5/16
    bond-slaves swp31 swp32

auto br2001
iface br2001
    address 12.0.1.3/24
    bridge-ports bond1.2001 bond2.2001
    bridge-stp on

```

Using `ifup --with-dependents br2001` brings up all dependents of `br2001`: `bond1.2001`, `bond2.2001`, `bond1`, `bond2`, `bond1.2001`, `bond2.2001`, `swp29`, `swp30`, `swp31`, `swp32`.

```
cumulus@switch:~$ sudo ifup --with-dependents br2001
```

Similarly, specifying `ifdown --with-dependents br2001` brings down all dependents of `br2001`: `bond1.2001`, `bond2.2001`, `bond1`, `bond2`, `bond1.2001`, `bond2.2001`, `swp29`, `swp30`, `swp31`, `swp32`.

```
cumulus@switch:~$ sudo ifdown --with-dependents br2001
```

! As mentioned earlier, `ifdown2` always deletes logical interfaces after bringing them down. Use the `--admin-state` option if you only want to administratively bring the interface up or down. In terms of the above example, `ifdown br2001` deletes `br2001`.

To guide you through which interfaces will be brought down and up, use the `--print-dependency` option to get the list of dependents.

Use `ifquery --print-dependency=list -a` to get the dependency list of all interfaces:

```
cumulus@switch:~$ sudo ifquery --print-dependency=list -a
lo : None
eth0 : None
bond0 : ['swp25', 'swp26']
bond1 : ['swp29', 'swp30']
bond2 : ['swp31', 'swp32']
br0 : ['bond1', 'bond2']
bond1.2000 : ['bond1']
bond2.2000 : ['bond2']
br2000 : ['bond1.2000', 'bond2.2000']
bond1.2001 : ['bond1']
bond2.2001 : ['bond2']
br2001 : ['bond1.2001', 'bond2.2001']
swp40 : None
swp25 : None
swp26 : None
swp29 : None
swp30 : None
swp31 : None
swp32 : None
```

To print the dependency list of a single interface, use:

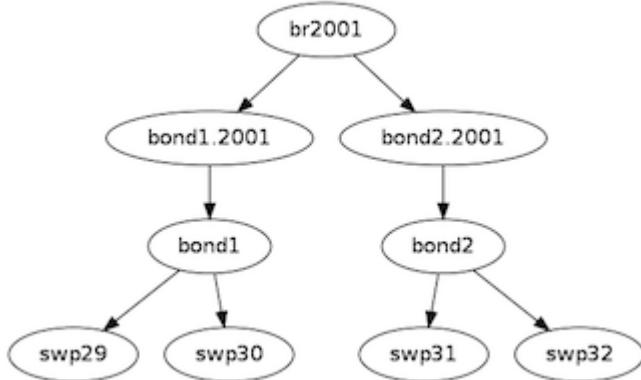
```
cumulus@switch:~$ sudo ifquery --print-dependency=list br2001
br2001 : ['bond1.2001', 'bond2.2001']
bond1.2001 : ['bond1']
bond2.2001 : ['bond2']
bond1 : ['swp29', 'swp30']
bond2 : ['swp31', 'swp32']
swp29 : None
swp30 : None
swp31 : None
```

```
swp32 : None
```

To print the dependency information of an interface in dot format:

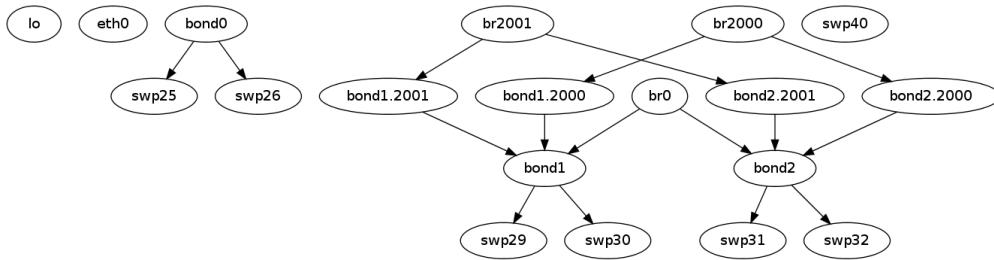
```
cumulus@switch:~$ sudo ifquery --print-dependency=dot br2001
/* Generated by GvGen v.0.9 (http://software.inl.fr/trac/wiki/GvGen)
*/
digraph G {
    compound=true;
    node1 [label="br2001"];
    node2 [label="bond1.2001"];
    node3 [label="bond2.2001"];
    node4 [label="bond1"];
    node5 [label="bond2"];
    node6 [label="swp29"];
    node7 [label="swp30"];
    node8 [label="swp31"];
    node9 [label="swp32"];
    node1->node2;
    node1->node3;
    node2->node4;
    node3->node5;
    node4->node6;
    node4->node7;
    node5->node8;
    node5->node9;
}
```

You can use dot to render the graph on an external system where dot is installed.



To print the dependency information of the entire interfaces file:

```
cumulus@switch:~$ sudo ifquery --print-dependency=dot -a
>interfaces_all.dot
```



ifup Handling of Upper (Parent) Interfaces

When you run `ifup` on a logical interface (like a bridge, bond or VLAN interface), if the `ifup` resulted in the creation of the logical interface, by default it implicitly tries to execute on the interface's upper (or parent) interfaces as well. This helps in most cases, especially when a bond is brought down and up, as in the example below. This section describes the behavior of bringing up the upper interfaces.

Consider this example configuration:

```

auto br100
iface br100
    bridge-ports bond1.100 bond2.100

auto bond1
iface bond1
    bond-slaves swp1 swp2

```

If you run `ifdown bond1`, `ifdown` deletes bond1 and the VLAN interface on bond1 (bond1.100); it also removes bond1 from the bridge br100. Next, when you run `ifup bond1`, it creates bond1 and the VLAN interface on bond1 (bond1.100); it also executes `ifup br100` to add the bond VLAN interface (bond1.100) to the bridge br100.

As you can see above, implicitly bringing up the upper interface helps, but there can be cases where an upper interface (like br100) is not in the right state, which can result in warnings. The warnings are mostly harmless.

If you want to disable these warnings, you can disable the implicit upper interface handling by setting `skip_upperinterfaces=1` in `/etc/network/ifupdown2/ifupdown2.conf`.

With `skip_upperinterfaces=1`, you will have to explicitly execute `ifup` on the upper interfaces. In this case, you will have to run `ifup br100` after an `ifup bond1` to add bond1 back to bridge br100.



Although specifying a subinterface like `swp1.100` and then running `ifup swp1.100` will also result in the automatic creation of the `swp1` interface in the kernel, Cumulus Networks recommends you specify the parent interface `swp1` as well. A parent interface is one where any physical layer configuration can reside, such as `link-speed 1000` or `link-duplex full`.

It's important to note that if you only create `swp1.100` and not `swp1`, then you cannot run `ifup swp1` since you did not specify it.



Configuring IP Addresses

IP addresses are configured with the `net add interface` command.

ⓘ Example IP Address Configuration

The following commands configure three IP addresses for `swp1`: two IPv4 addresses, and one IPv6 address.

```
cumulus@switch:~$ net add interface swp1 ip address 12.0.0.1/30
cumulus@switch:~$ net add interface swp1 ip address 12.0.0.2/30
cumulus@switch:~$ net add interface swp1 ipv6 address 2001:DB8::1/126
cumulus@switch:~$ net pending
cumulus@switch:~$ net commit
```



You can specify both IPv4 and IPv6 addresses for the same interface.

These commands create the following code snippet:

```
auto swp1
iface swp1
    address 12.0.0.1/30
    address 12.0.0.2/30
    address 2001:DB8::1/126
```



The address method and address family are added by NCLU when needed, specifically when you are creating DHCP or loopback interfaces.

```
auto lo
iface lo inet loopback
```

To show the assigned address on an interface, use `ip addr show`:

```
cumulus@switch:~$ ip addr show dev swp1
3: swp1: <BROADCAST,MULTICAST,SLAVE,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP qlen 500
    link/ether 44:38:39:00:03:c1 brd ff:ff:ff:ff:ff:ff
```

```
inet 192.0.2.1/30 scope global swp1
inet 192.0.2.2/30 scope global swp1
inet6 2001:DB8::1/126 scope global tentative
    valid_lft forever preferred_lft forever
```

Specifying IP Address Scope

`ifupdown2` does not honor the configured IP address scope setting in `/etc/network/interfaces`, treating all addresses as global. It does not report an error. Consider this example configuration:

```
auto swp2
iface swp2
    address 35.21.30.5/30
    address 3101:21:20::31/80
    scope link
```

When you run `ifreload -a` on this configuration, `ifupdown2` considers all IP addresses as global.

```
cumulus@switch:~$ ip addr show swp2
5: swp2: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast
state UP group default qlen 1000
link/ether 74:e6:e2:f5:62:82 brd ff:ff:ff:ff:ff:ff
inet 35.21.30.5/30 scope global swp2
    valid_lft forever preferred_lft forever
inet6 3101:21:20::31/80 scope global
    valid_lft forever preferred_lft forever
inet6 fe80::76e6:e2ff:fef5:6282/64 scope link
    valid_lft forever preferred_lft forever
```

To work around this issue, configure the IP address scope:

ⓘ Example post-up Configuration

```
cumulus@switch:~$ net add interface swp6 post-up ip address
add 71.21.21.20/32 dev swp6 scope site
cumulus@switch:~$ net pending
cumulus@switch:~$ net commit
```

These commands create the following code snippet in the `/etc/network/interfaces` file:

```
auto swp6
iface swp6
    post-up ip address add 71.21.21.20/32 dev swp6 scope site
```



Now it has the correct scope:

```
cumulus@switch:~$ ip addr show swp6
9: swp6: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast
    state UP group default qlen 1000
    link/ether 74:e6:e2:f5:62:86 brd ff:ff:ff:ff:ff:ff
    inet 71.21.21.20/32 scope site swp6
        valid_lft forever preferred_lft forever
    inet6 fe80::76e6:e2ff:fef5:6286/64 scope link
        valid_lft forever preferred_lft forever
```

Purging Existing IP Addresses on an Interface

By default, `ifupdown2` purges existing IP addresses on an interface. If you have other processes that manage IP addresses for an interface, you can disable this feature including the `address-purge` setting in the interface's configuration.

```
cumulus@switch:~$ net add interface swp1 address-purge no
cumulus@switch:~$ net pending
cumulus@switch:~$ net commit
```

These commands create the following configuration snippet in the `/etc/network/interfaces` file:

```
auto swp1
iface swp1
    address-purge no
```



Purging existing addresses on interfaces with multiple `iface` stanzas is not supported. Doing so can result in the configuration of multiple addresses for an interface after you change an interface address and reload the configuration with `ifreload -a`. If this happens, you must shut down and restart the interface with `ifup` and `ifdown`, or manually delete superfluous addresses with `ip address delete specify.ip.address.here/mask dev DEVICE`. See also the [Caveats and Errata \(see page 200\)](#) section below for some cautions about using multiple `iface` stanzas for the same interface.

Specifying User Commands

You can specify additional user commands in the `interfaces` file. As shown in the example below, the interface stanzas in `/etc/network/interfaces` can have a command that runs at pre-up, up, post-up, pre-down, down, and post-down:

```
cumulus@switch:~$ net add interface swp1 post-up /sbin/foo bar
cumulus@switch:~$ net add interface ip address 12.0.0.1/30
cumulus@switch:~$ net pending
cumulus@switch:~$ net commit
```

These commands create the following configuration in the `/etc/network/interfaces` file:

```
auto swp1
iface swp1
    address 12.0.0.1/30
    post-up /sbin/foo bar
```

Any valid command can be hooked in the sequencing of bringing an interface up or down, although commands should be limited in scope to network-related commands associated with the particular interface.

For example, it wouldn't make sense to install some Debian package on `ifup` of `swp1`, even though that is technically possible. See `man interfaces` for more details.



If your `post-up` command also starts, restarts or reloads any `systemd` service, you must use the `--no-block` option with `systemctl`. Otherwise, that service or even the switch itself may hang after starting or restarting.

For example, to restart the `dhcrelay` service after bringing up VLAN 100, first run:

```
cumulus@switch:~$ net add vlan 100 post-up systemctl --no-block restart dhcrelay.service
```

This command creates the following configuration in the `/etc/network/interfaces` file:

```
auto bridge
iface bridge
    bridge-vids 100
    bridge-vlan-aware yes

auto vlan100
iface vlan100
    post-up systemctl --no-block restart dhcrelay.service
    vlan-id 100
    vlan-raw-device bridge
```

Sourcing Interface File Snippets

Sourcing interface files helps organize and manage the `interfaces` file. For example:



```
cumulus@switch:~$ cat /etc/network/interfaces
# The loopback network interface
auto lo
iface lo inet loopback

# The primary network interface
auto eth0
iface eth0 inet dhcp

source /etc/network/interfaces.d/bond0
```

The contents of the sourced file used above are:

```
cumulus@switch:~$ cat /etc/network/interfaces.d/bond0
auto bond0
iface bond0
    address 14.0.0.9/30
    address 2001:ded:beef:2::1/64
    bond-slaves swp25 swp26
```

Using Globs for Port Lists

NCLU supports globs to define port lists (that is, a range of ports). The `glob` keyword is implied when you specify bridge ports and bond slaves:

```
cumulus@switch:~$ net add bridge bridge ports swp1-4,6,10-12
cumulus@switch:~$ net pending
cumulus@switch:~$ net commit
```

These commands produce the following snippet in the `/etc/network/interfaces` file:

```
...
auto swp1
iface swp1

auto swp2
iface swp2

auto swp3
iface swp3

auto swp4
iface swp4
```



```
auto swp6
iface swp6

auto swp10
iface swp10

auto swp11
iface swp11

auto swp12
iface swp12
```

Using Templates

`ifupdown2` supports [Mako-style templates](#). The Mako template engine is run over the `interfaces` file before parsing.

Use the template to declare cookie-cutter bridges in the `interfaces` file:

```
%for v in [11,12]:
auto vlan${v}
iface vlan${v}
    address 10.20.${v}.3/24
    bridge-ports glob swp19-20.${v}
    bridge-stp on
%endfor
```

And use it to declare addresses in the `interfaces` file:

```
%for i in [1,12]:
auto swp${i}
iface swp${i}
    address 10.20.${i}.3/24
```



Regarding Mako syntax, use square brackets ([1,12]) to specify a list of individual numbers (in this case, 1 and 12). Use `range(1,12)` to specify a range of interfaces.



You can test your template and confirm it evaluates correctly by running `mako-render /etc/network/interfaces`.



For more examples of configuring Mako templates, read this [knowledge base article](#).



Commenting out Mako Templates

To comment out content in Mako templates, use double hash marks (##). For example:

```
## % for i in range(1, 4):
## auto swp${i}
## iface swp${i}
## % endfor
##
```

Adding Descriptions to Interfaces

You can add descriptions to the interfaces configured in /etc/network/interfaces by using the *alias* keyword.

ⓘ Example Alias Configuration

The following commands create an alias for swp1:

```
cumulus@switch:~$ net add interface swp1 alias
hypervisor_port_1
cumulus@switch:~$ net pending
cumulus@switch:~$ net commit
```

These commands create the following code snippet:

```
auto swp1
iface swp1
    alias hypervisor_port_1
```

You can query the interface description using NCLU. The alias appears in the **Name** column after the actual interface name:

```
cumulus@switch$ net show interface swp1
      Name          MAC          Speed        MTU      Mode
--  -----
UP   swp1 (hypervisor_port_1)  44:38:39:00:00:04  1G        1500
Access/L2
```

Interface descriptions also appear in the [SNMP OID \(see page 638\)](#) IF-MIB::ifAlias.



 Aliases are limited to 256 characters.

Caveats and Errata

While `ifupdown2` supports the inclusion of multiple `iface` stanzas for the same interface, Cumulus Networks recommends you use a single `iface` stanza for each interface, if possible.

There are cases where you must specify more than one `iface` stanza for the same interface. For example, the configuration for a single interface can come from many places, like a template or a sourced file.

If you do specify multiple `iface` stanzas for the same interface, make sure the stanzas do not specify the same interface attributes. Otherwise, unexpected behavior can result.

For example, `swp1` is configured in two places:

```
cumulus@switch:~$ cat /etc/network/interfaces  
  
source /etc/interfaces.d/speed_settings  
  
auto swp1  
iface swp1  
    address 10.0.14.2/24
```

As well as `/etc/interfaces.d/speed_settings`

```
cumulus@switch:~$ cat /etc/interfaces.d/speed_settings  
  
auto swp1  
iface swp1  
    link-speed 1000  
    link-duplex full
```

`ifupdown2` correctly parses a configuration like this because the same attributes are not specified in multiple `iface` stanzas.

And, as stated in the note above, you cannot purge existing addresses on interfaces with multiple `iface` stanzas.

Related Information

- [Debian - Network Configuration](#)
- [Linux Foundation - Bonds](#)
- [Linux Foundation - Bridges](#)
- [Linux Foundation - VLANs](#)
- [man ifdown\(8\)](#)



- man ifquery(8)
- man ifreload
- man ifup(8)
- man ifupdown-addons-interfaces(5)
- man interfaces(5)

Layer 1 and Switch Port Attributes

This chapter discusses the various network interfaces on a switch running Cumulus Linux, how to configure various interface-level settings (if needed) and some troubleshooting commands.

Contents

This chapter covers ...

- Interface Types (see page 201)
- Interface Settings (see page 201)
 - Differences between Broadcom-based and Mellanox-based Switches (see page 202)
 - Enabling Auto-negotiation (see page 202)
 - Default Interface Configuration Settings (see page 203)
 - Port Speed and Duplexing (see page 209)
 - MTU (see page 210)
- Configuring Breakout Ports (see page 212)
 - Combining Four 10G Ports into One 40G Port (see page 214)
- Logical Switch Port Limitations (see page 214)
- Using ethtool to Configure Interfaces (see page 215)
- Verification and Troubleshooting Commands (see page 215)
 - Statistics (see page 215)
 - Querying SFP Port Information (see page 217)
- Caveats and Errata (see page 217)
 - Timeout Error on Quanta LY8 and LY9 Switches (see page 217)
- Related Information (see page 217)

Interface Types

Cumulus Linux exposes network interfaces for several types of physical and logical devices:

- lo, network loopback device
- ethN, switch management port(s), for out of band management only
- swpN, switch front panel ports
- (optional) brN, bridges (IEEE 802.1Q VLANs)
- (optional) bondN, bonds (IEEE 802.3ad link aggregation trunks, or port channels)



Interface Settings

Each physical network interface has a number of configurable settings:

- Auto-negotiation
- Duplex
- Forward error correction
- Link speed
- MTU, or [maximum transmission unit](#)

Almost all of these settings are configured automatically for you, depending upon your switch ASIC, although you must always set MTU manually.



You can only set MTU for logical interfaces. If you try to set auto-negotiation, duplex mode or link speed for a logical interface, an unsupported error gets returned.

Differences between Broadcom-based and Mellanox-based Switches

On a Broadcom-based switch, all you need to do is enable auto-negotiation. Once enabled, Cumulus Linux automatically configures the link speed, duplex mode and [forward error correction](#) (FEC, if the cable or optic requires it) for every switch port, based on the switch model and cable or optic used on the port, as listed in the [table below \(see page 203\)](#).

Ports are always automatically configured on a Mellanox-based switch, with one exception — you only need to configure is [MTU \(see page 210\)](#). You don't even need to enable auto-negotiation, as the Mellanox firmware configures everything for you.

Enabling Auto-negotiation

To configure auto-negotiation for a Broadcom-based switch, set `link-autoneg` to `on` for all the switch ports. For example, to enable auto-negotiation for `swp1` through `swp52`:

```
cumulus@switch:~$ net add interface swp1-52 link autoneg on
cumulus@switch:~$ net pending
cumulus@switch:~$ net commit
```

Any time you enable auto-negotiation, Cumulus Linux restores the default configuration settings specified in the [table below \(see page \)](#).

By default on a Broadcom-based switch, auto-negotiation is disabled — except on 10G and 1G BASE-T switches, where it's required for links to work at all. And for RJ45-SFP converters, you need to manually configure the settings as described in the [default settings table below \(see page \)](#).

If you disable it later or never enable it, then you have to configure the duplex, FEC and link speed settings manually using [NCLU \(see page 80\)](#) — see the relevant sections below. The default speed if you disable auto-negotiation depends on the type of connector used with the port. For example, a QSFP28 optic defaults to 100G, while a QSFP+ optic defaults to 40G and SFP+ defaults to 10G.



You cannot or should not disable auto-negotiation off for any type of copper cable, including:

- 10G BASE-T



- 10G DAC
- 40G DAC
- 100G DAC

However, RJ-45 (10/100/1000 BASE-T) adapters do not work with auto-negotiation enabled. You must manually configure these ports using the settings below (link-autoneg=off, link-speed=1000|100|10, link-duplex=full|half).

Depending upon the connector used for a port, enabling auto-negotiation also enables forward error correction (FEC), if the cable requires it (see the [table below \(see page \)](#)). FEC always adjusts for the speed of the cable. However, you **cannot** disable FEC separately using [NCLU \(see page 80\)](#).

Default Interface Configuration Settings

On a Broadcom-based switch, the configuration for each type of interface is described in the following table. Except as noted below, the settings for both sides of the link are expected to be the same.



If the other side of the link is running a version of Cumulus Linux earlier than 3.2, depending up on the interface type, auto-negotiation may not work on that switch. Cumulus Networks recommends you use the default settings on this switch in this case.

For Mellanox-based switches, the Spectrum firmware decides on the best settings based on the switch model and connector type.

Speed	Auto-negotiation	FEC Setting	Manual Configuration Steps	Notes
10/100 (RJ45)	Off	N/A (does not apply at this speed)	<pre>\$ net add interface swp1 link speed 100 \$ net add interface swp1 link autoneg off</pre> Configuration in /etc/network/interfaces <pre>auto swp1 iface swp1 link- autoneg off</pre>	<ul style="list-style-type: none">The module has two sets of electronics — the port side, which communicates to the switch ASIC, and the RJ45 side.Auto-negotiation is always used on the RJ45 side of the link by the PHY built into the module. This is independent of the switch setting. Set link-autoneg to off.Auto-negotiation needs to be enabled on the server side in this scenario.



Speed	Auto-negotiation	FEC Setting	Manual Configuration Steps	Notes
			<pre>link-speed 100</pre>	
1000BASE-SX, 1000BASE-LX, 1000BASE-CX (1G Fiber)	Recommended On	N/A	<pre>\$ net add interface swp1 link speed 1000 \$ net add interface swp1 link autoneg on</pre> Configuration in /etc/network/interfaces <pre>auto swp1 iface swp1 link- autoneg on link-speed 1000</pre>	<ul style="list-style-type: none">Without auto-negotiation, the link stays up when there is a single fiber break.
1000BASE-T (RJ45)	Off	N/A	<pre>\$ net add interface swp1 link speed 1000 \$ net add interface swp1 link autoneg off</pre> Configuration in /etc/network/interfaces <pre>auto swp1 iface swp1 link- autoneg off link-speed 1000</pre>	<ul style="list-style-type: none">The module has two sets of electronics — the port side, which communicates to the switch ASIC, and the RJ45 side.Auto-negotiation is always used on the RJ45 side of the link by the PHY built into the module. This is independent of the switch setting. Set link-autoneg to off.Auto-negotiation needs to be enabled on the server side in this scenario.



Speed	Auto-negotiation	FEC Setting	Manual Configuration Steps	Notes
			<pre>auto swp1 iface swp1 link- autoneg off link-speed 1000</pre>	
10G BASE-T	On	N/A	<pre>\$ net add interface swp1 link speed 10000 \$ net add interface swp1 link autoneg on</pre> <p>Configuration in /etc/network/interfaces</p> <pre>auto swp1 iface swp1 link- autoneg on link-speed 10000</pre>	
10G BASE-CR, 10G BASE-LR, 10G BASE-SR, 10G AOC	Off	N/A	<pre>\$ net add interface swp1 link speed 10000 \$ net add interface swp1 link autoneg off</pre>	



Speed	Auto-negotiation	FEC Setting	Manual Configuration Steps	Notes
			<p>Configuration in /etc/network/interfaces</p> <pre>auto swp1 iface swp1 link- autoneg off link-speed 10000 link-duplex full</pre>	
40G BASE-CR	Recommended On	Disable it	<pre>\$ net add interface swp1 link speed 40000 \$ net add interface swp1 link autoneg on</pre> <p>Configuration in /etc/network/interfaces</p> <pre>auto swp1 iface swp1 link- autoneg on link-speed 40000</pre>	
40G BASE-SR, 40G BASE-LR, 40G AOC	Off	Disable it	<pre>\$ net add interface swp1 link speed 40000</pre>	



Speed	Auto-negotiation	FEC Setting	Manual Configuration Steps	Notes
			<pre>\$ net add interface swp1 link autoneg off</pre> Configuration in /etc/network/interfaces <pre>auto swp1 iface swp1 link- autoneg off link-speed 40000</pre>	
100G BASE-CR	On	auto-negotiated	<pre>\$ net add interface swp1 link speed 100000 \$ net add interface swp1 link autoneg on</pre> Configuration in /etc/network/interfaces <pre>auto swp1 iface swp1 link- autoneg on link-speed 100000</pre>	
	Off	RS		



Speed	Auto-negotiation	FEC Setting	Manual Configuration Steps	Notes
100G BASE-SR, 100G BASE-LR, 100G AOC			<pre>\$ net add interface swp1 link speed 100000 \$ net add interface swp1 link autoneg off \$ net add interface swp1 link fec rs</pre> <p>Configuration in /etc/network/interfaces</p> <pre>auto swp1 iface swp1 link- autoneg off link-speed 100000 link-fec rs</pre>	
25G BASE-CR	On	auto-negotiated*	<pre>\$ net add interface swp1 link speed 25000 \$ net add interface swp1 link autoneg on</pre> <p>Configuration in /etc/network/interfaces</p> <pre>auto swp1</pre>	



Speed	Auto-negotiation	FEC Setting	Manual Configuration Steps	Notes
			<pre>iface swp1 link- autoneg on link-speed 25000</pre>	
25G BASE-SR, 25G BASE-LR	Off	Base-R	<pre>\$ net add interface swp1 link speed 25000 \$ net add interface swp1 link autoneg off \$ net add interface swp1 link fec baser</pre> Configuration in /etc /network /interfaces <pre>auto swp1 iface swp1 link- autoneg off link-speed 25000 link-fec baser</pre>	

Port Speed and Duplexing

Cumulus Linux supports both half- and **full-duplex** configurations. Supported port speeds include 100M, 1G, 10G, 25G, 40G, 50G and 100G. If you need to manually set the speed on a Broadcom-based switch, set it in terms of Mbps, where the setting for 1G is 1000, 40G is 40000 and 100G is 100000, for example.

The duplex mode setting defaults to *full*. You only need to specify `link duplex` if you want half-duplex mode.



ⓘ Example Port Speed and Duplexing Configuration

The following NCLU commands configure the port speed for the swp1 interface:

```
cumulus@switch:~$ net add interface swp1 link speed 10000
cumulus@switch:~$ net pending
cumulus@switch:~$ net commit
```

The above commands create the following /etc/network/interfaces code snippet:

```
auto swp1
iface swp1
    link-speed 10000
```

Port Speed Limitations

Ports can be configured to one speed less than their maximum speed.

Switch port Type	Lowest Configurable Speed
1G	100 Mb
10G	1 Gigabit (1000 Mb)
40G	10G*
100G	50G & 40G (with or without breakout port), 25G*, 10G*

*Requires the port to be converted into a breakout port. [See below \(see page 212\)](#).

MTU

Interface MTU ([maximum transmission unit](#)) applies to traffic traversing the management port, front panel /switch ports, bridge, VLAN subinterfaces and bonds — in other words, both physical and logical interfaces.

MTU is the only interface setting that must be set manually.

In Cumulus Linux, ifupdown2 assigns 1500 as the default MTU setting. You can override this default value by specifying a policy file in /etc/network/ifupdown2/policy.d/, like in the following example:

```
cumulus@switch:~$ cat /etc/network/ifupdown2/policy.d/address.json
{
    "address": {
        "defaults": { "mtu": "9000" }
    }
}
```

MTU for a Bridge

The MTU setting is the lowest MTU setting of any interface that is a member of that bridge (that is, every interface specified in `bridge-ports` in the bridge configuration in the `interfaces` file), even if another bridge member has a higher MTU value. There is **no** need to specify an MTU on the bridge. Consider this bridge configuration:

```
auto bridge
iface bridge
    bridge-ports bond1 bond2 bond3 bond4 peer5
    bridge-vids 100-110
    bridge-vlan-aware yes
```

In order for `bridge` to have an MTU of 9000, set the MTU for each of the member interfaces (bond1 to bond 4, and peer5), to 9000 at minimum.

ⓘ Use MTU 9216 for a bridge

Two common MTUs for jumbo frames are 9216 and 9000 bytes. The corresponding MTUs for the VNIs would be 9166 and 8950.

When configuring MTU for a bond, configure the MTU value directly under the bond interface; the configured value is inherited by member links/slave interfaces. If you need a different MTU on the bond, set it on the bond interface, as this ensures the slave interfaces pick it up. There is no need to specify MTU on the slave interfaces.

VLAN interfaces inherit their MTU settings from their physical devices or their lower interface; for example, `swp1.100` inherits its MTU setting from `swp1`. Hence, specifying an MTU on `swp1` ensures that `swp1.100` inherits `swp1`'s MTU setting.

If you are working with [VXLANS \(see page 346\)](#), the MTU for a virtual network interface (VNI) must be 50 bytes smaller than the MTU of the physical interfaces on the switch, as those 50 bytes are required for various headers and other data. You should also consider setting the MTU much higher than the default 1500.

ⓘ Example MTU Configuration

In general, the policy file specified above handles default MTU settings for all interfaces on the switch. If you need to configure a different MTU setting for a subset of interfaces, use [NCLU \(see page 80\)](#).

The following commands configure an MTU minimum value of 9000 on `swp1`:

```
cumulus@switch:~$ net add interface swp1 mtu 9000
cumulus@switch:~$ net pending
cumulus@switch:~$ net commit
```

These commands create the following code snippet:



```
auto swp1
iface swp1
    mtu 9000
```



You must take care to ensure there are no MTU mismatches in the conversation path. MTU mismatches will result in dropped or truncated packets, degrading or blocking network performance.

To view the MTU setting, use `net show interface <interface>`:

```
cumulus@switch:~$ net show interface swp1
  Name      MAC                  Speed      MTU      Mode
  --  -----  -----
UP   swp1    44:38:39:00:00:04  1G        1500    Access/L2
```

Configuring Breakout Ports

Cumulus Linux has the ability to:

- Break out 100G switch ports into the following with breakout cables:
 - 2x50G, 4x25G, 4x10G
- Break out 40G switch ports into four separate 10G ports for use with breakout cables.
- Combine (also called *aggregating* or *ganging*) four 10G switch ports into one 40G port for use with a breakout cable ([not to be confused with a bond \(see page 268\)](#)).

To configure a 4x25G breakout port, run:

```
cumulus@switch:~$ net add interface swp1 breakout 4x
cumulus@switch:~$ net pending
cumulus@switch:~$ net commit
```

The breakout port configuration is stored in the `/etc/cumulus/ports.conf` file.



`/etc/cumulus/ports.conf` varies across different hardware platforms. Check the current list of supported platforms on the [hardware compatibility list](#).

A snippet from the `/etc/cumulus/ports.conf` looks on a Tomahawk switch like this:

```
# The Dell Z9100 has:
#
#      32 QSFP28 ports numbered 1-32
#      These ports are configurable as 100G, 50G, 40G, 2x50G,
#      4x25G, 4x10G
```

```

#      or disabled.
#
#      Two SFP+ ports. These ports are configurable as 10G or
disabled.
#
#      The system can only handle 128 logical ports.
#
#      This means that if all 32 QSFP28 ports are broken out
into
#      4x25G or 4x10G mode, the two 10G ports (33 and 34) must
be
#      set to "disabled".
# If you make changes to this file, you must restart switchd fo
r the
# changes to take effect.
# QSFP28 ports
#
# <port label 1-32> = [4x10G|4x25G|2x50G|40G|50G|100G|disabled]
1=4x25G
2=100G
3=100G
4=100G

...
# SFP+ ports
#
# <port label 33-34> = [10G|disabled]
33=disabled
34=disabled

```

Notice that you can break out any of the 100G ports into a variety of options: four 10G ports, four 25G ports or two 50G ports. Keep in mind that you cannot have more than 128 total logical ports on a Broadcom switch.



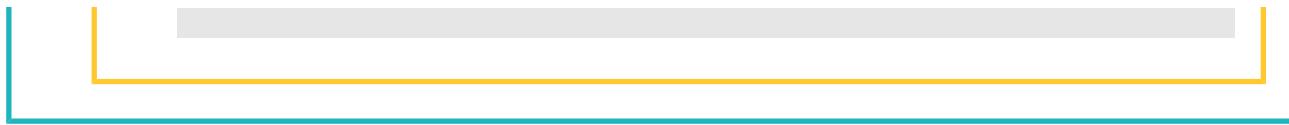
For both 100G and 40G switches using Mellanox Spectrum chipsets, there is a limit of 64 logical ports in total. However, the logical ports must be configured as follows:

- You can only break out odd-numbered ports into 4 logical ports.
- You must disable the next even-numbered port.

For example, if you have a 100G Mellanox SN-2700 switch and configure port 11 as 4x25G logical ports, you must configure port 12 as disabled. In `/etc/cumulus/ports.conf`:

```

...
11=4x25G
12=disabled
...
```



Combining Four 10G Ports into One 40G Port

You can *gang* (or aggregate) four 10G ports into one 40G port for use with a breakout cable, provided you follow these requirements:

- You must gang four 10G ports in sequential order. For example, you cannot gang swp1, swp10, swp20 and swp40 together.
- The ports must be in increments of four, with the starting port being swp1 (or swp5, swp9, or so forth); so you cannot gang swp2, swp3, swp4 and swp5 together.

For example, to gangs swp1 through swp4 into a 40G port, run:

```
cumulus@switch:~$ net add int swp1-4 breakout /4
cumulus@switch:~$ net pending
cumulus@switch:~$ net commit
```

These commands create the following configuration snippet in the `/etc/cumulus/ports.conf` file:

```
# SFP+ ports#
# <port label 1-48> = [10G|40G/4]
1=40G/4
2=40G/4
3=40G/4
4=40G/4
5=10G
```

Logical Switch Port Limitations

100G and 40G switches with Spectrum, Tomahawk, Trident II and Trident II+ chipsets (check the [HCL](#)) can support a certain number of logical ports, depending upon the manufacturer.

Before you configure any logical/unganged ports on a switch, check the limitations listed in `/etc/cumulus/ports.conf`; this file is specific to each manufacturer.

For example, the Dell S6000 `ports.conf` file indicates the logical port limitation like this:

```
# ports.conf --
#
# This file controls port aggregation and subdivision. For example,
QSFP+
# ports are typically configurable as either one 40G interface or four
# 10G/1000/100 interfaces. This file sets the number of interfaces
per port
# while /etc/network/interfaces and ethtool configure the link speed f
or each
```

```
# interface.
#
# You must restart switchd for changes to take effect.
#
# The DELL S6000 has:
#   32 QSFP ports numbered 1-32
#   These ports are configurable as 40G, split into 4x10G ports or
#   disabled.
#
#   The X pipeline covers QSFP ports 1 through 16 and the Y pipeline
#   covers QSFP ports 17 through 32.
#
#   The Trident2 chip can only handle 52 logical ports per pipeline.
#
#   This means 13 is the maximum number of 40G ports you can ungang
#   per pipeline, with the remaining three 40G ports set to
#   "disabled". The 13 40G ports become 52 unganged 10G ports, which
#   totals 52 logical ports for that pipeline.
```

The means the maximum number of ports for this Dell S6000 is 104.

For switches using Mellanox Spectrum chipsets, there is a limit of 64 logical ports in total. However, the logical ports must be configured in a specific way. See [the note \(see page 212\)](#) above.

Using ethtool to Configure Interfaces

The Cumulus Linux ethtool command is an alternative for configuring interfaces as well as viewing and troubleshooting them.

For example, to manually set link speed, auto-negotiation, duplex mode and FEC on swp1, run:

```
cumulus@switch:~$ sudo ethtool -s swp1 speed 25000 autoneg off duplex
full
cumulus@switch:~$ sudo ethtool --set-fec swp1 encoding off
```

To view the FEC setting on an interface, run:

```
cumulus@switch:~$ sudo ethtool --show-fec swp1
Auto-negotiation: off
FEC encodings : RS
```

Verification and Troubleshooting Commands

Statistics

High-level interface statistics are available with the `net show interface` command:



```
cumulus@switch:~$ net show interface swp1

      Name      MAC                  Speed      MTU  Mode
--  -----  -----
UP  swp1    44:38:39:00:00:04   1G        1500 Access/L2

Vlans in disabled State
-----
br0

Counters      TX      RX
-----  -----
errors        0       0
unicast       0       0
broadcast     0       0
multicast     0       0

LLDP
-----
swp1      ===  44:38:39:00:00:03 (server01)
```

Low-level interface statistics are available with ethtool:

```
cumulus@switch:~$ sudo ethtool -S swp1
NIC statistics:
  HwIfInOctets: 21870
  HwIfInUcastPkts: 0
  HwIfInBcastPkts: 0
  HwIfInMcastPkts: 243
  HwIfOutOctets: 1148217
  HwIfOutUcastPkts: 0
  HwIfOutMcastPkts: 11353
  HwIfOutBcastPkts: 0
  HwIfInDiscards: 0
  HwIfInL3Drops: 0
  HwIfInBufferDrops: 0
  HwIfInAclDrops: 0
  HwIfInBlackholeDrops: 0
  HwIfInDot3LengthErrors: 0
  HwIfInErrors: 0
  SoftInErrors: 0
  SoftInDrops: 0
  SoftInFrameErrors: 0
  HwIfOutDiscards: 0
  HwIfOutErrors: 0
  HwIfOutQDrops: 0
  HwIfOutNonQDrops: 0
```



```
SoftOutErrors: 0
SoftOutDrops: 0
SoftOutTxFifoFull: 0
HwIfOutQLen: 0
```

Querying SFP Port Information

You can verify SFP settings using `ethtool -m`. The following example shows the output for 1G and 10G modules:

```
cumulus@switch:~# sudo ethtool -m | egrep '(swp|RXPower :|TXPower :|EthernetComplianceCode)'

swp1: SFP detected
    EthernetComplianceCodes : 1000BASE-LX
    RXPower : -10.4479dBm
    TXPower : 18.0409dBm
swp3: SFP detected
    10GEthernetComplianceCode : 10G Base-LR
    RXPower : -3.2532dBm
    TXPower : -2.0817dBm
```

Caveats and Errata

Timeout Error on Quanta LY8 and LY9 Switches

On Quanta T5048-LY8 and T3048-LY9 switches, an "Operation timed out" error occurs while removing and reinserting QSFP module.

The QSPFx2 module cannot be removed while the switch is powered on, as it is not hot-swappable. However, if this occurs, you can get the link to come up; however, this involves [restarting `switchd`](#) (see [page 173](#)), which disrupts your network.

On the T3048-LY9, run the following commands:

```
cumulus@switch:~$ sudo echo 0 > qsfpd_power_enable/value
cumulus@switch:~$ sudo rmmod quanta_ly9_rangeley_platform
cumulus@switch:~$ sudo modprobe quanta_ly9_rangeley_platform
cumulus@switch:~$ sudo systemctl restart switchd.service
```

On the T5048-LY8, run the following commands:

```
cumulus@switch:~$ sudo echo 0 > qsfpd_power_enable/value
cumulus@switch:~$ sudo systemctl restart switchd.service
```



Related Information

- Debian - Network Configuration
- Linux Foundation - VLANs
- Linux Foundation - Bridges
- Linux Foundation - Bonds

Buffer and Queue Management

Hardware datapath configuration manages packet buffering, queueing and scheduling in hardware. There are two configuration input files:

- `/etc/cumulus/datapath/traffic.conf`, which describes priority groups and assigns the scheduling algorithm and weights
- `/usr/lib/python2.7/dist-packages/cumulus/__chip_config/[bcm|mlx]/datapath.conf`, which assigns buffer space and egress queues



Versions of these files prior to Cumulus Linux 2.1 are incompatible with Cumulus Linux 2.1 and later; using older files causes `switchd` to fail to start and, on Broadcom switches, returns an error that it cannot find the `/var/lib/cumulus/rc.datapath` file.

Each packet is assigned to an ASIC Class of Service (CoS) value based on the packet's priority value stored in the 802.1p (Class of Service) or DSCP (Differentiated Services Code Point) header field. The choice to schedule packets based on COS or DSCP is a configurable option in the `/etc/cumulus/datapath/traffic.conf` file.

Priority groups include:

- *Control*: Highest priority traffic
- *Service*: Second-highest priority traffic
- *Bulk*: All remaining traffic

The scheduler is configured to use a hybrid scheduling algorithm. It applies strict priority to control traffic queues and a weighted round robin selection from the remaining queues. Unicast packets and multicast packets with the same priority value are assigned to separate queues, which are assigned equal scheduling weights.

Datapath configuration takes effect when you initialize `switchd`. Changes to the `traffic.conf` file require you to [restart the `switchd` \(see page 173\)](#) service.

Contents

This chapter covers ...

- [Commands \(see page 219\)](#)
- [Configuration Files \(see page 219\)](#)
- [Configuring Traffic Marking through ACL Rules \(see page 221\)](#)
- [Configuring Priority Flow Control \(see page 222\)](#)
 - [Understanding Port Groups \(see page 224\)](#)
- [Configuring Link Pause \(see page 225\)](#)



- Configuring Explicit Congestion Notification (see page 226)
 - Configuring ECN (see page 226)
- Caveats and Errata (see page 227)
- Related Information (see page 227)

Commands

If you modify the configuration in the `/etc/cumulus/datapath/traffic.conf` file, you must [restart `switchd`](#) (see page 173) for the changes to take effect:

```
cumulus@switch:~$ sudo systemctl restart switchd.service
```

Configuration Files

The following configuration applies to 10G, 40G, and 100G switches on Tomahawk, Trident II+ or Trident II platforms only.

- `/etc/cumulus/datapath/traffic.conf`: The datapath configuration file.

[Click to view sample traffic.conf file ...](#)

```
cumulus@switch:~$ cat /etc/cumulus/datapath/traffic.conf
#
# /etc/cumulus/datapath/traffic.conf
#
# packet header field used to determine the packet priority
level
# fields include {802.1p,
dscp}
traffic.packet_priority_source = 802.
1p
# remark packet priority
value
# fields include {802.1p,
none}
traffic.remark_packet_priority = none
# packet priority values assigned to each internal cos
value
# internal cos values {cos_0..
cos_7}
# (internal cos 3 has been reserved for CPU-generated
traffic)
# 802.1p values = {0..7}, dscp values = {0..63}
traffic.cos_0.packet_priorities = [0]
traffic.cos_1.packet_priorities = [1]
traffic.cos_2.packet_priorities = [2]
traffic.cos_3.packet_priorities = []
traffic.cos_4.packet_priorities = [3,4]
```

```

traffic.cos_5.packet_priorities = [5]
traffic.cos_6.packet_priorities = [6]
traffic.cos_7.packet_priorities = [7]
# priority groups
traffic.priority_group_list = [control, service, bulk]
# internal cos values assigned to each priority group
# each cos value should be assigned exactly once
# internal cos values {0..7}
priority_group.control.cos_list = [7]
priority_group.service.cos_list = [2]
priority_group.bulk.cos_list = [0,1,3,4,5,6]
# to configure priority flow control on a group of ports:
# -- assign cos value(s) to the cos list
# -- add or replace a port group names in the port group list
# -- for each port group in the list
#     -- populate the port set, e.g.
#         swp1-swp4,swp8,swp50s0-swp50s3
#     -- set a PFC buffer size in bytes for each port in the group
#     -- set the xoff byte limit (buffer limit that triggers PFC frame
transmit to start)
#     -- set the xon byte delta (buffer limit that triggers PFC frame
transmit to stop)
#     -- enable PFC frame transmit and/or PFC frame receive
# priority flow control
# pfc.port_group_list = [pfc_port_group]
# pfc.pfc_port_group.cos_list = []
# pfc.pfc_port_group.port_set = swp1-swp4,swp6
# pfc.pfc_port_group.port_buffer_bytes = 25000
# pfc.pfc_port_group.xoff_size = 10000
# pfc.pfc_port_group.xon_delta = 2000
# pfc.pfc_port_group.tx_enable = true
# pfc.pfc_port_group.rx_enable = true
# to configure pause on a group of ports:
# -- add or replace port group names in the port group list
# -- for each port group in the list
#     -- populate the port set, e.g.
#         swp1-swp4,swp8,swp50s0-swp50s3
#     -- set a pause buffer size in bytes for each port in the group
#     -- set the xoff byte limit (buffer limit that triggers pause
frames transmit to start)
#     -- set the xon byte delta (buffer limit that triggers pause
frames transmit to stop)
# link pause
# link_pause.port_group_list = [pause_port_group]
# link_pause.pause_port_group.port_set = swp1-swp4,swp6
# link_pause.pause_port_group.port_buffer_bytes = 25000
# link_pause.pause_port_group.xoff_size = 10000
# link_pause.pause_port_group.xon_delta = 2000
# link_pause.pause_port_group.rx_enable = true
# link_pause.pause_port_group.tx_enable = true
# scheduling algorithm: algorithm values = {dwrr}
scheduling.algorithm = dwrr

```

```
# traffic group scheduling weight
# weight values = {0..127}
# '0' indicates strict priority
priority_group.control.weight = 0
priority_group.service.weight = 32
priority_group.bulk.weight = 16
# To turn on/off Denial of service (DOS) prevention checks
dos_enable = false
# Cut-through is disabled by default on all chips with the exception
of
# Spectrum. On Spectrum cut-through cannot be disabled.
#cut_through_enable = false
# Enable resilient hashing
#resilient_hash_enable = FALSE
# Resilient hashing flowset entries per ECMP group
# Valid values - 64, 128, 256, 512, 1024
#resilient_hash_entries_ecmp = 128
# Enable symmetric hashing
#symmetric_hash_enable = TRUE
# Set sflow/sample ingress cpu packet rate and burst in packets/sec
# Values: {0..16384}
#sflow.rate = 16384
#sflow.burst = 16384
#Specify the maximum number of paths per route entry.
# Maximum paths supported is 200.
# Default value 0 takes the number of physical ports as the max path
size.
#ecmp_max_paths = 0
#Specify the hash seed for Equal cost multipath entries
# Default value 0
# Value Rang: {0..4294967295}
#ecmp_hash_seed = 42
# Specify the forwarding table resource allocation profile, applicable
# only on platforms that support universal forwarding resources.
#
# /usr/cumulus/sbin/cl-resource-query reports the allocated table sizes
# based on the profile setting.
#
# Values: one of {'default', 'l2-heavy', 'v4-lpm-heavy', 'v6-lpm-
heavy'}
# Default value: 'default'
# Note: some devices may support more modes, please consult user
# guide for more details
#
#forwarding_table.profile = default
```

Configuring Traffic Marking through ACL Rules

You can mark traffic for egress packets through `iptables` or `ip6tables` rule classifications. To enable these rules, you do one of the following:



- Mark DSCP values in egress packets.
- Mark 802.1p CoS values in egress packets.

To enable traffic marking, use `c1-acltool`. Add the `-p` option to specify the location of the policy file. By default, if you don't include the `-p` option, `c1-acltool` looks for the policy file in `/etc/cumulus/acl/policy.d/`.

The iptables-/ip6tables-based marking is supported via the following action extension:

```
-j SETQOS --set-dscp 10 --set-cos 5
```

You can specify one of the following targets for SETQOS:

Option	Description
<code>--set-cos INT</code>	Sets the datapath resource/queuing class value. Values are defined in IEEE_P802.1p .
<code>--set-dscp value</code>	Sets the DSCP field in packet header to a value, which can be either a decimal or hex value.
<code>--set-dscp-class class</code>	Sets the DSCP field in the packet header to the value represented by the DiffServ class value. This class can be EF, BE or any of the CSxx or AFxx classes.



You can specify either `--set-dscp` or `--set-dscp-class`, but not both.

Here are two example rules:

```
[iptables]
-t mangle -A FORWARD --in-interface swp+ -p tcp --dport bgp -j SETQOS
--set-dscp 10 --set-cos 5

[ip6tables]
-t mangle -A FORWARD --in-interface swp+ -j SETQOS --set-dscp 10
```

You can put the rule in either the *mangle* table or the default *filter* table; the mangle table and filter table are put into separate TCAM slices in the hardware.

To put the rule in the mangle table, include `-t mangle`; to put the rule in the filter table, omit `-t mangle`.

Configuring Priority Flow Control

Priority flow control, as defined in the [IEEE 802.1Qbb standard](#), provides a link-level flow control mechanism that can be controlled independently for each Class of Service (CoS) with the intention to ensure no data frames are lost when congestion occurs in a bridged network.



PFC is a layer 2 mechanism that prevents congestion by throttling packet transmission. When PFC is enabled for received packets on a set of switch ports, the switch detects congestion in the ingress buffer of the receiving port and signals the upstream switch to stop sending traffic. If the upstream switch has PFC enabled for packet transmission on the designated priorities, it responds to the downstream switch and stops sending those packets for a period of time.

PFC operates between two adjacent neighbor switches; it does not provide end-to-end flow control. However, when an upstream neighbor throttles packet transmission, it could build up packet congestion and propagate PFC frames further upstream: eventually the sending server could receive PFC frames and stop sending traffic for a time.

The PFC mechanism can be enabled for individual switch priorities on specific switch ports for RX and/or TX traffic. The switch port's ingress buffer occupancy is used to measure congestion. If congestion is present, the switch transmits flow control frames to the upstream switch. Packets with priority values that do not have PFC configured are not counted during congestion detection; neither do they get throttled by the upstream switch when it receives flow control frames.

PFC congestion detection is implemented on the switch using xoff and xon threshold values for the specific ingress buffer which is used by the targeted switch priorities. When a packet enters the buffer and the buffer occupancy is above the xoff threshold, the switch transmits an Ethernet PFC frame to the upstream switch to signal packet transmission should stop. When the buffer occupancy drops below the xon threshold, the switch sends another PFC frame upstream to signal that packet transmission can resume. (PFC frames contain a quanta value to indicate a timeout value for the upstream switch: packet transmission can resume after the timer has expired, or when a PFC frame with quanta == 0 is received from the downstream switch.)

After the downstream switch has sent a PFC frame upstream, it continues to receive packets until the upstream switch receives and responds to the PFC frame. The downstream ingress buffer must be large enough to store those additional packets after the xoff threshold has been reached.



Before Cumulus Linux 3.1.1, PFC was designated as a *lossless* priority group. The lossless priority group has been removed from Cumulus Linux.

Priority flow control is fully supported on both [Broadcom](#) and [Mellanox switches](#).

PFC is disabled by default in Cumulus Linux. Enabling priority flow control (PFC) requires configuring the following settings in `/etc/cumulus/datapath/traffic.conf` on the switch:

- Specifying the name of the port group in `pfc.port_group_list` in brackets; for example, `pfc.port_group_list = [pfc_port_group]`.
- Assigning a CoS value to the port group in `pfc.pfc_port_group.cos_list` setting. Note that `pfc_port_group` is the name of a port group you specified above and is used throughout the following settings.
- Populating the port group with its member ports in `pfc.pfc_port_group.port_set`.
- Setting a PFC buffer size in `pfc.pfc_port_group.port_buffer_bytes`. This is the maximum number of bytes allocated for storing bursts of packets, guaranteed at the ingress port. The default is 25000 bytes.
- Setting the xoff byte limit in `pfc.pfc_port_group.xoff_size`. This is a threshold for the PFC buffer; when this limit is reached, an xoff transition is initiated, signaling the upstream port to stop sending traffic, during which time packets continue to arrive due to the latency of the communication. The default is 10000 bytes.

- Setting the xon delta limit in `pfc.pfc_port_group.xon_delta`. This is the number of bytes to subtract from the xoff limit, which results in a second threshold at which the egress port resumes sending traffic. After the xoff limit is reached and the upstream port stops sending traffic, the buffer begins to drain. When the buffer reaches 8000 bytes (assuming default xoff and xon settings), the egress port signals that it can start receiving traffic again. The default is 2000 bytes.
- Enabling the egress port to signal the upstream port to stop sending traffic (`pfc.pfc_port_group.tx_enable`). The default is `true`.
- Enabling the egress port to receive notifications and act on them (`pfc.pfc_port_group.rx_enable`). The default is `true`.
- The switch priority value(s) are mapped to the specific ingress buffer for each targeted switch port. Cumulus Linux looks at either the 802.1p bits or the IP layer DSCP bits depending on which is configured in the `traffic.conf` file to map packets to internal switch priority values.

The following configuration example shows PFC configured for ports swp1 through swp4 and swp6:

```
# to configure priority flow control on a group of ports:
# -- assign cos value(s) to the cos list
# -- add or replace a port group names in the port group list
# -- for each port group in the list
#     -- populate the port set, e.g.
#         swp1-swp4,swp8,swp50s0-swp50s3
#     -- set a PFC buffer size in bytes for each port in the group
#     -- set the xoff byte limit (buffer limit that triggers PFC frame
transmit to start)
#     -- set the xon byte delta (buffer limit that triggers PFC frame
transmit to stop)
#     -- enable PFC frame transmit and/or PFC frame receive
# priority flow control
pfc.port_group_list = [pfc_port_group]
pfc.pfc_port_group.cos_list = []
pfc.pfc_port_group.port_set = swp1-swp4,swp6
pfc.pfc_port_group.port_buffer_bytes = 25000
pfc.pfc_port_group.xoff_size = 10000
pfc.pfc_port_group.xon_delta = 2000
pfc.pfc_port_group.tx_enable = true
pfc.pfc_port_group.rx_enable = true
```

Understanding Port Groups

A *port group* refers to one or more sequences of contiguous ports. Multiple port groups can be defined by:

- Adding a comma-separated list of port group names to the `port_group_list`.
- Adding the `port_set`, `rx_enable`, and `tx_enable` configuration lines for each port group.

You can specify the set of ports in a port group in comma-separated sequences of contiguous ports; you can see which ports are contiguous in `/var/lib/cumulus/porttab`. The syntax supports:

- A single port (swp1s0 or swp5)
- A sequence of regular swp ports (swp2-swp5)



- A sequence within a breakout swp port (swp6s0-swp6s3)
- A sequence of regular and breakout ports, provided they are all in a contiguous range. For example:

```
...  
swp2  
swp3  
swp4  
swp5  
swp6s0  
swp6s1  
swp6s2  
swp6s3  
swp7  
...
```

Restart `switchd` (see page 173) to allow the PFC configuration changes to take effect:

```
cumulus@switch:~$ sudo systemctl restart switchd.service
```

Configuring Link Pause

The PAUSE frame is a flow control mechanism that halts the transmission of the transmitter for a specified period of time. A server or other network node within the data center may be receiving traffic faster than it can handle it, thus the PAUSE frame. In Cumulus Linux, individual ports can be configured to execute link pause by:

- Transmitting pause frames when its ingress buffers become congested (TX pause enable) and/or
- Responding to received pause frames (RX pause enable).

Link pause is disabled by default. Enabling link pause requires configuring settings in `/etc/cumulus/datapath/traffic.conf`, similar to how you configure [priority flow control \(see page 218\)](#). The settings are explained in that section as well.

Here is an example configuration which turns off both types of link pause for swp1 through swp4 and swp6:

```
# to configure pause on a group of ports:  
# -- add or replace port group names in the port group list  
# -- for each port group in the list  
#     -- populate the port set, e.g.  
#         swp1-swp4,swp8,swp50s0-swp50s3  
#     -- set a pause buffer size in bytes for each port in the group  
#     -- set the xoff byte limit (buffer limit that triggers pause  
frames transmit to start)  
#     -- set the xon byte delta (buffer limit that triggers pause  
frames transmit to stop)  
  
# link pause  
link_pause.port_group_list = [pause_port_group]
```

```
link_pause.pause_port_group.port_set = swp1-swp4,swp6
link_pause.pause_port_group.port_buffer_bytes = 25000
link_pause.pause_port_group.xoff_size = 10000
link_pause.pause_port_group.xon_delta = 2000
link_pause.pause_port_group.rx_enable = true
link_pause.pause_port_group.tx_enable = true
```

Restart `switchd` (see page 173) to allow link pause configuration changes to take effect:

```
cumulus@switch:~$ sudo systemctl restart switchd.service
```

Configuring Explicit Congestion Notification

Explicit Congestion Notification (ECN) is defined by [RFC 3168](#). ECN gives a Cumulus Linux switch the ability to mark a packet to signal impending congestion instead of dropping the packet outright, which is how TCP typically behaves when ECN is not enabled.

ECN is a layer 3 end-to-end congestion notification mechanism only. Packets can be marked as *ECN-capable transport* (ECT) by the sending server. If congestion is observed by any switch while the packet is getting forwarded, the ECT-enabled packet can be marked by the switch to indicate the congestion. The end receiver can respond to the ECN-marked packets by signaling the sending server to slow down transmission. The sending server marks a packet *ECT* by setting the least 2 significant bits in an IP header `DiffServ` (ToS) field to `01` or `10`. A packet that has the least 2 significant bits set to `00` indicates a non-ECT-enabled packet.

The ECN mechanism on a switch only marks packets to notify the end receiver. It does not take any other action or change packet handling in any way, nor does it respond to packets that have already been marked ECN by an upstream switch.



On Trident-II switches only, if ECN is enabled on a specific queue, the ASIC also enables WRED on the same queue. If the packet is ECT marked (the ECN bits are `01` or `10`), the ECN mechanism executes as described above. However, if it is entering an ECN-enabled queue but is not ECT marked (the ECN bits are `00`), then the WRED mechanism uses the same threshold and probability values to decide whether to drop the packet. Packets entering a non-ECN-enabled queue do not get marked or dropped due to ECN or WRED in any case.

ECN is implemented on the switch using minimum and maximum threshold values for the egress queue length. When a packet enters the queue and the average queue length is between the minimum and maximum threshold values, a configurable probability value will determine whether the packet will be marked. If the average queue length is above the maximum threshold value, the packet is always marked.

The downstream switches with ECN enabled perform the same actions as the traffic is received. If the ECN bits are set, they remain set. The only way to overwrite ECN bits is to enable it — that is, set the ECN bits to `11`.

ECN is supported on [Broadcom Tomahawk](#), [Trident II+](#) and [Trident II](#), and [Mellanox Spectrum](#) switches only.

Configuring ECN

Click to learn how to configure ECN ...



ECN is disabled by default in Cumulus Linux. You can enable ECN for individual switch priorities on specific switch ports. ECN requires configuring the following settings in `/etc/cumulus/datapath/traffic.conf` on the switch:

- Specifying the name of the port group in `ecn.port_group_list` in brackets; for example, `ecn.port_group_list = [ecn_port_group]`.
- Assigning a CoS value to the port group in `ecn.ecn_port_group.cos_list`. Note that `ecn_port_group` is the name of a port group you specified above.
- Populating the port group with its member ports (`ecn.ecn_port_group.port_set`), where `ecn_port_group` is the name of the port group you specified above. Congestion is measured on the egress port queue for the ports listed here, using the average queue length: if congestion is present, a packet entering the queue may be marked to indicate that congestion was observed. Marking a packet involves setting the least 2 significant bits in the IP header DiffServ (ToS) field to 11.
- The switch priority value(s) are mapped to specific egress queues for the target switch ports.
- The `ecn.ecn_port_group.probability` value indicates the probability of a packet being marked if congestion is experienced.

The following configuration example shows ECN configured for ports swp1 through swp4 and swp6:

```
# Explicit Congestion Notification
# to configure ECN on a group of ports:
# -- add or replace port group names in the port group list
# -- assign cos value(s) to the cos list
# -- for each port group in the list
#     -- populate the port set, e.g.
#         swp1-swp4,swp8,swp50s0-swp50s3
ecn.port_group_list = [ecn_port_group]
ecn.ecn_port_group.cos_list = [3]
ecn.ecn_port_group.port_set = swp1-swp4,swp6
ecn.ecn_port_group.min_threshold_bytes = 40000
ecn.ecn_port_group.max_threshold_bytes = 200000
ecn.ecn_port_group.probability = 100
```

Restart `switchd` (see page 173) to allow the ECN configuration changes to take effect:

```
cumulus@switch:~$ sudo systemctl restart switchd.service
```

Caveats and Errata

- You can configure Quality of Service (QoS) for 10G, 40G, and 100G switches on the Broadcom Tomahawk, Trident II+ or Trident II platforms and Mellanox Spectrum platform only.

Related Information

- [iptables-extensions man page](#)



Configuring Hardware-enabled DDOS Protection

The DDOS protection mechanism protects data plane, control plane and management plane traffic in the switch. It drops any packets that match one or more of the following criteria while incurring no performance impact:

- Source IP address matches the destination address for IPv4 and IPv6 packets
- Source MAC address matches the destination MAC address
- Unfragmented or first fragment SYN packets with a source port of 0-1023
- TCP packets with control flags = 0 and seq number == 0
- TCP packets with FIN, URG and PSH bits set and seq number == 0
- TCP packets with both SYN and FIN bits set
- TCP source PORT matches the destination PORT
- UDP source PORT matches the destination PORT
- First TCP fragment with partial TCP header
- TCP header has fragment offset value of 1
- ICMPv6 ping packets payload larger than programmed value of ICMP max size
- ICMPv4 ping packets payload larger than programmed value of ICMP max size
- Fragmented ICMP packet
- IPv6 fragment lower than programmed minimum IPv6 packet size



This configuration option is only available for Broadcom Trident, Trident II, and Tomahawk chipsets.

Cumulus Networks recommends enabling this feature when deploying a switch with the above mentioned ASICs, as hardware-based DDOS protection is disabled by default. Although Cumulus recommends enabling all of the above criteria, they can be individually enabled if desired.

Configure Persistent DDOS Protection

1. Open the `/etc/cumulus/datapath/traffic.conf` file in a text editor.
2. Enable DOS prevention checks by changing the following value to `true`, and save the file:

```
# To turn on/off Denial of Service (DOS) prevention checks  
dos_enable = true
```

3. Open the `/usr/lib/python2.7/dist-packages/cumulus/__chip_config/bcm/datapath.conf` file (on a Broadcom-based switch) or `/usr/lib/python2.7/dist-packages/cumulus/__chip_config/datapath.conf` file (on a Mellanox-based switch) in a text editor.
4. Set the following checks to `true`, and save the file:

```
# Enabling/disabling Denial of service (DOS) prevention checks
# To change the default configuration:
# enable/disable the individual DOS checks.
dos.sip_eq_dip = true
dos.smac_eq_dmac = true
dos.tcp_hdr_partial = true
dos.tcp_syn_frag = true
dos.tcp_ports_eq = true
dos.tcp_flags_syn_fin = true
dos.tcp_flags_fup_seq0 = true
dos.tcp_offset1 = true
dos.tcp_ctrl0_seq0 = true
dos.udp_ports_eq = true
dos.icmp_frag = true
dos.icmpv4_length = true
dos.icmpv6_length = true
dos.ipv6_min_frag = true
```

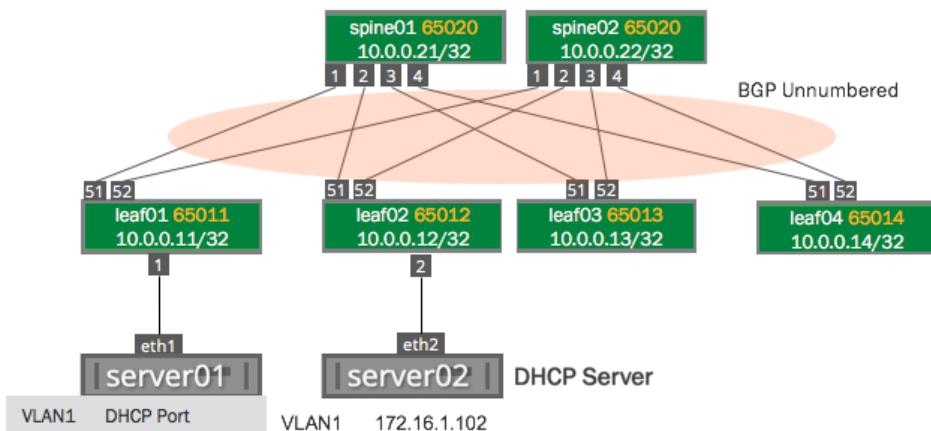
5. Restart switchd to enable DOS protection:

```
cumulus@switch:~$ sudo systemctl restart switchd.service
```

Configuring DHCP Relays

You can configure DHCP relays for IPv4 and IPv6.

To run DHCP for both IPv4 and IPv6, initiate the DHCP relay once for IPv4 and once for IPv6. Following are the configurations on the server hosts, DHCP relay and DHCP server using the following topology:



The `dhcpd` and `dhcrelay` services are disabled by default. After you finish configuring the DHCP relays and servers, you need to start those services.



Contents

This chapter covers ...

- Configuring IPv4 DHCP Relays (see page 230)
- Configuring IPv6 DHCP Relays (see page 231)
- Configuring the DHCP Relay Service Manually (Advanced) (see page 232)
- Troubleshooting the DHCP Relays (see page 232)
 - Looking at the Log on Switch where DHCP Relay Is Configured (see page 232)

Configuring IPv4 DHCP Relays

Configure `isc-dhcp-relay` using [NCLU](#) (see page 80), specifying the IP addresses to each DHCP server and the interfaces that are used as the uplinks.

In the examples below, the DHCP server IP address is 172.16.1.102, VLAN 1 (the SVI is `vlan1`) and the uplinks are `sdp51` and `sdp52`.



You configure a DHCP relay on a per-VLAN basis, specifying the SVI, not the parent bridge — in our example, you would specify `vlan1` as the SVI for VLAN 1; do not specify the bridge named `bridge` in this case.

As per [RFC 3046](#), you can specify as many server IP addresses that can fit in 255 octets, specifying each address only once.

```
cumulus@leaf01:~$ net add dhcp relay interface sdp51
cumulus@leaf01:~$ net add dhcp relay interface sdp52
cumulus@leaf01:~$ net add dhcp relay interface vlan1
cumulus@leaf01:~$ net add dhcp relay server 172.16.1.102
cumulus@leaf01:~$ net pending
cumulus@leaf01:~$ net commit
```

These commands create the following configuration in the `/etc/default/isc-dhcp-relay` file:

```
cumulus@leaf01:~$ cat /etc/default/isc-dhcp-relay
SERVERS="172.16.1.102"
INTF_CMD="-i vlan1 -i sdp51 -i sdp52"
OPTIONS=""
```

After you've finished configuring the DHCP relay, restart then enable the `dhcrelay` service so the configuration persists between reboots:

```
cumulus@leaf01:~$ sudo systemctl restart dhcrelay.service
cumulus@leaf01:~$ sudo systemctl enable dhcrelay.service
```

To see the status of the DHCP relay, use the `systemctl status dhcrelay.service` command:



```
cumulus@leaf01:~$ sudo systemctl status dhcrelay.service
dhcrelay.service - DHCPv4 Relay Agent Daemon
   Loaded: loaded (/lib/systemd/system/dhcrelay.service; enabled)
   Active: active (running) since Fri 2016-12-02 17:09:10 UTC; 2min
          16s ago
     Docs: man:dhcrelay(8)
 Main PID: 1997 (dhcrelay)
    CGroup: /system.slice/dhcrelay.service
              1997 /usr/sbin/dhcrelay --nl -d -q -i vlan1 -i swp51 -i
              swp52 172.16.1.102
```

Configuring IPv6 DHCP Relays

If you're configuring IPv6, the `/etc/default/isc-dhcp-relay6` variables file has a different format than the `/etc/default/isc-dhcp-relay` file for IPv4 DHCP relays. Make sure to configure the variables appropriately by editing this file.



You cannot use NCLU to configure IPv6 relays.

```
cumulus@leaf01:$ sudo nano /etc/default/isc-dhcp-relay6
SERVERS=" -u 2001:db8:100::2%swp51 -u 2001:db8:100::2%swp52"
INTF_CMD="-l vlan1"
```

After you've finished configuring the DHCP relay, save your changes, restart the `dhcrelay6` service, then enable the `dhcrelay6` service so the configuration persists between reboots:

```
cumulus@leaf01:~$ sudo systemctl restart dhcrelay6.service
cumulus@leaf01:~$ sudo systemctl enable dhcrelay6.service
```

To see the status of the IPv6 DHCP relay, use the `systemctl status dhcrelay6.service` command:

```
cumulus@leaf01:~$ sudo systemctl status dhcrelay6.service
dhcrelay6.service - DHCPv6 Relay Agent Daemon
   Loaded: loaded (/lib/systemd/system/dhcrelay6.service; disabled)
   Active: active (running) since Fri 2016-12-02 21:00:26 UTC; 1s ago
     Docs: man:dhcrelay(8)
 Main PID: 6152 (dhcrelay)
    CGroup: /system.slice/dhcrelay6.service
              6152 /usr/sbin/dhcrelay -6 --nl -d -q -l vlan1 -u 2001:db8:
              100::2 swp51 -u 2001:db8:100::2 swp52
```



Configuring the DHCP Relay Service Manually (Advanced)

Configuring the DHCP service manually ...

By default, Cumulus Linux configures the DHCP relay service automatically. However, in older versions of Cumulus Linux, you needed to edit the `dhcrelay.service` file as described below. The IPv4 `dhcrelay.service` *Unit* script calls `/etc/default/isc-dhcp-relay` to find launch variables.

```
cumulus@switch:~$ cat /lib/systemd/system/dhcrelay.service
[Unit]
Description=DHCPv4 Relay Agent Daemon
Documentation=man:dhcrelay(8)
After=network-online.target networking.service syslog.service
[Service]
Type=simple
EnvironmentFile=/etc/default/isc-dhcp-relay
# Here, we are expecting the INTF_CMD to contain
# the -i for each interface specified,
# e.g. "-i eth0 -i swp1"
ExecStart=/usr/sbin/dhcrelay -d -q $INTF_CMD $SERVERS $OPTIONS
[Install]
WantedBy=multi-user.target
```

The `/etc/default/isc-dhcp-relay` variables file needs to reference both interfaces participating in DHCP relay (facing the server and facing the client) and the IP address of the server. If the client-facing interface is a bridge port, specify the switch virtual interface (SVI) name if using a [VLAN-aware bridge \(see page 277\)](#) (for example, `vlan100`), or the bridge name if using traditional bridging (for example, `br100`).

Troubleshooting the DHCP Relays

If you are experiencing issues with the DHCP relay, you can run the following commands to determine whether or not the issue is with `systemd`. The following commands manually activate the DHCP relay process, and they do not persist when you reboot the switch:

```
cumulus@switch:~$ /usr/sbin/dhcrelay -4 -i <interface_facing_host>
<ip_address_dhcp_server> -i <interface_facing_dhcp_server>
cumulus@switch:~$ /usr/sbin/dhcrelay -6 -l <interface_facing_host> -u
<ip_address_dhcp_server>%<interface_facing_dhcp_server>
```

For example:

```
cumulus@leaf01:~$ /usr/sbin/dhcrelay -4 -i vlan1 172.16.1.102 -i swp51
cumulus@leaf01:~$ /usr/sbin/dhcrelay -6 -l vlan1 -u 2001:db8:100::2%
swp51
```

See `man dhcrelay` for more information.



Looking at the Log on Switch where DHCP Relay Is Configured

Use the `journalctl` command to look at the behavior on the Cumulus Linux switch that is providing the DHCP relay functionality:

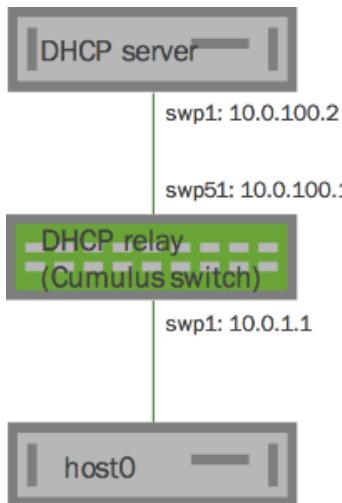
```
cumulus@leaf01:~$ sudo journalctl -l -n 20 | grep dhcrelay
Dec 05 20:58:55 leaf01 dhcrelay[6152]: sending upstream swp52
Dec 05 20:58:55 leaf01 dhcrelay[6152]: sending upstream swp51
Dec 05 20:58:55 leaf01 dhcrelay[6152]: Relaying Reply to fe80::4638:
39ff:fe00:3 port 546 down.
Dec 05 20:58:55 leaf01 dhcrelay[6152]: Relaying Reply to fe80::4638:
39ff:fe00:3 port 546 down.
Dec 05 21:03:55 leaf01 dhcrelay[6152]: Relaying Renew from fe80::4638:
39ff:fe00:3 port 546 going up.
Dec 05 21:03:55 leaf01 dhcrelay[6152]: sending upstream swp52
Dec 05 21:03:55 leaf01 dhcrelay[6152]: sending upstream swp51
Dec 05 21:03:55 leaf01 dhcrelay[6152]: Relaying Reply to fe80::4638:
39ff:fe00:3 port 546 down.
Dec 05 21:03:55 leaf01 dhcrelay[6152]: Relaying Reply to fe80::4638:
39ff:fe00:3 port 546 down.
```

You can run the command `journalctl` command with the `--since` flag to specify a time period:

```
cumulus@leaf01:~$ sudo journalctl -l --since "2 minutes ago" | grep
dhcrelay
Dec 05 21:08:55 leaf01 dhcrelay[6152]: Relaying Renew from fe80::4638:
39ff:fe00:3 port 546 going up.
Dec 05 21:08:55 leaf01 dhcrelay[6152]: sending upstream swp52
Dec 05 21:08:55 leaf01 dhcrelay[6152]: sending upstream swp51
```

Configuring DHCP Servers

To run DHCP for both IPv4 and IPv6, you need to initiate the DHCP server twice: once for IPv4 and once for IPv6. The following configuration uses the following topology for the host, DHCP relay and DHCP server:



For the configurations used in this chapter, the DHCP server is a switch running Cumulus Linux; however, the DHCP server can also be located on a dedicated server in your environment.



The `dhcpd` and `dhcrelay` services are disabled by default. After you finish configuring the DHCP relays and servers, you need to start those services.

Contents

This chapter covers ...

- Configuring DHCP Server on Cumulus Linux Switches (see page 234)
 - Configuring the IPv4 DHCP Server (see page 234)
 - Configuring the IPv6 DHCP Server (see page 235)
- Troubleshooting the Log from a DHCP Server (see page 236)

Configuring DHCP Server on Cumulus Linux Switches

You can use the following sample configurations for `dhcp.conf` and `dhcpd6.conf` to start both an IPv4 and an IPv6 DHCP server. The configuration files for the two DHCP server instances need to have two pools:

- Pool 1: Subnet overlaps interfaces
- Pool 2: Subnet that includes the addresses

Configuring the IPv4 DHCP Server

In a text editor, edit the `dhcpd.conf` file with a configuration similar to the following:

```

cumulus@switch:~$ cat /etc/dhcp/dhcpd.conf
ddns-update-style none;

default-lease-time 600;
max-lease-time 7200;
  
```



```
subnet 10.0.100.0 netmask 255.255.255.0 {  
}  
subnet 10.0.1.0 netmask 255.255.255.0 {  
    range 10.0.1.50 10.0.1.60;  
}
```

Just as you did with the DHCP relay scripts, edit the DHCP server configuration file so it can launch the DHCP server when the system boots. Here is a sample configuration:

```
cumulus@switch:~$ cat /etc/default/isc-dhcp-server  
DHCPD_CONF="-cf /etc/dhcp/dhcpd.conf"  
  
INTERFACES="swp1"
```

After you've finished configuring the DHCP server, enable the `dhcpd` service immediately:

```
cumulus@switch:~$ sudo systemctl enable dhcpd.service
```

Configuring the IPv6 DHCP Server

In a text editor, edit the `dhcpd6.conf` file with a configuration similar to the following:

```
cumulus@switch:~$ cat /etc/dhcp/dhcpd6.conf  
ddns-update-style none;  
  
default-lease-time 600;  
max-lease-time 7200;  
  
subnet6 2001:db8:100::/64 {  
}  
subnet6 2001:db8:1::/64 {  
    range6 2001:db8:1::100 2001:db8:1::200;  
}
```

Just as you did with the DHCP relay scripts, edit the DHCP server configuration file so it can launch the DHCP server when the system boots. Here is a sample configuration:

```
cumulus@switch:~$ cat /etc/default/isc-dhcp-server6  
DHCPD_CONF="-cf /etc/dhcp/dhcpd6.conf"  
  
INTERFACES="swp1"
```



You cannot use NCLU to configure IPv6 DHCP servers.



After you've finished configuring the DHCP server, enable the `dhcpd6` service immediately:

```
cumulus@switch:~$ sudo systemctl enable dhcpd6.service
```

Troubleshooting the Log from a DHCP Server

The DHCP server knows whether a DHCP request is a relay or a non-relay DHCP request. On isc-dhcp-server, for example, it is possible to tail the log and look at the behavior firsthand:

```
cumulus@server02:~$ sudo tail /var/log/syslog | grep dhcpcd
2016-12-05T19:03:35.379633+00:00 server02 dhcpcd: Relay-forward
message from 2001:db8:101::1 port 547, link address 2001:db8:101::1,
peer address fe80::4638:39ff:fe00:3
2016-12-05T19:03:35.380081+00:00 server02 dhcpcd: Advertise NA:
address 2001:db8:1::110 to client with duid 00:01:00:01:1f:d8:75:3a:
44:38:39:00:00:03 iaid = 956301315 valid for 600 seconds
2016-12-05T19:03:35.380470+00:00 server02 dhcpcd: Sending Relay-reply
to 2001:db8:101::1 port 547
```



Layer One and Two

Spanning Tree and Rapid Spanning Tree

Spanning tree protocol (STP) is always recommended in layer 2 topologies, as it prevents bridge loops and broadcast radiation on a bridged network. STP also provides redundant links for automatic failover when an active link fails. STP is disabled by default on bridges in Cumulus Linux.

Contents

This chapter covers ...

- Supported Modes (see page 237)
 - STP for a VLAN-aware Bridge (see page 238)
 - STP within a Traditional Mode Bridge (see page 238)
- Viewing Bridge and STP Status/Logs (see page 238)
 - Using Linux to Check Spanning Tree Status (Advanced) (see page 241)
- Customizing Spanning Tree Protocol (see page 242)
 - Spanning Tree Priority (see page 242)
 - PortAdminEdge/PortFast Mode (see page 243)
 - PortAutoEdge (see page 244)
 - BPDU Guard (see page 244)
 - Bridge Assurance (see page 247)
 - BPDU Filter (see page 247)
 - Storm Control (see page 248)
 - Configuring Other Spanning Tree Parameters (see page 248)
- Caveats and Errata (see page 251)
- Related Information (see page 251)

Supported Modes

The STP modes Cumulus Linux supports vary depending upon whether the [traditional or VLAN-aware bridge driver mode](#) (see page 272) is in use.

Bridges configured in [VLAN-aware](#) (see page 277) mode operate **only** in RSTP mode. [NCLU](#) (see page 80), the network command line utility for configuring Cumulus Linux, only supports bridges in VLAN-aware mode.

For a bridge configured in *traditional* mode, PVST and PVRST are supported, with the default set to PVRST. Each traditional bridge has its own separate STP instance.

Since you cannot use NCLU to configure a traditional mode bridge, you must configure it directly in the `/etc/network/interfaces` file.

STP for a VLAN-aware Bridge

VLAN-aware (see page 277) bridges only operate in RSTP mode. STP bridge protocol data units (BPDUs) are transmitted on the native VLAN.

If a bridge running RSTP (802.1w) receives a common STP (802.1D) BPDU, it will automatically fall back to 802.1D operation. RSTP interoperates with MST seamlessly, creating a single instance of spanning tree, which transmits BPDUs on the native VLAN. RSTP treats the MST domain as if it were one giant switch.



As of version 3.2.1, STP is enabled by default in Cumulus Linux. There is no need to specify `bridge-stp on` for the bridge any more.

STP within a Traditional Mode Bridge

Per VLAN Spanning Tree (PVST) creates a spanning tree instance for a bridge. Rapid PVST (PVRST) supports RSTP enhancements for each spanning tree instance. In order to use PVRST with a traditional bridge, a bridge corresponding to the untagged native VLAN must be created, and all the physical switch ports must be part of the same VLAN.



When connected to a switch that has a native VLAN configuration, the native VLAN **must** be configured to be VLAN 1 only for maximum interoperability.

Viewing Bridge and STP Status/Logs

To check STP status for a bridge, run `net show bridge spanning-tree`:

Click to reveal the output ...

```
cumulus@switch:~$ net show bridge spanning-tree
bridge CIST info
  enabled      yes
  bridge id    1.000.44:39:39:FF:40:90
  designated root 1.000.44:39:39:FF:40:90
  regional root 1.000.44:39:39:FF:40:90
  root port    none
  path cost    0          internal path cost  0
  max age      20         bridge max age     20
  forward delay 15        bridge forward delay 15
  tx hold count 6          max hops       20
  hello time   2          ageing time    300
  force protocol version  rstp
  time since topology change 253343s
  topology change count      4
  topology change            no
  topology change port      peerlink
  last topology change port leaf03-04
bridge:exit01-02 CIST info
```



enabled	no	role
Disabled		
port id	8.004	state
discarding		
external port cost	305	admin external cost 0
internal port cost	305	admin internal cost 0
designated root	1.000.44:38:39:00:00:27	dsgn external cost 0
dsgn regional root	1.000.44:38:39:00:00:27	dsgn internal cost 0
designated bridge	1.000.44:38:39:00:00:27	designated port
8.004		
admin edge port	no	auto edge port yes
oper edge port	no	topology change ack no
point-to-point	yes	admin point-to-point auto
restricted role	no	restricted TCN no
port hello time	2	disputed no
bpdu guard port	no	bpdu guard error no
network port	no	BA inconsistent no
Num TX BPDU	2	Num TX TCN 0
Num RX BPDU	0	Num RX TCN 0
Num Transition FWD	0	Num Transition BLK 2
bpdufilter port	no	
clag ISL	no	clag ISL Oper UP no
clag role	primary	clag dual conn mac 00:
00:00:00:00:00:00		
clag remote portID	F.FFFF	clag system mac 44:
39:39:FF:40:90		
bridge:leaf01-02 CIST info		
enabled	yes	role
Designated		
port id	8.003	state
forwarding		
external port cost	10000	admin external cost 0
internal port cost	10000	admin internal cost 0
designated root	1.000.44:39:39:FF:40:90	dsgn external cost 0
dsgn regional root	1.000.44:39:39:FF:40:90	dsgn internal cost 0
designated bridge	1.000.44:39:39:FF:40:90	designated port
8.003		
admin edge port	no	auto edge port yes
oper edge port	no	topology change ack no
point-to-point	yes	admin point-to-point auto
restricted role	no	restricted TCN no
port hello time	2	disputed no
bpdu guard port	no	bpdu guard error no
network port	no	BA inconsistent no
Num TX BPDU	253558	Num TX TCN 2
Num RX BPDU	253373	Num RX TCN 4
Num Transition FWD	126675	Num Transition BLK
126694		
bpdufilter port	no	
clag ISL	no	clag ISL Oper UP no
clag role	primary	clag dual conn mac 44:
39:39:FF:40:94		

clag remote portID F.FFF	clag system mac	44:
39:39:FF:40:90		
bridge:leaf03-04 CIST info		
enabled yes	role	
Designated		
port id 8.001	state	
forwarding		
external port cost 10000	admin external cost 0	
internal port cost 10000	admin internal cost 0	
designated root 1.000.44:39:39:FF:40:90	dsgn external cost 0	
dsgn regional root 1.000.44:39:39:FF:40:90	dsgn internal cost 0	
designated bridge 1.000.44:39:39:FF:40:90	designated port	
8.001		
admin edge port no	auto edge port yes	
oper edge port no	topology change ack no	
point-to-point yes	admin point-to-point auto	
restricted role no	restricted TCN no	
port hello time 2	disputed no	
bpdu guard port no	bpdu guard error no	
network port no	BA inconsistent no	
Num TX BPDU 130960	Num TX TCN 6	
Num RX BPDU 4	Num RX TCN 1	
Num Transition FWD 2	Num Transition BLK 1	
bpdufilter port no		
clag ISL no	clag ISL Oper UP no	
clag role primary	clag dual conn mac 44:	
39:39:FF:40:93		
clag remote portID F.FFF	clag system mac	44:
39:39:FF:40:90		
bridge:peerlink CIST info		
enabled yes	role	
Designated		
port id F.002	state	
forwarding		
external port cost 10000	admin external cost 0	
internal port cost 10000	admin internal cost 0	
designated root 1.000.44:39:39:FF:40:90	dsgn external cost 0	
dsgn regional root 1.000.44:39:39:FF:40:90	dsgn internal cost 0	
designated bridge 1.000.44:39:39:FF:40:90	designated port F.	
002		
admin edge port no	auto edge port yes	
oper edge port no	topology change ack no	
point-to-point yes	admin point-to-point auto	
restricted role no	restricted TCN no	
port hello time 2	disputed no	
bpdu guard port no	bpdu guard error no	
network port no	BA inconsistent no	
Num TX BPDU 126700	Num TX TCN 2	
Num RX BPDU 6	Num RX TCN 3	
Num Transition FWD 2	Num Transition BLK 1	
bpdufilter port no		
clag ISL yes	clag ISL Oper UP yes	

```

    clag role           primary
00:00:00:00:00
    clag remote portID F.FFFF
39:39:FF:40:90
                                clag dual conn mac  00:
                                clag system mac   44:

```

Using Linux to Check Spanning Tree Status (Advanced)

Using Linux to check STP status ...

`mstptcl` is the utility provided by the `mstpd` service to configure STP. The `mstpd` daemon is an open source project used by Cumulus Linux to implement IEEE802.1D 2004 and IEEE802.1Q 2011.

`mstpd` is started by default when the switch boots. `mstpd` logs and errors are located in `/var/log/syslog`.



`mstpd` is the preferred utility for interacting with STP on Cumulus Linux. `brctl` also provides certain methods for configuring STP; however, they are not as complete as the tools offered in `mstpd` and [output from brctl can be misleading](#) in some cases.

To get the bridge state, use:

```

cumulus@switch:~$ sudo brctl show
bridge name      bridge id      STP enabled      interfaces
bridge          8000.001401010100  yes            swp1
                           swp4
                           swp5

```

To get the `mstpd` bridge state, use:

```

cumulus@switch:~$ net show bridge spanning-tree
bridge CIST info
  enabled        yes
  bridge id      F.000.00:14:01:01:01:00
  designated root F.000.00:14:01:01:01:00
  regional root  F.000.00:14:01:01:01:00
  root port      none
  path cost      0          internal path cost  0
  max age        20         bridge max age     20
  forward delay  15         bridge forward delay 15
  tx hold count  6          max hops          20
  hello time     2          ageing time       200
  force protocol version  rstp
  time since topology change 90843s
  topology change count      4
  topology change          no
  topology change port      swp4
  last topology change port swp5

```

To get the `mstpd` bridge port state, use:

```

cumulus@switch:~$ sudo mstpctl showport bridge
E swp1 8.001 forw F.000.00:14:01:01:01:00 F.000.00:14:01:01:01:00
8.001 Desg
  swp4 8.002 forw F.000.00:14:01:01:01:00 F.000.00:14:01:01:01:00
8.002 Desg
  E swp5 8.003 forw F.000.00:14:01:01:01:00 F.000.00:14:01:01:01:00
8.003 Desg
cumulus@switch:~$ net show bridge spanning-tree
...
bridge:swp1 CIST info
  enabled          yes           role
Designated
  port id         8.001          state
forwarding
  external port cost 2000      admin external cost  0
  internal port cost 2000      admin internal cost 0
  designated root    F.000.00:14:01:01:01:00 dsgn external cost  0
  dsgn regional root F.000.00:14:01:01:01:00 dsgn internal cost 0
  designated bridge  F.000.00:14:01:01:01:00 designated port
8.001
  admin edge port   no          auto edge port      yes
  oper edge port    yes         topology change ack no
  point-to-point    yes         admin point-to-point auto
  restricted role   no          restricted TCN       no
  port hello time   2           disputed            no
  bpdu guard port   no          bpdu guard error  no
  network port      no          BA inconsistent    no
  Num TX BPDU       45772        Num TX TCN        4
  Num RX BPDU       0           Num RX TCN        0
  Num Transition FWD 2          Num Transition BLK 2

```

Customizing Spanning Tree Protocol

There are a number of ways you can customize STP in Cumulus Linux. You should exercise extreme caution with many of the settings below to prevent malfunctions in STP's loop avoidance.

Spanning Tree Priority

If you have an MSTI (multiple spanning tree instance), you can set the *tree priority* for a bridge. The bridge with the lowest priority is elected the *root bridge*. The priority must be a number between 0 and 65535 and must be a multiple of 4096; the default is 32768.



For `msti`, only 0 is supported currently.

To set the tree priority, run:

```
cumulus@switch:~$ net add bridge stp treeprio 8192
cumulus@switch:~$ net pending
cumulus@switch:~$ net commit
```

PortAdminEdge/PortFast Mode

PortAdminEdge is equivalent to the PortFast feature offered by other vendors. It enables or disables the *initial edge state* of a port in a bridge.

All ports configured with PortAdminEdge bypass the listening and learning states to move immediately to forwarding.



Using PortAdminEdge mode has the potential to cause loops if it is not accompanied by the [BPDU guard \(see page 244\)](#) feature.

While it is common for edge ports to be configured as access ports for a simple end host, this is not mandatory. In the data center, edge ports typically connect to servers, which may pass both tagged and untagged traffic.

Example VLAN-aware Bridge Configuration

To configure PortAdminEdge mode, use the `bpduguard` and `mstpctl-portadminedge` NCLU configuration commands:

```
cumulus@switch:~$ net add interface swp5 stp bpduguard
cumulus@switch:~$ net add interface swp5 stp portadminedge
cumulus@switch:~$ net pending
cumulus@switch:~$ net commit
```

The NCLU commands above create the following code snippet:

```
auto swp5
iface swp5
    mstpctl-bpduguard yes
    mstpctl-portadminedge yes
```

Example Traditional Bridge Configuration

For a bridge in [traditional mode \(see page 272\)](#), configure PortAdminEdge under the bridge stanza in `/etc/network/interfaces`:

```
auto br2
iface br2 inet static
```

```
bridge-ports swp1 swp2 swp3 swp4
mstpctl-bpduguard swp1=yes swp2=yes swp3=yes swp4=yes
mstpctl-portadminedge swp1=yes swp2=yes swp3=yes swp4=yes
```

To load the new configuration, run `ifreload -a`:

```
cumulus@switch:~$ sudo ifreload -a
```

PortAutoEdge

PortAutoEdge is an enhancement to the standard PortAdminEdge (PortFast) mode, which allows for the automatic detection of edge ports. PortAutoEdge enables and disables the *auto transition* to/from the edge state of a port in a bridge.



Edge ports and access ports are not the same thing. Edge ports transition directly to the forwarding state and skip the listening and learning stages. Upstream topology change notifications are not generated when an edge port's link changes state. Access ports only forward untagged traffic; however, there is no such restriction on edge ports, which can forward both tagged and untagged traffic.

When a BPDU is received on a port configured with `portautoedge`, the port ceases to be in the edge port state and transitions into a normal STP port.

When BPDUs are no longer received on the interface, the port becomes an edge port, and transitions through the discarding and learning states before resuming forwarding.

To configure PortAutoEdge for an interface:

```
cumulus@switch:~$ net add interface swp1 stp portautoedge
cumulus@switch:~$ net pending
cumulus@switch:~$ net commit
```

BPDU Guard

To protect the spanning tree topology from unauthorized switches affecting the forwarding path, you can configure *BPDU guard* (Bridge Protocol Data Unit). One very common example is when someone hooks up a new switch to an access port off of a leaf switch. If this new switch is configured with a low priority, it could become the new root switch and affect the forwarding path for the entire layer 2 topology.

Example BPDU Guard Configuration

To configure BPDU guard, set the `bpduguard` value for the interface:

```
cumulus@switch:~$ net add interface swp5 stp bpduguard
cumulus@switch:~$ net pending
```



```
cumulus@switch:~$ net commit
```

This creates the following stanza in the `/etc/network/interfaces` file:

```
auto swp5
iface swp5
    mstpctl-bpduguard yes
```

Recovering a Port Disabled by BPDU Guard

If a BPDU is received on the port, STP will bring down the port and log an error in `/var/log/syslog`. The following is a sample error:

```
mstpd: error, MSTP_IN_rx_bpdu: bridge:bond0 Recvd BPDU on BPDU Guard
Port - Port Down
```

To determine whether BPDU guard is configured, or if a BPDU has been received, run:

```
cumulus@switch:~$ net show bridge spanning-tree | grep bpdu
bpdu guard port      yes                                bpdu guard error      yes
```

The only way to recover a port that has been placed in the disabled state is to manually un-shut or bring up the port with `sudo ifup [port]`, as shown in the example below.



Bringing up the disabled port does not fix the problem if the configuration on the connected end-station has not been rectified.

```
cumulus@leaf2$ mstpcl showportdetail bridge bond0
bridge:bond0 CIST info
  enabled          no           role
Disabled
  port id        8.001         state
discarding
  external port cost 305           admin external cost  0
  internal port cost 305          admin internal cost  0
  designated root   8.000.6C:64:1A:00:4F:9C dsgn external cost  0
  dsgn regional root 8.000.6C:64:1A:00:4F:9C dsgn internal cost  0
  designated bridge 8.000.6C:64:1A:00:4F:9C designated port
8.001
  admin edge port    no           auto edge port      yes
  oper edge port     no           topology change ack no
  point-to-point     yes          admin point-to-point auto
```

```

restricted role      no          restricted TCN      no
port hello time    10         disputed           no
bpdu guard port    yes        bpdu guard error   yes
network port       no         BA inconsistent   no
Num TX BPDU        3          Num TX TCN       2
Num RX BPDU        488        Num RX TCN       2
Num Transition FWD 1          Num Transition BLK 2
bpdufilter port   no
clag ISL           no
clag role          unknown
0:0:0:0
  clag remote portID F.FFF
0:0:0:0

```

cumulus@leaf2\$ sudo ifup bond0

```

cumulus@leaf2$ mstpcctl showportdetail bridge bond0
bridge:bond0 CIST info
  enabled          yes        role             Root
  port id         8.001      state
forwarding
  external port cost 305      admin external cost 0
  internal port cost 305     admin internal cost 0
  designated root    8.000.6C:64:1A:00:4F:9C dsgn external cost 0
  dsgn regional root 8.000.6C:64:1A:00:4F:9C dsgn internal cost 0
  designated bridge  8.000.6C:64:1A:00:4F:9C designated port
8.001
  admin edge port    no        auto edge port    yes
  oper edge port    no        topology change ack no
  point-to-point    yes      admin point-to-point auto
  restricted role   no        restricted TCN   no
  port hello time  2          disputed           no
  bpdu guard port  no        bpdu guard error  no
  network port     no        BA inconsistent   no
  Num TX BPDU      3          Num TX TCN       2
  Num RX BPDU      43         Num RX TCN       1
  Num Transition FWD 1        Num Transition BLK 0
  bpdufilter port no
  clag ISL          no
  clag role         unknown
0:0:0:0
  clag remote portID F.FFF
0:0:0:0

```

Bridge Assurance

On a point-to-point link where RSTP is running, if you want to detect unidirectional links and put the port in a discarding state (in error), you can enable bridge assurance on the port by enabling a port type network. The port would be in a bridge assurance inconsistent state until a BPDU is received from the peer. You need to configure the port type network on both the ends of the link in order for bridge assurance to operate properly.

The default setting for bridge assurance is off. This means that there is no difference between disabling bridge assurance on an interface and not configuring bridge assurance on an interface.

Example Bridge Assurance Configuration

To enable bridge assurance on an interface, add the `portnetwork` option to the interface:

```
cumulus@switch:~$ net add interface swp1 stp portnetwork  
cumulus@switch:~$ net pending  
cumulus@switch:~$ net commit
```

This creates the following interface stanza:

```
auto swp1  
iface swp1  
    mstpctl-portnetwork yes
```

You can monitor logs for bridge assurance messages by doing the following:

```
cumulus@switch:~$ sudo grep -in assurance /var/log/syslog | grep mstp  
1365:Jun 25 18:03:17 mstpd: br1007:swp1.1007 Bridge assurance  
inconsistent
```

BPDU Filter

You can enable `bpdufilter` on a switch port, which filters BPDUs in both directions. This effectively disables STP on the port as no BPDUs are transiting.



Using BDPU filter inappropriately can cause layer 2 loops. Use this feature deliberately and with extreme caution.

Example BPDU Filter Configuration

To configure the BPDU filter, add the `portbpdufilter` option to the interface:

```
cumulus@switch:~$ net add interface swp6 stp portbpdufilter  
cumulus@switch:~$ net pending  
cumulus@switch:~$ net commit
```

These commands create the following stanza in the `/etc/network/interfaces` file:

```
auto swp6  
iface swp6  
    mstpctl-portbpdufilter yes
```

Storm Control

Storm control provides protection against excessive inbound BUM (broadcast, unknown unicast, multicast) traffic on layer 2 switch port interfaces, which can cause poor network performance.

You configure storm control for each physical port by editing `/etc/cumulus/switchd.conf`. The configuration persists across reboots and restarting `switchd`. If you change the storm control configuration in this file after rebooting the switch, you must [restart `switchd` \(see page 173\)](#) to activate the new configuration.

To enable broadcast and multicast storm control at 400 packets per second (pps) and 3000 pps, respectively, for `swp1` with `/etc/cumulus/switchd.conf`:

```
# Storm Control setting on a port, in pps, 0 means disable  
interface.swp1.storm_control.broadcast = 400  
interface.swp1.storm_control.multicast = 3000
```

Configuring Other Spanning Tree Parameters

Spanning tree parameters are defined in the IEEE [802.1D](#), [802.1Q](#) specifications and in the table below.

For a comparison of STP parameter configuration between `mstptcl` and other vendors, [please read this knowledge base article](#).

The table below describes the configuration parameters available.



You configure these parameters using NCLU on the interfaces, not the bridge itself. Most of these parameters are blacklisted in `netd.conf` in the `ifupdown_blacklist`.



Parameter	NCLU Command <code>net add interface <interface> stp</code> ...	Description
mstpctl-maxage	maxage	Sets the bridge's <i>maximum age</i> to <code><max_age></code> seconds. The default is 20. The maximum age must meet the condition $2 * (\text{Bridge Forward Delay} - 1 \text{ second}) \geq \text{Bridge Max Age}$.
mstpctl-ageing	ageing	Sets the Ethernet (MAC) address <i>ageing time</i> in <code><time></code> seconds for the bridge when the running version is STP, but not RSTP/MSTP. The default is 300.
mstpctl-fdelay	fdelay	Sets the bridge's <i>bridge forward delay</i> to <code><time></code> seconds. The default is 15. The bridge forward delay must meet the condition $2 * (\text{Bridge Forward Delay} - 1 \text{ second}) \geq \text{Bridge Max Age}$.
mstpctl-maxhops	maxhops	Sets the bridge's <i>maximum hops</i> to <code><max_hops></code> . The default value is 20.
mstpctl-txholdcount	txholdcount	Sets the bridge's <i>bridge transmit hold count</i> to <code><tx_hold_count></code> . The default is 6.
mstpctl-forcevers	forcevers	Sets the bridge's <i>force STP version</i> to either RSTP/STP. MSTP is not supported currently. The default is <i>RSTP</i> .
mstpctl-treepprio	treepprio	Sets the bridge's <i>tree priority</i> to <code><priority></code> for an MSTI instance. The priority value is a number between 0 and 65535 and must be a multiple of 4096. The bridge with the lowest priority is elected the <i>root bridge</i> . The default is 32768. <div style="border: 2px solid red; padding: 10px; margin-top: 10px;">! For <code>msti</code>, only 0 is supported currently.</div>
mstpctl-treeportpprio	treeportpprio	Sets the <i>priority</i> of port <code><port></code> to <code><priority></code> for the MSTI instance. The priority value is a number between 0 and 240 and must be a multiple of 16. The default is 128. <div style="border: 2px solid red; padding: 10px; margin-top: 10px;">! For <code>msti</code>, only 0 is supported currently.</div>
mstpctl-hello	hello	Sets the bridge's <i>bridge hello time</i> to <code><time></code> seconds. The default is 2.
	portpathcost	

Parameter	NCLU Command <code>net add interface <interface> stp</code> ...	Description
<code>mstptctl-portpathcost</code>		<p>Sets the <i>port cost</i> of the port <code><port></code> in bridge <code><bridge></code> to <code><cost></code>. The default is 0.</p> <p><code>mstpctl</code> supports only long mode; that is, 32 bits for the path cost.</p>
<code>mstptctl-portadminedge</code>	<code>portadminedge</code>	<p>Enables/disables the <i>initial edge state</i> of the port <code><port></code> in bridge <code><bridge></code>. The default is <i>no</i>.</p>
<code>mstptctl-portautoedge</code>	<code>portautoedge</code>	<p>Enables/disables the <i>auto transition</i> to/from the edge state of the port <code><port></code> in bridge <code><bridge></code>. The default is <i>yes</i>.</p> <p><code>portautoedge</code> is an enhancement to the standard PortAdminEdge (PortFast) mode, which allows for the automatic detection of edge ports.</p> <div style="border: 2px solid orange; padding: 10px;"> <p>! Edge ports and access ports are not the same thing. Edge ports transition directly to the forwarding state and skip the listening and learning stages. Upstream topology change notifications are not generated when an edge port's link changes state. Access ports only forward untagged traffic; however, there is no such restriction on edge ports, which can forward both tagged and untagged traffic.</p> </div> <p>When a BPDU is received on a port configured with <code>portautoedge</code>, the port ceases to be in the edge port state and transitions into a normal STP port.</p> <p>When BPDUs are no longer received on the interface, the port becomes an edge port, and transitions through the discarding and learning states before resuming forwarding.</p>
<code>mstptctl-portp2p</code>	<code>portp2p</code>	<p>Enables/disables the <i>point-to-point detection mode</i> of the port <code><port></code> in bridge <code><bridge></code>. The default is <i>auto</i>.</p>
<code>mstptctl-portrestrrole</code>	<code>portrestrrole</code>	<p>Enables/disables the ability of the port <code><port></code> in bridge <code><bridge></code> to take the <i>root role</i>. The default is <i>no</i>.</p>
<code>mstptctl-portrestrtcn</code>	<code>portrestrtcn</code>	<p>Enables/disables the ability of the port <code><port></code> in bridge <code><bridge></code> to propagate <i>received topology change notifications</i>. The default is <i>no</i>.</p>
	<code>portnetwork</code>	



Parameter	NCLU Command <code>net add interface <interface> stp</code> ...	Description
mstpctl-portnetwork		Enables/disables the <i>bridge assurance capability</i> for a network port <code><port></code> in bridge <code><bridge></code> . The default is <i>no</i> .
mstpctl-bpduguard	<code>bpduguard</code>	Enables/disables the <i>BPDU guard configuration</i> of the port <code><port></code> in bridge <code><bridge></code> . The default is <i>no</i> .
mstpctl-portbpdufilter	<code>portbpdufilter</code>	Enables/disables the <i>BPDU filter</i> functionality for a port <code><port></code> in bridge <code><bridge></code> . The default is <i>no</i> .
mstpctl-treeportcost	<code>treeportcost</code>	Sets the spanning tree port cost to a value from 0 to 255. The default is 0.

Caveats and Errata

- MSTP is not supported currently. However, interoperability with MSTP networks can be accomplished using PVRSTP or PVSTP.

Related Information

The source code for `mstpd/mstpctl` was written by [Vitalii Demianets](#) and is hosted at the sourceforge URL below.

- [Sourceforge - mstpd project](#)
- [Wikipedia - Spanning Tree Protocol](#)
- [brctl\(8\)](#)
- [bridge-utils-interfaces\(5\)](#)
- [ifUpDown-addons-interfaces\(5\)](#)
- [mstpctl\(8\)](#)
- [mstpctl-utils-interfaces\(5\)](#)

Link Layer Discovery Protocol

The `lldpd` daemon implements the IEEE802.1AB (Link Layer Discovery Protocol, or LLDP) standard. LLDP allows you to know which ports are neighbors of a given port. By default, `lldpd` runs as a daemon and is started at system boot. `lldpd` command line arguments are placed in `/etc/default/lldpd`. `lldpd` configuration options are placed in `/etc/lldpd.conf` or under `/etc/lldpd.d/`.

For more details on the command line arguments and config options, please see `man lldpd(8)`.

`lldpd` supports CDP (Cisco Discovery Protocol, v1 and v2). `lldpd` logs by default into `/var/log/daemon.log` with an `lldpd` prefix.



`lldpcli` is the CLI tool to query the `lldpd` daemon for neighbors, statistics and other running configuration information. See `man lldpcli(8)` for details.

Contents

This chapter covers ...

- Configuring LLDP (see page 252)
- Example `lldpcli` Commands (see page 252)
- Enabling the SNMP Subagent in LLDP (see page 257)
- Caveats and Errata (see page 257)
- Related Information (see page 257)

Configuring LLDP

You configure `lldpd` settings in `/etc/lldpd.conf` or `/etc/lldpd.d/`.

Here is an example persistent configuration:

```
cumulus@switch:~$ sudo cat /etc/lldpd.conf
configure lldp tx-interval 40
configure lldp tx-hold 3
configure system interface pattern-blacklist "eth0"
```

`lldpd` logs to `/var/log/daemon.log` with the `lldp` prefix:

```
cumulus@switch:~$ sudo tail -f /var/log/daemon.log | grep lldp
Aug  7 17:26:17 switch lldpd[1712]: unable to get system name
Aug  7 17:26:17 switch lldpd[1712]: unable to get system name
Aug  7 17:26:17 switch lldpcli[1711]: lldpd should resume operations
Aug  7 17:26:32 switch lldpd[1805]: NET-SNMP version 5.4.3 AgentX
subagent connected
```

Example `lldpcli` Commands

To see all neighbors on all ports/interfaces:

```
cumulus@switch:~$ sudo lldpcli show neighbors
-----
-----
LLDP neighbors:
-----
-----
Interface:    eth0, via: LLDP, RID: 1, Time: 0 day, 17:38:08
  Chassis:
    ChassisID:   mac 08:9e:01:e9:66:5a
```



```
SysName: PIONEERMS22
SysDescr: Cumulus Linux version 2.5.4 running on quanta 1b9
MgmtIP: 192.168.0.22
Capability: Bridge, on
Capability: Router, on
Port:
  PortID:      ifname swp47
  PortDescr:   swp47
-----
-----
Interface:    swp1, via: LLDP, RID: 10, Time: 0 day, 17:08:27
Chassis:
  ChassisID:  mac 00:01:00:00:09:00
  SysName:    MSP-1
  SysDescr:   Cumulus Linux version 3.0.0 running on QEMU
Standard PC (i440FX + PIIIX, 1996)
  MgmtIP:     192.0.2.9
  MgmtIP:     fe80::201:ff:fe00:900
  Capability: Bridge, off
  Capability: Router, on
Port:
  PortID:      ifname swp1
  PortDescr:   swp1
-----
-----
Interface:    swp2, via: LLDP, RID: 10, Time: 0 day, 17:08:27
Chassis:
  ChassisID:  mac 00:01:00:00:09:00
  SysName:    MSP-1
  SysDescr:   Cumulus Linux version 3.0.0 running on QEMU
Standard PC (i440FX + PIIIX, 1996)
  MgmtIP:     192.0.2.9
  MgmtIP:     fe80::201:ff:fe00:900
  Capability: Bridge, off
  Capability: Router, on
Port:
  PortID:      ifname swp2
  PortDescr:   swp2
-----
-----
Interface:    swp3, via: LLDP, RID: 11, Time: 0 day, 17:08:27
Chassis:
  ChassisID:  mac 00:01:00:00:0a:00
  SysName:    MSP-2
  SysDescr:   Cumulus Linux version 3.0.0 running on QEMU
Standard PC (i440FX + PIIIX, 1996)
  MgmtIP:     192.0.2.10
  MgmtIP:     fe80::201:ff:fe00:a00
  Capability: Bridge, off
  Capability: Router, on
Port:
  PortID:      ifname swp1
```



```
PortDescr:      swp1
-----
-----
Interface:      swp4, via: LLDP, RID: 11, Time: 0 day, 17:08:27
Chassis:
  ChassisID:    mac 00:01:00:00:0a:00
  SysName:      MSP-2
  SysDescr:     Cumulus Linux version 3.0.0 running on QEMU
Standard PC (i440FX + PIIX, 1996)
  MgmtIP:       192.0.2.10
  MgmtIP:       fe80::201:ff:fe00:a00
  Capability:   Bridge, off
  Capability:   Router, on
Port:
  PortID:       ifname swp2
  PortDescr:    swp2
-----
-----
Interface:      swp49s1, via: LLDP, RID: 9, Time: 0 day, 16:55:00
Chassis:
  ChassisID:    mac 00:01:00:00:0c:00
  SysName:      TORC-1-2
  SysDescr:     Cumulus Linux version 3.0.0 running on QEMU
Standard PC (i440FX + PIIX, 1996)
  MgmtIP:       192.0.2.12
  MgmtIP:       fe80::201:ff:fe00:c00
  Capability:   Bridge, on
  Capability:   Router, on
Port:
  PortID:       ifname swp6
  PortDescr:    swp6
-----
-----
Interface:      swp49s0, via: LLDP, RID: 9, Time: 0 day, 16:55:00
Chassis:
  ChassisID:    mac 00:01:00:00:0c:00
  SysName:      TORC-1-2
  SysDescr:     Cumulus Linux version 3.0.0 running on QEMU
Standard PC (i440FX + PIIX, 1996)
  MgmtIP:       192.0.2.12
  MgmtIP:       fe80::201:ff:fe00:c00
  Capability:   Bridge, on
  Capability:   Router, on
Port:
  PortID:       ifname swp5
  PortDescr:    swp5
-----
```

To see lldpd statistics for all ports:



```
cumulus@switch:~$ sudo lldpcli show statistics
```

```
-----  
LLDP statistics:
```

```
Interface: eth0
```

```
    Transmitted: 9423  
    Received: 17634  
    Discarded: 0  
    Unrecognized: 0  
    Ageout: 10  
    Inserted: 20  
    Deleted: 10
```

```
-----  
Interface: swp1
```

```
    Transmitted: 9423  
    Received: 6264  
    Discarded: 0  
    Unrecognized: 0  
    Ageout: 0  
    Inserted: 2  
    Deleted: 0
```

```
-----  
Interface: swp2
```

```
    Transmitted: 9423  
    Received: 6264  
    Discarded: 0  
    Unrecognized: 0  
    Ageout: 0  
    Inserted: 2  
    Deleted: 0
```

```
-----  
Interface: swp3
```

```
    Transmitted: 9423  
    Received: 6265  
    Discarded: 0  
    Unrecognized: 0  
    Ageout: 0  
    Inserted: 2  
    Deleted: 0
```

```
-----  
... and more (output truncated to fit this document)
```

To see lldpd statistics summary for all ports:

```
cumulus@switch:~$ sudo lldpcli show statistics summary
```

```
-----  
LLDP Global statistics:
```

```
-----  
Summary of stats:
```

```
    Transmitted: 648186
```



```
Received:      437557
Discarded:    0
Unrecognized: 0
Ageout:       10
Inserted:    38
Deleted:     10
```

To see the lldpd running configuration:

```
cumulus@switch:~$ sudo lldpccli show running-configuration
-----
Global configuration:
-----
Configuration:
  Transmit delay: 30
  Transmit hold: 4
  Receive mode: no
  Pattern for management addresses: (none)
  Interface pattern: (none)
  Interface pattern blacklist: (none)
  Interface pattern for chassis ID: (none)
  Override description with: (none)
  Override platform with: Linux
  Override system name with: (none)
  Advertise version: yes
  Update interface descriptions: no
  Promiscuous mode on managed interfaces: no
  Disable LLDP-MED inventory: yes
  LLDP-MED fast start mechanism: yes
  LLDP-MED fast start interval: 1
  Source MAC for LLDP frames on bond slaves: local
  Portid TLV Subtype for lldp frames: ifname
-----
```

Runtime Configuration (Advanced)



A runtime configuration does not persist when you reboot the switch — all changes are lost.

To configure active interfaces:

```
cumulus@switch:~$ sudo lldpccli configure system interface pattern "swp
* "
```

To configure inactive interfaces:



```
cumulus@switch:~$ sudo lldpcli configure system interface pattern-blacklist "eth0"
```



The active interface list always overrides the inactive interface list.

To reset any interface list to none:

```
cumulus@switch:~$ sudo lldpcli configure system interface pattern-blacklist ""
```

Enabling the SNMP Subagent in LLDP

LLDP does not enable the SNMP subagent by default. You need to edit `/etc/default/lldpd` and enable the `-x` option.

```
cumulus@switch:~$ sudo nano /etc/default/lldpd

# Add "-x" to DAEMON_ARGS to start SNMP subagent

# Enable CDP by default
DAEMON_ARGS="-c"
```

Caveats and Errata

- Annex E (and hence Annex D) of IEEE802.1AB (lldp) is not supported.

Related Information

- [GitHub - llpd project](#)
- [Wikipedia - Link Layer Discovery Protocol](#)

Prescriptive Topology Manager - PTM

In data center topologies, right cabling is a time-consuming endeavor and is error prone. Prescriptive Topology Manager (PTM) is a dynamic cabling verification tool to help detect and eliminate such errors. It takes a Graphviz-DOT specified network cabling plan (something many operators already generate), stored in a `topology.dot` file, and couples it with runtime information derived from LLDP to verify that the cabling matches the specification. The check is performed on every link transition on each node in the network.

You can customize the `topology.dot` file to control `ptmd` at both the global/network level and the node /port level.

PTM runs as a daemon, named `ptmd`.

For more information, see `man ptmd(8)`.



Contents

This chapter covers ...

- Supported Features (see page 258)
- Configuring PTM (see page 258)
- Basic Topology Example (see page 259)
- ptmd Scripts (see page 260)
- Configuration Parameters (see page 260)
 - Host-only Parameters (see page 260)
 - Global Parameters (see page 261)
 - Per-port Parameters (see page 261)
 - Templates (see page 262)
 - Supported BFD and LLDP Parameters (see page 262)
- Bidirectional Forwarding Detection (BFD) (see page 263)
- Checking Link State with Quagga (see page 264)
- Using ptmd Service Commands (see page 264)
- Using ptmctl Commands (see page 265)
 - ptmctl Examples (see page 265)
 - ptmctl Error Outputs (see page 267)
- Caveats and Errata (see page 268)
- Related Information (see page 268)

Supported Features

- Topology verification using LLDP. `ptmd` creates a client connection to the LLDP daemon, `lldpd`, and retrieves the neighbor relationship between the nodes/ports in the network and compares them against the prescribed topology specified in the `topology.dot` file.
- Only physical interfaces, like `swp1` or `eth0`, are currently supported. Cumulus Linux does not support specifying virtual interfaces like bonds or subinterfaces like `eth0.200` in the topology file.
- Forwarding path failure detection using **Bidirectional Forwarding Detection** (BFD); however, demand mode is not supported. For more information on how BFD operates in Cumulus Linux, read the **Bidirectional Forwarding Detection - BFD** (see page 549) chapter and read `man ptmd(8)`.
- Integration with Quagga (PTM to Quagga notification).
- Client management: `ptmd` creates an abstract named socket `/var/run/ptmd.socket` on startup. Other applications can connect to this socket to receive notifications and send commands.
- Event notifications: see Scripts below.
- User configuration via a `topology.dot` file; **see below** (see page 258).

Configuring PTM

`ptmd` verifies the physical network topology against a DOT-specified network graph file, `/etc/ptm.d/topology.dot`.



This file must be present or else `ptmd` will not start. You can specify an alternate file using the `-c` option.

PTM supports [undirected graphs](#).

At startup, `ptmd` connects to `lldpd`, the LLDP daemon, over a Unix socket and retrieves the neighbor name and port information. It then compares the retrieved port information with the configuration information that it read from the topology file. If there is a match, then it is a PASS, else it is a FAIL.

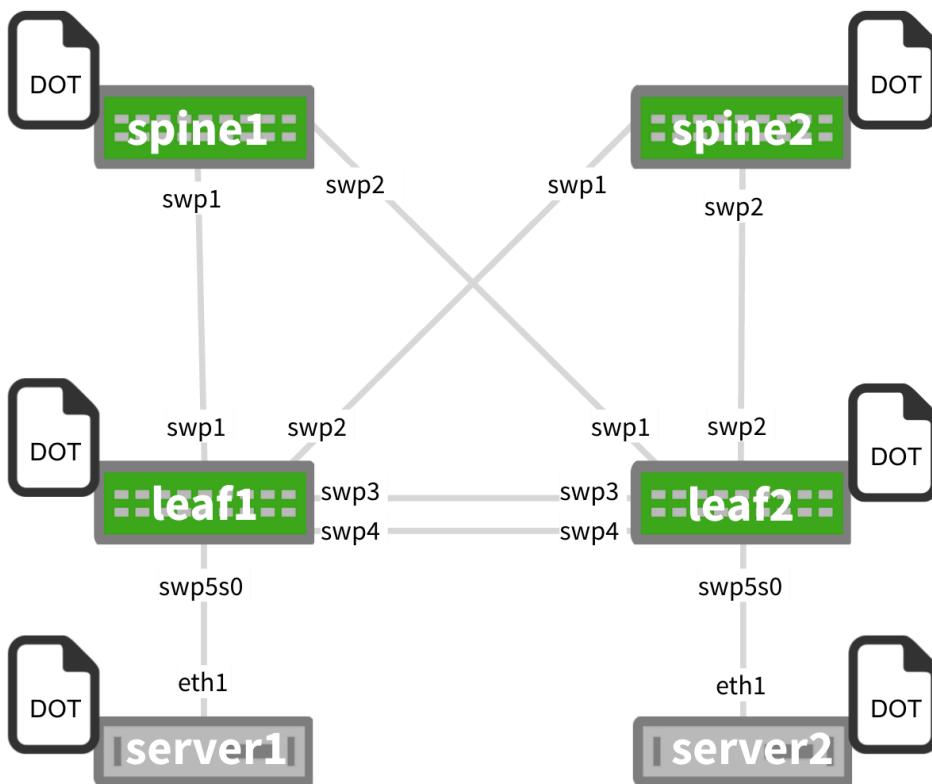


PTM performs its LLDP neighbor check using the PortID ifname TLV information. Previously, it used the PortID port description TLV information.

Basic Topology Example

This is a basic example DOT file and its corresponding topology diagram. You should use the same `topology.dot` file on all switches, and don't split the file per device; this allows for easy automation by pushing/pulling the same exact file on each device!

```
graph G {
    "spine1": "swp1" -- "leaf1": "swp1";
    "spine1": "swp2" -- "leaf2": "swp1";
    "spine2": "swp1" -- "leaf1": "swp2";
    "spine2": "swp2" -- "leaf2": "swp2";
    "leaf1": "swp3" -- "leaf2": "swp3";
    "leaf1": "swp4" -- "leaf2": "swp4";
    "leaf1": "swp5s0" -- "server1": "eth1";
    "leaf2": "swp5s0" -- "server2": "eth1";
}
```



ptmd Scripts

`ptmd` executes scripts at `/etc/ptm.d/if-topo-pass` and `/etc/ptm.d/if-topo-fail` for each interface that goes through a change, running `if-topo-pass` when an LLDP or BFD check passes and running `if-topo-fails` when the check fails. The scripts receive an argument string that is the result of the `ptmctl` command, described in the [ptmd commands section below](#) (see page 264).

You should modify these default scripts as needed.

Configuration Parameters

You can configure `ptmd` parameters in the topology file. The parameters are classified as host-only, global, per-port/node and templates.

Host-only Parameters

Host-only parameters apply to the entire host on which PTM is running. You can include the `hostnametype` host-only parameter, which specifies whether PTM should use only the host name (`hostname`) or the fully-qualified domain name (`fqdn`) while looking for the `self-node` in the graph file. For example, in the graph file below, PTM will ignore the FQDN and only look for `switch04`, since that is the host name of the switch it's running on:





It's a good idea to always wrap the hostname in double quotes, like "www.example.com". Otherwise, `ptmd` can fail if you specify a fully-qualified domain name as the hostname and do not wrap it in double quotes.

Further, to avoid errors when starting the `ptmd` process, make sure that `/etc/hosts` and `/etc/hostname` both reflect the hostname you are using in the `topology.dot` file.

```
graph G {
    hostnametype="hostname"
    BFD="upMinTx=150,requiredMinRx=250"
    "cumulus":"swp44" -- "switch04.cumulusnetworks.com": "swp20"
    "cumulus":"swp46" -- "switch04.cumulusnetworks.com": "swp22"
}
```

However, in this next example, PTM will compare using the FQDN and look for `switch05.cumulusnetworks.com`, which is the FQDN of the switch it's running on:

```
graph G {
    hostnametype="fqdn"
    "cumulus":"swp44" -- "switch05.cumulusnetworks.com": "swp20"
    "cumulus":"swp46" -- "switch05.cumulusnetworks.com": "swp22"
}
```

Global Parameters

Global parameters apply to every port listed in the topology file. There are two global parameters: LLDP and BFD. LLDP is enabled by default; if no keyword is present, default values are used for all ports. However, BFD is disabled if no keyword is present, unless there is a per-port override configured. For example:

```
graph G {
    LLDP=""
    BFD="upMinTx=150,requiredMinRx=250,afi=both"
    "cumulus":"swp44" -- "qct-ly2-04": "swp20"
    "cumulus":"swp46" -- "qct-ly2-04": "swp22"
}
```

Per-port Parameters

Per-port parameters provide finer-grained control at the port level. These parameters override any global or compiled defaults. For example:

```
graph G {
    LLDP=""
    BFD="upMinTx=300,requiredMinRx=100"
```

```

    "cumulus" : "swp44" -- "qct-ly2-04" : "swp20" [BFD="upMinTx=150,
requiredMinRx=250,afi=both"]
    "cumulus" : "swp46" -- "qct-ly2-04" : "swp22"
}

```

Templates

Templates provide flexibility in choosing different parameter combinations and applying them to a given port. A template instructs `ptmd` to reference a named parameter string instead of a default one. There are two parameter strings `ptmd` supports:

- `bfdtmpl`, which specifies a custom parameter tuple for BFD.
- `lldptmpl`, which specifies a custom parameter tuple for LLDP.

For example:

```

graph G {
    LLDP=""
    BFD="upMinTx=300,requiredMinRx=100"
    BFD1="upMinTx=200,requiredMinRx=200"
    BFD2="upMinTx=100,requiredMinRx=300"
    LLDP1="match_type=ifname"
    LLDP2="match_type=portdescr"
    "cumulus" : "swp44" -- "qct-ly2-04" : "swp20" [BFD="bfdtmpl=BFD1"
, LLDP="lldptmpl=LLDP1"]
    "cumulus" : "swp46" -- "qct-ly2-04" : "swp22" [BFD="bfdtmpl=BFD2"
, LLDP="lldptmpl=LLDP2"]
    "cumulus" : "swp46" -- "qct-ly2-04" : "swp22"
}

```

In this template, LLDP1 and LLDP2 are templates for LLDP parameters while BFD1 and BFD2 are templates for BFD parameters.

Supported BFD and LLDP Parameters

`ptmd` supports the following BFD parameters:

- `upMinTx`: the minimum transmit interval, which defaults to 300ms, specified in milliseconds.
- `requiredMinRx`: the minimum interval between received BFD packets, which defaults to 300ms, specified in milliseconds.
- `detectMult`: the detect multiplier, which defaults to 3, and can be any non-zero value.
- `afi`: the address family to be supported for the edge. The address family must be one of the following:
 - `v4`: BFD sessions will be built for only IPv4 connected peer. This is the default value.
 - `v6`: BFD sessions will be built for only IPv6 connected peer.
 - `both`: BFD sessions will be built for both IPv4 and IPv6 connected peers.

The following is an example of a topology with BFD applied at the port level:



```
graph G {
    "cumulus-1": "swp44" -- "cumulus-2": "swp20" [BFD="upMinTx=300,
requiredMinRx=100,afi=v6"]
    "cumulus-1": "swp46" -- "cumulus-2": "swp22" [BFD="detectMult=4
"]
}
```

ptmd supports the following LLDP parameters:

- `match_type`, which defaults to the interface name (`ifname`), but can accept a port description (`portdescr`) instead if you want `lldpd` to compare the topology against the port description instead of the interface name. You can set this parameter globally or at the per-port level.
- `match_hostname`, which defaults to the host name (`hostname`), but enables PTM to match the topology using the fully-qualified domain name (`fqdn`) supplied by LLDP.

The following is an example of a topology with LLDP applied at the port level:

```
graph G {
    "cumulus-1": "swp44" -- "cumulus-2": "swp20" [LLDP="match_hostname=fqdn"]
    "cumulus-1": "swp46" -- "cumulus-2": "swp22" [LLDP="match_type=portdescr"]
}
```



When you specify `match_hostname=fqdn`, `ptmd` will match the entire FQDN, like *cumulus-2.domain.com* in the example below. If you do not specify anything for `match_hostname`, `ptmd` will match based on `hostname` only, like *cumulus-3* below, and ignore the rest of the URL:

```
graph G {
    "cumulus-1": "swp44" -- "cumulus-2.domain.com": "swp20"
[LLDP="match_hostname=fqdn"]
    "cumulus-1": "swp46" -- "cumulus-3": "swp22" [LLDP="match_type=portdescr"]
```

Bidirectional Forwarding Detection (BFD)

BFD provides low overhead and rapid detection of failures in the paths between two network devices. It provides a unified mechanism for link detection over all media and protocol layers. Use BFD to detect failures for IPv4 and IPv6 single or multihop paths between any two network devices, including unidirectional path failure detection. For information about configuring BFD using PTM, see the [BFD chapter](#) (see page 549).



Checking Link State with Quagga

The Quagga routing suite enables additional checks to ensure that routing adjacencies are formed only on links that have connectivity conformant to the specification, as determined by `ptmd`.



You only need to do this to check link state; you don't need to enable PTM to determine BFD status.

The check is enabled by default. Every interface has an implied `ptm-enable` line in the configuration stanza in the interfaces file.

To disable the checks, delete the `ptm-enable` parameter from the interface. For example:

```
cumulus@switch:~$ net del interface swp51 ptm-enable  
cumulus@switch:~$ net pending  
cumulus@switch:~$ net commit
```

If you need to re-enable PTM for that interface, run:

```
cumulus@switch:~$ net add interface swp51 ptm-enable  
cumulus@switch:~$ net pending  
cumulus@switch:~$ net commit
```

With PTM enabled on an interface, the `zebra` daemon connects to `ptmd` over a Unix socket. Any time there is a change of status for an interface, `ptmd` sends notifications to `zebra`. Zebra maintains a `ptm-status` flag per interface and evaluates routing adjacency based on this flag. To check the per-interface `ptm-status`:

```
cumulus@switch:~$ net show interface swp1  
  
Interface swp1 is up, line protocol is up  
  Link ups:      0      last: (never)  
  Link downs:    0      last: (never)  
  PTM status: disabled  
  vrf: Default-IP-Routing-Table  
  index 3 metric 0 mtu 1550  
  flags: <UP,BROADCAST,RUNNING,MULTICAST>  
  HWaddr: c4:54:44:bd:01:41
```

Using `ptmd` Service Commands

PTM sends client notifications in CSV format.

`cumulus@switch:~$ sudo systemctl start|restart|force-reload ptmd.service`: Starts or restarts the `ptmd` service. The `topology.dot` file must be present in order for the service to start.



cumulus@switch:~\$ sudo systemctl reload ptmd.service: Instructs ptmd to read the topology.dot file again without restarting, applying the new configuration to the running state.

cumulus@switch:~\$ sudo systemctl stop ptmd.service: Stops the ptmd service.

cumulus@switch:~\$ sudo systemctl status ptmd.service: Retrieves the current running state of ptmd.

Using ptmctl Commands

ptmctl is a client of ptmd; it retrieves the operational state of the ports configured on the switch and information about BFD sessions from ptmd. ptmctl parses the CSV notifications sent by ptmd.

See `man ptmctl` for more information.

ptmctl Examples

For basic output, use `ptmctl` without any options:

```
cumulus@switch:~$ sudo ptmctl

-----
port 缆线 BFD      BFD
      状态   状态   对端
      local   类型
-----
swp1 通过   通过   11.0.0.2
      N/A     N/A
      N/A     N/A
      N/A     N/A
      N/A     N/A
```

For more detailed output, use the `-d` option:

```
cumulus@switch:~$ sudo ptmctl -d

-----
port 缆线 exp      行为   系统名称 portID  portDescr  匹配
      BFD    BFD      BFD      det_mult tx_timeout
      rx_timeout echo_tx_timeout echo_rx_timeout max_hop_cnt
      状态   nbr      nbr
      on
upd    命令   状态   对端   DownDiag
-----
swp45 通过   h1:swp1 h1:swp1 h1      swp1      swp1      IfName 5m:
5s    N/A     N/A     N/A     N/A      N/A       N/A       N/A
/A          N/A
      N/A
swp46 失效   h2:swp1 h2:swp1 h2      swp1      swp1      IfName 5m:
5s    N/A     N/A     N/A     N/A      N/A       N/A       N/A
/A          N/A
      N/A
```

To return information on active BFD sessions `ptmd` is tracking, use the `-b` option:

```
cumulus@switch:~$ sudo ptmctl -b

-----
port  peer          state local      type      diag
-----
swp1  11.0.0.2     Up    N/A        singlehop  N/A
N/A   12.12.12.1   Up    12.12.12.4  multihop  N/A
```

To return LLDP information, use the `-l` option. It returns only the active neighbors currently being tracked by `ptmd`.

```
cumulus@switch:~$ sudo ptmctl -l

-----
port  sysname  portID  port      match  last
                descr   on       upd
-----
swp45 h1        swp1    swp1    IfName  5m:59s
swp46 h2        swp1    swp1    IfName  5m:59s
```

To return detailed information on active BFD sessions `ptmd` is tracking, use the `-b` and `-d` options (results are for an IPv6-connected peer):

```
cumulus@switch:~$ sudo ptmctl -b -d

-----
-----  

-----  

port  peer          state local      type      diag  det
tx_timeout  rx_timeout echo        echo      max   rx_ctrl
tx_ctrl    rx_echo   tx_echo  

mult                  tx_timeout  rx_timeout
hop_cnt  

-----  

-----  

-----  

swp1  fe80::202:ff:fe00:1  Up    N/A        singlehop  N/A   3   300
900   0           0         N/A        187172   185986  0
0
swp1  3101:abc:bcad::2    Up    N/A        singlehop  N/A   3   300
900   0           0         N/A        501      533    0
0
```

ptmctl Error Outputs

If there are errors in the topology file or there isn't a session, PTM will return appropriate outputs. Typical error strings are:

```
Topology file error [/etc/ptm.d/topology.dot] [cannot find node
cumulus] -
please check /var/log/ptmd.log for more info

Topology file error [/etc/ptm.d/topology.dot] [cannot open file
(errno 2)] -
please check /var/log/ptmd.log for more info

No Hostname/MgmtIP found [Check LLDPD daemon status] -
please check /var/log/ptmd.log for more info

No BFD sessions . Check connections

No LLDP ports detected. Check connections

Unsupported command
```

For example:

```
cumulus@switch:~$ sudo ptmctl
-----
---  
cmd      error
-----
---  
get-status Topology file error [/etc/ptm.d/topology.dot] [cannot
open file (errno 2)] - please check /var/log/ptmd.log for more info
```



If you encounter errors with the `topology.dot` file, you can use `dot` (included in the Graphviz package) to validate the syntax of the topology file.

By simply opening the topology file with Graphviz, you can ensure that it is readable and that the file format is correct.

If you edit `topology.dot` file from a Windows system, be sure to double check the file formatting; there may be extra characters that keep the graph from working correctly.



Caveats and Errata

- Prior to version 2.1, Cumulus Linux stored the `ptmd` configuration files in `/etc/cumulus/ptm.d`. When you upgrade to version 2.1 or later, all the existing `ptmd` files are copied from their original location to `/etc/ptm.d` with a `dpkg-old` extension, except for `topology.dot`, which gets copied to `/etc/ptm.d`.
If you customized the `if-topo-pass` and `if-topo-fail` scripts, they are also copied to `dpkg-old`, and you must modify them so they can parse the CSV output correctly.
Sample `if-topo-pass` and `if-topo-fail` scripts are available in `/etc/ptm.d`. A sample `topology.dot` file is available in `/usr/share/doc/ptmd/examples`.

Related Information

- Bidirectional Forwarding Detection (BFD)
- Graphviz
- LLDP on Wikipedia
- PTMd GitHub repo

Bonding - Link Aggregation

Linux bonding provides a method for aggregating multiple network interfaces (*slaves*) into a single logical bonded interface (*bond*). The only bonding mode supported in Cumulus Linux is the IEEE 802.3ad link aggregation mode, which allows one or more links to be aggregated together to form a *link aggregation group* (LAG), such that a media access control (MAC) client can treat the link aggregation group as if it were a single link.

The benefits of link aggregation include:

- Linear scaling of bandwidth as links are added to LAG
- Load balancing
- Failover protection

Cumulus Linux uses version 1 of the LAG control protocol (LACP).

To temporarily bring up a bond even when there is no LACP partner, use [LACP Bypass](#) (see page 333).

Contents

This chapter covers ...

- [Hash Distribution](#) (see page 268)
- [Creating a Bond](#) (see page 269)
 - [Configuration Options](#) (see page 269)
- [Example Configuration: Bonding 4 Slaves](#) (see page 270)
- [Caveats and Errata](#) (see page 272)
- [Related Information](#) (see page 272)



Hash Distribution

Egress traffic through a bond is distributed to a slave based on a packet hash calculation, providing load balancing over the slaves; many conversation flows are distributed over all available slaves to load balance the total traffic. Traffic for a single conversation flow always hashes to the same slave.

The hash calculation uses packet header data to pick which slave to transmit the packet to:

- For IP traffic, IP header source and destination fields are used in the calculation.
- For IP + TCP/UDP traffic, source and destination ports are included in the hash calculation.



In a failover event, the hash calculation is adjusted to steer traffic over available slaves.

Creating a Bond

Bonds can be created and configured using the Network Command Line Utility ([NCLU \(see page 80\)](#)). Follow the steps below to create a new bond:

1. SSH into the switch.
2. Add a bond using the `net add bond [bond-name] bond slaves [slaves]` command, replacing `[bond-name]` with the name of the bond, and `[slaves]` with the list of slaves:

```
cumulus@switch:~$ net add bond [bond-name] bond slaves [slaves]
cumulus@switch:~$ net pending
cumulus@switch:~$ net commit
```



The name of the bond must be:

- Compliant with Linux interface naming conventions.
- Unique within the switch.

Configuration Options

The configuration options, and their default values, are listed in the table below.



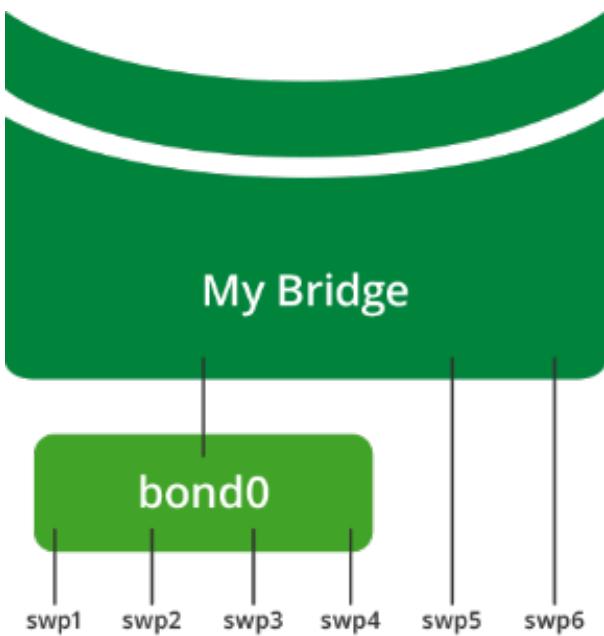
Each bond configuration option, except for `bond slaves`, is set to the recommended value by default in Cumulus Linux. They should only be configured if a different setting is needed. For more information on configuration values, refer to the Related Information section below.

NCLU Configuration Option	Description	Default Value
<code>bond mode</code>	The defined bonding mode.	802.3ad

NCLU Configuration Option	Description	Default Value
	Cumulus Linux <i>only</i> supports IEEE 802.3ad link aggregation mode. This setting must not be changed.	
bond slaves	The list of slaves in the bond.	N/A
bond miimon	Defines how often the link state of each slave is inspected for failures.	100
bond use-carrier	Determines the link state.	1
bond xmit-hash-policy	Hash method used to select the slave for a given packet. ❗ This setting must not be changed.	layer3+4
bond lacp-rate	Sets the rate to ask the link partner to transmit LACP control packets. ⚠ At this time, you can set the LACP rate to slow (0) only by editing the setting <code>bond-lacp-rate</code> to <code>0</code> in the <code>/etc/network/interfaces</code> file.	1
bond min-links	Defines the minimum number of links that must be active before the bond is put into service. ℹ A value greater than 1 is useful if higher level services need to ensure a minimum aggregate bandwidth level before activating a bond. Keeping <code>bond-min-links</code> set to 1 indicates the bond must have at least one active member. If the number of active members drops below the <code>bond-min-links</code> setting, the bond will appear to upper-level protocols as <code>link-down</code> . When the number of active links returns to greater than or equal to <code>bond-min-links</code> , the bond will become <code>link-up</code> .	1

Example Configuration: Bonding 4 Slaves

In the following example, the front panel port interfaces swp1-swp4 are slaves in bond0, while swp5 and swp6 are not part of bond0.



(i) Example Bond Configuration

The following commands create a bond with four slaves:

```

cumulus@switch:~$ net add bond bond0 address 10.0.0.1/30
cumulus@switch:~$ net add bond bond0 bond slaves swp1-4
cumulus@switch:~$ net pending
cumulus@switch:~$ net commit
  
```

These commands create this code snippet in the /etc/network/interfaces file:

```

auto bond0
iface bond0
    address 10.0.0.1/30
    bond-slaves swp1 swp2 swp3 swp4
  
```



If you are intending that the bond become part of a bridge, you don't need to specify an IP address.

When networking is started on switch, bond0 is created as MASTER and interfaces swp1-swp4 come up in SLAVE mode, as seen in the `ip link show` command:

```

cumulus@switch:~$ ip link show
...
  
```

```

3: swp1: <BROADCAST,MULTICAST,SLAVE,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast
    master bond0 state UP mode DEFAULT qlen 500
        link/ether 44:38:39:00:03:c1 brd ff:ff:ff:ff:ff:ff
4: swp2: <BROADCAST,MULTICAST,SLAVE,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast
    master bond0 state UP mode DEFAULT qlen 500
        link/ether 44:38:39:00:03:c1 brd ff:ff:ff:ff:ff:ff
5: swp3: <BROADCAST,MULTICAST,SLAVE,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast
    master bond0 state UP mode DEFAULT qlen 500
        link/ether 44:38:39:00:03:c1 brd ff:ff:ff:ff:ff:ff
6: swp4: <BROADCAST,MULTICAST,SLAVE,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast
    master bond0 state UP mode DEFAULT qlen 500
        link/ether 44:38:39:00:03:c1 brd ff:ff:ff:ff:ff:ff

...
55: bond0: <BROADCAST,MULTICAST,MASTER,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP mode DEFAULT
    link/ether 44:38:39:00:03:c1 brd ff:ff:ff:ff:ff:ff

```



All slave interfaces within a bond have the same MAC address as the bond. Typically, the first slave added to the bond donates its MAC address as the bond MAC address, while the other slaves' MAC addresses are set to the bond MAC address.

The bond MAC address is used as source MAC address for all traffic leaving the bond, and provides a single destination MAC address to address traffic to the bond.

Caveats and Errata

- An interface cannot belong to multiple bonds.
- A bond can have subinterfaces, but not the other way around.
- A bond cannot enslave VLAN subinterfaces.
- Slave ports within a bond should all be set to the same speed/duplex, and should match the link partner's slave ports.

Related Information

- [Linux Foundation - Bonding](#)
- [802.3ad \(Accessible writeup\)](#)
- [Wikipedia - Link aggregation](#)

Ethernet Bridging - VLANs

Ethernet bridges provide a means for hosts to communicate through layer 2, by connecting all of the physical and logical interfaces in the system into a single layer 2 domain. The bridge is a logical interface with a MAC address and an [MTU \(see page 210\)](#) (maximum transmission unit). The bridge MTU is the

minimum MTU among all its members. The bridge's MAC address is inherited from the first interface that is added to the bridge as a member. The bridge MAC address remains unchanged until the member interface is removed from the bridge, at which point the bridge will inherit from the next member interface, if any. The bridge can also be assigned an IP address, as discussed [below \(see page 274\)](#).



Bridge members can be individual physical interfaces, bonds or logical interfaces that traverse an 802.1Q VLAN trunk.



Cumulus Networks recommends using [VLAN-aware mode \(see page 277\)](#) bridges, rather than *traditional mode* bridges. The bridge driver in Cumulus Linux is capable of VLAN filtering, which allows for configurations that are similar to incumbent network devices. While Cumulus Linux supports Ethernet bridges in traditional mode, Cumulus Networks recommends using [VLAN-aware](#) (see page 277) mode.



For a comparison of traditional and VLAN-aware modes, read [this knowledge base article](#).



Cumulus Linux does not put all ports into a bridge by default.



You can configure both VLAN-aware and traditional mode bridges on the same network in Cumulus Linux; however you cannot have more than one VLAN-aware bridge on a given switch.



Contents

This chapter covers ...

- Creating a VLAN-aware Bridge (see page 274)
- Creating a Traditional Mode Bridge (see page 274)
- Configuring Bridge MAC Addresses (see page 274)
- Configuring an SVI (Switch VLAN Interface) (see page 274)
 - Keeping the SVI in an UP State (see page 275)
- Caveats and Errata (see page 277)
- Related Information (see page 277)

Creating a VLAN-aware Bridge

To learn about VLAN-aware bridges and how to configure them, read [VLAN-aware Bridge Mode for Large-scale Layer 2 Environments](#) (see page 277).

Creating a Traditional Mode Bridge

To create a traditional mode bridge, see [Traditional Mode Bridges](#) (see page 288).

Configuring Bridge MAC Addresses

The MAC address for a frame is learned when the frame enters the bridge via an interface. The MAC address is recorded in the bridge table, and the bridge forwards the frame to its intended destination by looking up the destination MAC address. The MAC entry is then maintained for a period of time defined by the `bridge-ageing` configuration option. If the frame is seen with the same source MAC address before the MAC entry age is exceeded, the MAC entry age is refreshed; if the MAC entry age is exceeded, the MAC address is deleted from the bridge table.

The following example output shows a MAC address table for the bridge:

```
cumulus@switch:~$ net show bridge macs
VLAN      Master      Interface      MAC          TunnelDest
State     Flags       LastSeen
-----  -----
-----  -----
untagged  bridge      swp1        44:38:39:00:00:
03                  00:00:15
untagged  bridge      swp1        44:38:39:00:00:04
permanent           20 days, 01:14:03
```

Configuring an SVI (Switch VLAN Interface)

Bridges can be included as part of a routing topology after being assigned an IP address. This enables hosts within the bridge to communicate with other hosts outside of the bridge, via a *switch VLAN interface* (SVI), which provides layer 3 routing. The IP address of the bridge is typically from the same subnet as the bridge's member hosts.



When an interface is added to a bridge, it ceases to function as a router interface, and the IP address on the interface, if any, becomes unreachable.

To configure the SVI, use [NCLU \(see page 80\)](#):

```
cumulus@switch:~$ net add bridge bridge ports swp1-2
cumulus@switch:~$ net add vlan 10 ip address 10.100.100.1/24
cumulus@switch:~$ net pending
cumulus@switch:~$ net commit
```

These commands create the following SVI configuration in the `/etc/network/interfaces` file:

```
auto bridge
iface bridge
    bridge-ports swp1 swp2
    bridge-vids 10
    bridge-vlan-aware yes

auto vlan10
iface vlan10
    address 10.100.100.1/24
    vlan-id 10
    vlan-raw-device bridge
```

Alternately, you can use the `bridge.VLAN-ID` naming convention for the SVI. The following example configuration can be manually created in the `/etc/network/interfaces` file, which functions identically to the above configuration:

```
auto bridge
iface bridge
    bridge-ports swp1 swp2
    bridge-vids 10
    bridge-vlan-aware yes

auto bridge.10
iface bridge.10
    address 10.100.100.1/24
```

Keeping the SVI in an UP State

When a switch is initially configured, all southbound bridge ports may be down, which means that, by default, the SVI is also down. However, you may want to force the SVI to always be up, to perform connectivity testing, for example. To do this, you essentially need to disable interface state tracking, leaving the SVI in the UP state always, even if all member ports are down. Other implementations describe this feature as "no autostate".



In Cumulus Linux, you can keep the SVI perpetually UP by creating a dummy interface, and making the dummy interface a member of the bridge. Consider the following configuration, without a dummy interface in the bridge:

```
cumulus@switch:~$ cat /etc/network/interfaces
...
auto bridge
iface bridge
    bridge-vlan-aware yes
    bridge-ports swp3
    bridge-vids 100
    bridge-pvid 1
...
...
```

With this configuration, when swp3 is down, the SVI is also down:

```
cumulus@switch:~$ ip link show swp3
5: swp3: <BROADCAST,MULTICAST> mtu 1500 qdisc pfifo_fast master
    bridge state DOWN mode DEFAULT group default qlen 1000
        link/ether 2c:60:0c:66:b1:7f brd ff:ff:ff:ff:ff:ff
cumulus@switch:~$ ip link show bridge
35: bridge: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc
noqueue state DOWN mode DEFAULT group default
    link/ether 2c:60:0c:66:b1:7f brd ff:ff:ff:ff:ff:ff
```

Now add the dummy interface to your network configuration:

1. Create a dummy interface, and add it to the bridge configuration. You do this by editing the `/etc/network/interfaces` file and adding the dummy interface stanza before the bridge stanza:

```
cumulus@switch:~$ sudo nano /etc/network/interfaces
...
auto dummy
iface dummy
    link-type dummy

auto bridge
iface bridge
...
```

2. Continue editing the `interfaces` file. Add the dummy interface to the `bridge-ports` line in the bridge configuration:

```
auto bridge
```



```
iface bridge
    bridge-vlan-aware yes
    bridge-ports swp3 dummy
    bridge-vids 100
    bridge-pvid 1
```

3. Save and exit the file, then reload the configuration:

```
cumulus@switch:~$ sudo ifreload -a
```

Now, even when swp3 is down, both the dummy interface and the bridge remain up:

```
cumulus@switch:~$ ip link show swp3
5: swp3: <BROADCAST,MULTICAST> mtu 1500 qdisc pfifo_fast master
bridge state DOWN mode DEFAULT group default qlen 1000
    link/ether 2c:60:0c:66:b1:7f brd ff:ff:ff:ff:ff:ff
cumulus@switch:~$ ip link show dummy
37: dummy: <BROADCAST,NOARP,UP,LOWER_UP> mtu 1500 qdisc noqueue
master bridge state UNKNOWN mode DEFAULT group default
    link/ether 66:dc:92:d4:f3:68 brd ff:ff:ff:ff:ff:ff
cumulus@switch:~$ ip link show bridge
35: bridge: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue
state UP mode DEFAULT group default
    link/ether 2c:60:0c:66:b1:7f brd ff:ff:ff:ff:ff:ff
```

Caveats and Errata

- A bridge cannot contain multiple subinterfaces of the **same** port. Attempting this configuration results in an error.
- In environments where both VLAN-aware and traditional bridges are in use, if a traditional bridge has a subinterface of a bond that is a normal interface in a VLAN-aware bridge, the bridge will be flapped when the traditional bridge's bond subinterface is brought down.

Related Information

- [Linux Foundation - Bridges](#)
- [Linux Foundation - VLANs](#)
- [Linux Journal - Linux as an Ethernet Bridge](#)
- [Comparing Traditional Bridge Mode to VLAN-aware Bridge Mode](#)

VLAN-aware Bridge Mode for Large-scale Layer 2 Environments

The Cumulus Linux bridge driver supports two configuration modes, one that is VLAN-aware, and one that follows a more traditional Linux bridge model.

For [traditional Linux bridges \(see page 288\)](#), the kernel supports VLANs in the form of VLAN subinterfaces. Enabling bridging on multiple VLANs means configuring a bridge for each VLAN and, for each member port on a bridge, creating one or more VLAN subinterfaces out of that port. This mode poses scalability challenges in terms of configuration size as well as boot time and run time state management, when the number of ports times the number of VLANs becomes large.

The VLAN-aware mode in Cumulus Linux implements a configuration model for large-scale L2 environments, with **one single instance** of [Spanning Tree \(see page 237\)](#). Each physical bridge member port is configured with the list of allowed VLANs as well as its port VLAN ID (either PVID or native VLAN — see below). MAC address learning, filtering and forwarding are *VLAN-aware*. This significantly reduces the configuration size, and eliminates the large overhead of managing the port/VLAN instances as subinterfaces, replacing them with lightweight VLAN bitmaps and state updates.



You can configure both VLAN-aware and traditional mode bridges on the same network in Cumulus Linux; however you should not have more than one VLAN-aware bridge on a given switch.

Contents

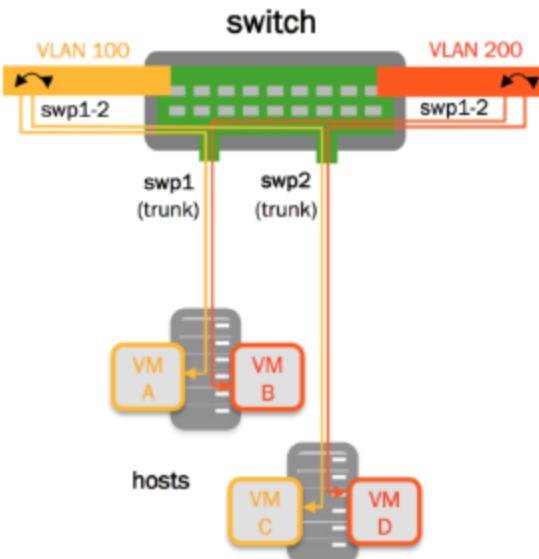
This chapter covers ...

- [Configuring a VLAN-aware Bridge \(see page 278\)](#)
- [Example Configurations \(see page 280\)](#)
 - [VLAN Filtering/VLAN Pruning \(see page 280\)](#)
 - [Untagged/Access Ports \(see page 280\)](#)
 - [Dropping Untagged Frames \(see page 281\)](#)
 - [VLAN Layer 3 Addressing — Switch Virtual Interfaces and Other VLAN Attributes \(see page 282\)](#)
 - [Configuring Multiple Ports in a Range \(see page 283\)](#)
 - [Access Ports and Pruned VLANs \(see page 284\)](#)
 - [Large Bond Set Configuration \(see page 285\)](#)
 - [VXLANs with VLAN-aware Bridges \(see page 286\)](#)
 - [Configuring a Static MAC Address Entry \(see page 287\)](#)
- [Caveats and Errata \(see page 287\)](#)

Configuring a VLAN-aware Bridge

VLAN-aware bridges can be configured with the Network Command Line Utility ([NCLU \(see page 80\)](#)). The example below shows the NCLU commands required to create a VLAN-aware bridge configured for STP, that contains two switch ports, and includes 3 VLANs — the tagged VLANs 100 and 200 and the untagged (native) VLAN of 1:

```
cumulus@switch:~$ net add bridge  
bridge ports swp1-2
```



```
cumulus@switch:~$ net add bridge
bridge vids 100,200
cumulus@switch:~$ net add bridge
bridge pvid 1
cumulus@switch:~$ net pending
cumulus@switch:~$ net commit
cumulus@switch:~$ net show
configuration files
```

...

```
auto bridge
iface bridge
    bridge-ports swp1 swp2
    bridge-pvid 1
    bridge-vids 100 200
    bridge-vlan-aware yes
```

...

The following attributes are useful for configuring VLAN-aware bridges:

- **bridge-vlan-aware**: Is automatically set to yes to indicate that the bridge is in VLAN-aware mode.
- **bridge-pvid**: A PVID is the *Primary VLAN Identifier*. The PVID defaults to 1; specifying the PVID identifies that VLAN as the native VLAN.
- **bridge-vids**: A VID is the *VLAN Identifier*, which declares the VLANs associated with this bridge.
- **bridge-access**: Declares the physical switch port as an *access port*. Access ports ignore all tagged packets; put all untagged packets into the **bridge-pvid**.
- **bridge-allow-untagged**: When set to *no*, it drops any untagged frames for a given switch port.

For a definitive list of bridge attributes, run `ifquery --syntax-help` and look for the entries under **bridge**, **bridgevlan** and **mstptcl**.



The **bridge-pvid 1** is implied by default. You do not have to specify **bridge-pvid**. And while it does not hurt the configuration, it helps other users for readability.

The following configurations are identical to each other and the configuration above:

```
auto bridge
iface bridge
    bridge-ports
        swp1 swp2
    bridge-vids
        1 100 200
    bridge-vlan-
        aware yes
```

```
auto bridge
iface bridge
    bridge-ports
        swp1 swp2
    bridge-pvid 1
    bridge-vids
        1 100 200
    bridge-vlan-
        aware yes
```

```
auto bridge
iface bridge
    bridge-ports
        swp1 swp2
    bridge-vids
        100 200
    bridge-vlan-
        aware yes
```

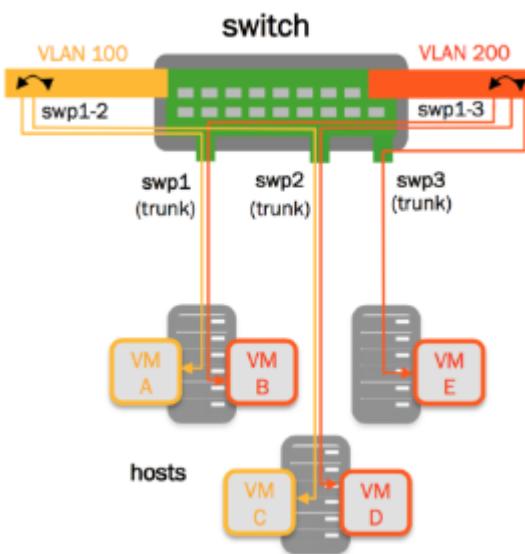


Do not try to bridge the management port, eth0, with any switch ports (like swp0, swp1, and so forth). For example, if you created a bridge with eth0 and swp1, it will not work properly and may disrupt access to the management interface.

Example Configurations

VLAN Filtering/VLAN Pruning

By default, the bridge port inherits the bridge VIDs. A port's configuration can override the bridge VIDs, by using the `bridge-vids` attribute:



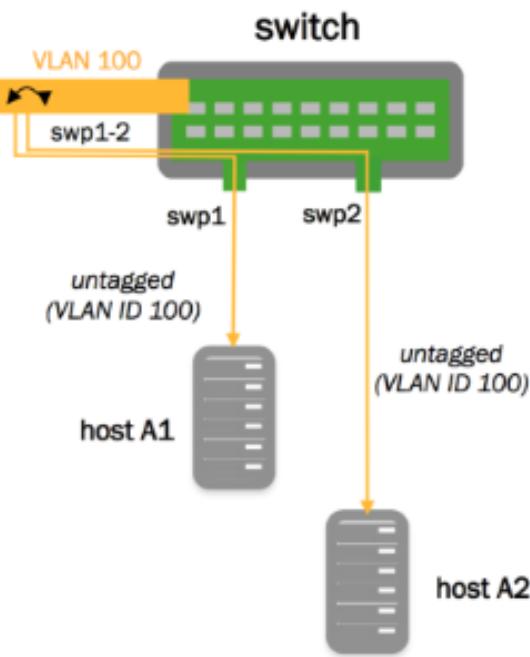
```
cumulus@switch:~$ net add bridge
bridge ports swp1-3
cumulus@switch:~$ net add bridge
bridge vids 100,200
cumulus@switch:~$ net add bridge
bridge pvid 1
cumulus@switch:~$ net add
interface swp3 bridge vids 200
cumulus@switch:~$ net pending
cumulus@switch:~$ net commit
cumulus@switch:~$ net show
configuration files

...
auto bridge
iface bridge
    bridge-ports swp1 swp2 swp3
    bridge-pvid 1
    bridge-vids 100 200
    bridge-vlan-aware yes

auto swp3
iface swp3
    bridge-vids 200
```

Untagged/Access Ports

Access ports ignore all tagged packets. In the configuration below, swp1 and swp2 are configured as access ports, while all untagged traffic goes to VLAN 100, as specified in the example below:



```

cumulus@switch:~$ net add bridge
bridge ports swp1-2
cumulus@switch:~$ net add bridge
bridge vids 100,200
cumulus@switch:~$ net add bridge
bridge pvid 1
cumulus@switch:~$ net add
interface swp1 bridge access 100
cumulus@switch:~$ net add
interface swp2 bridge access 100
cumulus@switch:~$ net pending
cumulus@switch:~$ net commit
cumulus@switch:~$ net show
configuration files

...
auto bridge
iface bridge
    bridge-ports swp1 swp2
    bridge-pvid 1
    bridge-vids 100 200
    bridge-vlan-aware yes

auto swp1
iface swp1
    bridge-access 100

auto swp2
iface swp2
    bridge-access 100
...

```

Dropping Untagged Frames

With VLAN-aware bridge mode, a switch port can be configured to drop any untagged frames. To do this, add `bridge-allow-untagged no` to the switch port. This leaves the bridge port without a PVID and drops untagged packets.

Consider the following example bridge:

```

auto bridge
iface bridge
    bridge-ports swp1 swp2
    bridge-pvid 1
    bridge-vids 10 100 100
    bridge-vlan-aware yes

```



Here is the VLAN membership for that configuration:

```
cumulus@switch:~$ net show bridge vlan

Interface      VLAN  Flags
-----  -----
swp1           1     PVID, Egress Untagged
              100
              200
swp2           1     PVID, Egress Untagged
              10
              100
              200
bridge         10
```

To configure swp2 to drop untagged frames, add `bridge-allow-untagged no`:

```
cumulus@switch:~$ net add interface swp2 bridge allow-untagged no
```

When you check VLAN membership for that port, it shows that there is **no** untagged VLAN.

```
cumulus@switch:~$ net show bridge vlan

Interface      VLAN  Flags
-----  -----
swp1           1     PVID, Egress Untagged
              100
              200
swp2           10
              100
              200
bridge         10
```

VLAN Layer 3 Addressing – Switch Virtual Interfaces and Other VLAN Attributes

When configuring the VLAN attributes for the bridge, specify the attributes for each VLAN interface, each of which is named `vlan<vlanid>`. If you are configuring the SVI for the native VLAN, you must declare the native VLAN and specify its IP address. Specifying the IP address in the bridge stanza itself returns an error.

```
cumulus@switch:~$ net add vlan 100 ip address 192.168.10.1/24
cumulus@switch:~$ net add vlan 100 ipv6 address 2001:db8::1/32
cumulus@switch:~$ net pending
cumulus@switch:~$ net commit
```



These commands create the following configuration in the /etc/network/interfaces file:

```
auto bridge
iface bridge
    bridge-ports swp1 swp2
    bridge-pvid 1
    bridge-vids 10 100 200
    bridge-vlan-aware yes

auto vlan100
iface vlan100
    address 192.168.10.1/24
    address 2001:db8::1/32
    vlan-id 100
    vlan-raw-device bridge
```



In the above configuration, if your switch is configured for multicast routing, you do not need to specify `bridge-igmp-querier-src`, as there is no need for a static IGMP querier configuration on the switch. Otherwise, the static IGMP querier configuration helps to probe the hosts to refresh their IGMP reports.

You can specify a range of VLANs as well. For example:

```
cumulus@switch:~$ net add vlan 1-2000
```

Configuring Multiple Ports in a Range

The `bridge-ports` attribute takes a range of numbers. The "swp1-52" in the example below indicates that swp1 through swp52 are part of the bridge, which is a shortcut that saves you from enumerating each port individually:

```
cumulus@switch:~$ net add bridge bridge ports swp1-52
cumulus@switch:~$ net pending
cumulus@switch:~$ net commit
```

These commands create the following configuration in the /etc/network/interfaces file:

```
auto bridge
iface bridge
    bridge-ports swp1 swp2 swp3 ... swp51 swp52
    bridge-vids 310 700 707 712 850 910
    bridge-vlan-aware yes
```

Access Ports and Pruned VLANs

The following example configuration contains an access port and switch port that are *pruned*; they only send and receive traffic tagged to/from a specific set of VLANs declared by the `bridge-vids` attribute. It also contains other switch ports that send and receive traffic from all the defined VLANs.

```
cumulus@switch:~$ net show configuration files

...
# ports swp3-swp48 are trunk ports which inherit vlans from the 'bridge'
# ie vlans 310,700,707,712,850,910
#
auto bridge
iface bridge
    bridge-ports swp1 swp2 swp3 ... swp51 swp52
    bridge-vids 310 700 707 712 850 910
    bridge-vlan-aware yes

auto swp1
iface swp1
    bridge-access 310
    mstpctl-bpduguard yes
    mstpctl-portadminedge yes

# The following is a trunk port that is "pruned".
# native vlan is 1, but only .1q tags of 707, 712, 850 are
# sent and received
#
auto swp2
iface swp2
    mstpctl-bpduguard yes
    mstpctl-portadminedge yes
    bridge-vids 707 712 850

# The following port is the trunk uplink and inherits all vlans
# from 'bridge'; bridge assurance is enabled using 'portnetwork'
attribute
auto swp49
iface swp49
    mstpctl-portnetwork yes
    mstpctl-portpathcost 10

# The following port is the trunk uplink and inherits all vlans
# from 'bridge'; bridge assurance is enabled using 'portnetwork'
attribute
auto swp50
iface swp50
    mstpctl-portnetwork yes
    mstpctl-portpathcost 0
```



...

Large Bond Set Configuration

The configuration below demonstrates a VLAN-aware bridge with a large set of bonds. The bond configurations are generated from a [Mako](#) template.

```
cumulus@switch:~$ net show configuration files

...
#
# vlan-aware bridge with bonds example
#
# uplink1, peerlink and downlink are bond interfaces.
# 'bridge' is a vlan aware bridge with ports uplink1, peerlink
# and downlink (swp2-20).
#
# native vlan is by default 1
#
# 'bridge-vids' attribute is used to declare vlans.
# 'bridge-pvid' attribute is used to specify native vlans if other
than 1
# 'bridge-access' attribute is used to declare access port
#
auto lo
iface lo

auto eth0
iface eth0 inet dhcp

# bond interface
auto uplink1
iface uplink1
    bond-slaves swp32
    bridge-vids 2000-2079

# bond interface
auto peerlink
iface peerlink
    bond-slaves swp30 swp31
    bridge-vids 2000-2079 4094

# bond interface
auto downlink
iface downlink
    bond-slaves swp1
    bridge-vids 2000-2079

#
# Declare vlans for all swp ports
```



```
# swp2-20 get vlans from 2004 to 2022.
# The below uses mako templates to generate iface sections
# with vlans for swp ports
#
%for port, vlanid in zip(range(2, 20), range(2004, 2022)) :
    auto swp${port}
    iface swp${port}
        bridge-vids ${vlanid}

%endfor

# svi vlan 2000
auto bridge.2000
iface bridge.2000
    address 11.100.1.252/24

# 12 attributes for vlan 2000
auto bridge.2000
vlan bridge.2000
    bridge-igmp-querier-src 172.16.101.1

#
# vlan-aware bridge
#
auto bridge
iface bridge
    bridge-ports uplink1 peerlink downlink swp1 swp2 swp49 swp50
    bridge-vlan-aware yes

# svi peerlink vlan
auto peerlink.4094
iface peerlink.4094
    address 192.168.10.1/30
    broadcast 192.168.10.3

...
```

VXLANs with VLAN-aware Bridges

Cumulus Linux supports using VXLANs with VLAN-aware bridge configuration. This provides improved scalability, as multiple VXLANs can be added to a single VLAN-aware bridge. A 1:1 association is used between the VXLAN VNI and the VLAN, using the bridge access VLAN definition on the VXLAN, and the VLAN membership definition on the local bridge member interfaces.

The configuration example below shows the differences between a VXLAN configured for traditional bridge mode and one configured for VLAN-aware mode. The configurations use head end replication (HER), along with the VLAN-aware bridge to map VLANs to VNIs.



The current tested scale limit for Cumulus Linux 3.2 is 512 VNIs.



```
cumulus@switch:~$ net show configuration files

...
auto lo
iface lo inet loopback
    address 10.35.0.10/32

auto bridge
iface bridge
    bridge-ports uplink regex vni.*
    bridge-pvid 1
    bridge-vids 1-100
    bridge-vlan-aware yes
auto vni-10000
iface vni-10000
    alias CUSTOMER X VLAN 10
    bridge-access 10
    vxlan-id 10000
    vxlan-local-tunnelip 10.35.0.10
    vxlan-remoteip 10.35.0.34

...
```

Configuring a Static MAC Address Entry

You can add a static MAC address entry to the layer 2 table for an interface within the VLAN-aware bridge by running a command similar to the following:

```
cumulus@switch:~$ sudo bridge fdb add 12:34:56:12:34:56 dev swp1 vlan
150 master static
cumulus@switch:~$ sudo bridge fdb show
44:38:39:00:00:7c dev swp1 master bridge permanent
12:34:56:12:34:56 dev swp1 vlan 150 master bridge static
44:38:39:00:00:7c dev swp1 self permanent
12:12:12:12:12:12 dev swp1 self permanent
12:34:12:34:12:34 dev swp1 self permanent
12:34:56:12:34:56 dev swp1 self permanent
12:34:12:34:12:34 dev bridge master bridge permanent
44:38:39:00:00:7c dev bridge vlan 500 master bridge permanent
12:12:12:12:12:12 dev bridge master bridge permanent
```

Caveats and Errata

- **Spanning Tree Protocol (STP):** VLAN-aware mode supports a single instance of STP across all VLANs, as STP is enabled on a per-bridge basis. A common practice when using a single STP instance for all VLANs is to define every VLAN on every switch in the spanning tree instance.

`mstpd` remains the user space protocol daemon.

Cumulus Linux supports Rapid Spanning Tree Protocol (RSTP).

- **IGMP Snooping:** IGMP snooping and group membership are supported on a per-VLAN basis, though the IGMP snooping configuration (including enable/disable and mrouter ports) are defined on a per-bridge port basis.
- **Reserved VLAN range:** For hardware data plane internal operations, the switching silicon requires VLANs for every physical port, Linux bridge, and layer 3 subinterface. Cumulus Linux reserves a range of 1000 VLANs by default; the reserved range is 3000-3999. The reserved range can be modified if it conflicts with any user-defined VLANs, as long the new range is a contiguous set of VLANs with IDs anywhere between 2 and 4094, and the minimum size of the range is 300 VLANs.

To configure the reserved range:

1. Open `/etc/cumulus/switchd.conf` in a text editor.
2. Uncomment the following line, specify a new range, and save the file:

```
resv_vlan_range
```

3. Restart `switchd` to implement the change:

```
cumulus@switch:~$ sudo systemctl restart switchd.service
```



While restarting `switchd`, all running ports will flap, and forwarding will be interrupted.

- **VLAN translation:** A bridge in VLAN-aware mode cannot have VLAN translation enabled for it. Only traditional mode bridges can utilize VLAN translation.
- **Converting bridges between supported modes:** Traditional mode bridges cannot be automatically converted to/from a VLAN-aware bridge. The original configuration must be deleted, and all member switch ports must be brought down, then a new bridge can be created.

Traditional Mode Bridges

Cumulus Networks recommends you use a [VLAN-aware bridge \(see page 277\)](#) on your switch. You use traditional mode bridges only if you need to run more than one bridge on the switch or if you need to use PVSTP+.

Contents

This chapter covers ...

- [Creating a Traditional Mode Bridge \(see page 288\)](#)
- [Using Trunks in Traditional Bridge Mode \(see page 290\)](#)
 - [Trunk Example \(see page 292\)](#)
 - [VLAN Tagging Examples \(see page 292\)](#)

Creating a Traditional Mode Bridge

You configure traditional mode bridges in `/etc/network/interfaces` file. To create a traditional mode bridge:

1. Open the `/etc/network/interfaces` file in a text editor.
2. Add a new stanza to create the bridge, and save the file. The example below creates a bridge with STP enabled and the MAC address ageing timer configured to a lower value than the default:

```
auto my_bridge
iface my_bridge
    bridge-ports bond0 swp5 swp6
    bridge-ageing 150
    bridge-stp on
```

Configuration Option	Description	Default Value
bridge-ports	List of logical and physical ports belonging to the logical bridge.	N/A
bridge-ageing	Maximum amount of time before a MAC addresses learned on the bridge expires from the bridge MAC cache.	300 seconds
bridge-stp	Enables spanning tree protocol on this bridge. The default spanning tree mode is Per VLAN Rapid Spanning Tree Protocol (PVRST). For more information on spanning-tree configurations see the configuration section: Spanning Tree and Rapid Spanning Tree (see page 237) .	off

⚠ The name of the bridge must be:

- Compliant with Linux interface naming conventions.
- Unique within the switch.

❗ Do not try to bridge the management port, eth0, with any switch ports (like swp0, swp1, and so forth). For example, if you created a bridge with eth0 and swp1, it will **not** work.

3. Reload the network configuration using the `ifreload` command:

```
cumulus@switch:~$ sudo ifreload -a
```



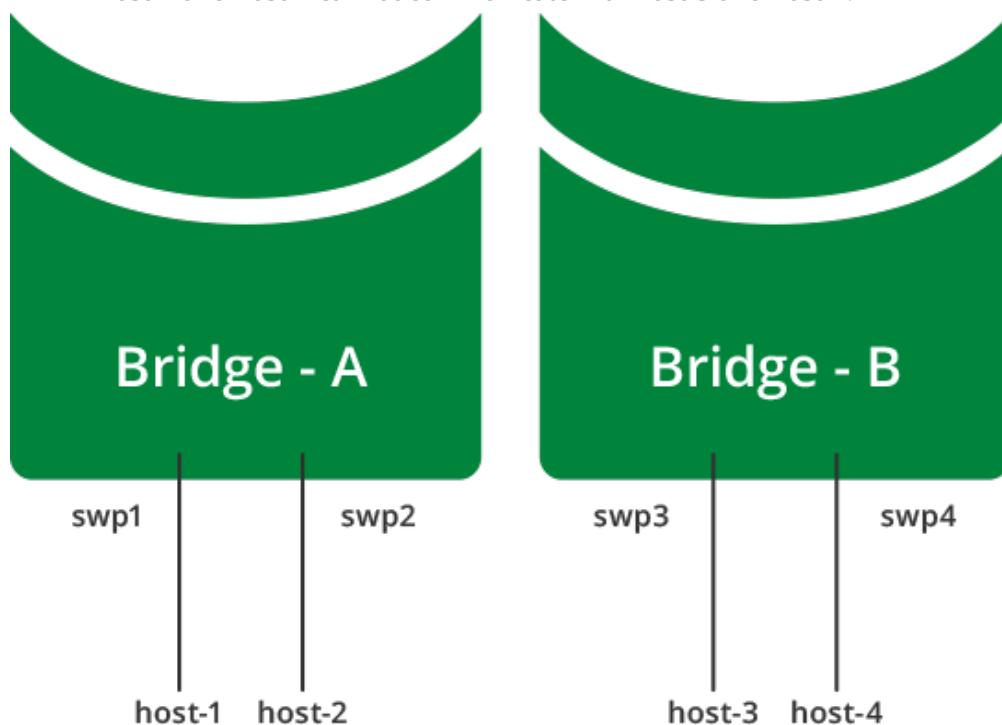
You can configure multiple bridges, in order to logically divide a switch into multiple layer 2 domains. This allows for hosts to communicate with other hosts in the same domain, while separating them from hosts in other domains.



You can create only one VLAN-aware bridge on a switch.

The diagram below shows a multiple bridge configuration, where host-1 and host-2 are connected to bridge-A, while host-3 and host-4 are connected to bridge-B. This means that:

- host-1 and host-2 can communicate with each other.
- host-3 and host-4 can communicate with each other.
- host-1 and host-2 cannot communicate with host-3 and host-4.



This example configuration looks like this in the `/etc/network/interfaces` file:

```

auto bridge-A
iface bridge-A
  bridge-ports swp1 swp2
  bridge-stp on

auto bridge-B
iface bridge-B
  bridge-ports swp3 swp4
  bridge-stp on

```



Using Trunks in Traditional Bridge Mode

The [IEEE standard](#) for trunking is 802.1Q. The 802.1Q specification adds a 4 byte header within the Ethernet frame that identifies the VLAN of which the frame is a member.

802.1Q also identifies an *untagged* frame as belonging to the *native VLAN* (most network devices default their native VLAN to 1). The concept of native, non-native, tagged or untagged has generated confusion due to mixed terminology and vendor-specific implementations. Some clarification is in order:

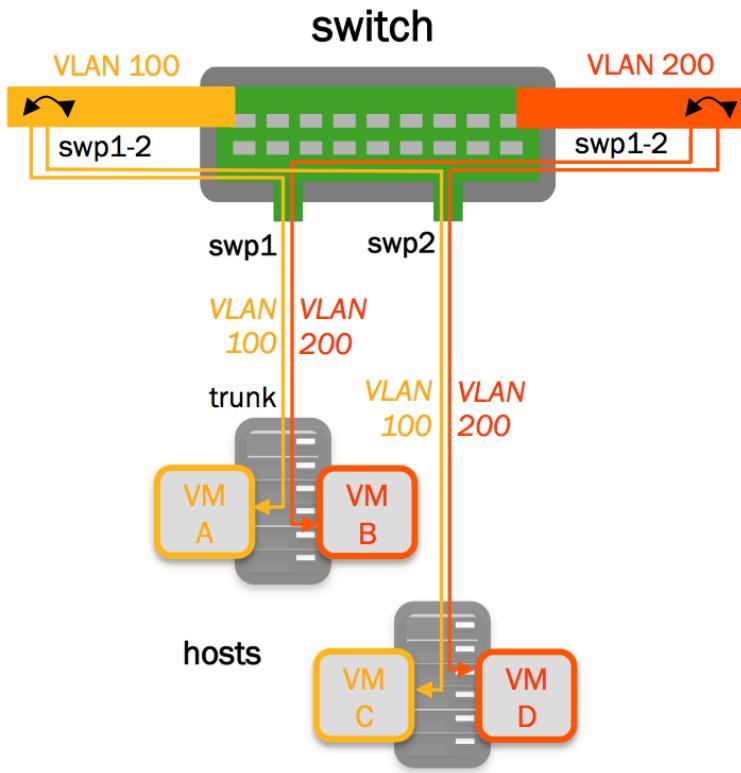
- A *trunk port* is a switch port configured to send and receive 802.1Q tagged frames.
- A switch sending an untagged (bare Ethernet) frame on a trunk port is sending from the native VLAN defined on the trunk port.
- A switch sending a tagged frame on a trunk port is sending to the VLAN identified by the 802.1Q tag.
- A switch receiving an untagged (bare Ethernet) frame on a trunk port places that frame in the native VLAN defined on the trunk port.
- A switch receiving a tagged frame on a trunk port places that frame in the VLAN identified by the 802.1Q tag.

A bridge in traditional mode has no concept of trunks, just tagged or untagged frames. With a trunk of 200 VLANs, there would need to be 199 bridges, each containing a tagged physical interface, and one bridge containing the native untagged VLAN. See the examples below for more information.



The interaction of tagged and un-tagged frames on the same trunk often leads to undesired and unexpected behavior. A switch that uses VLAN 1 for the native VLAN may send frames to a switch that uses VLAN 2 for the native VLAN, thus merging those two VLANs and their spanning tree state.

Trunk Example



To create the above example, add the following configuration to the `/etc/network/interfaces` file:

```

auto br-VLAN100
iface br-VLAN100
    bridge-ports swp1.100 swp2.100
    bridge-stp on

auto br-VLAN200
iface br-VLAN200
    bridge-ports swp1.200 swp2.200
    bridge-stp on

```

VLAN Tagging Examples

You can find more examples of VLAN tagging in [this chapter](#) (see page 292).

VLAN Tagging

This article shows two examples of VLAN tagging, one basic and one more advanced. They both demonstrate the streamlined interface configuration from `ifupdown2`.

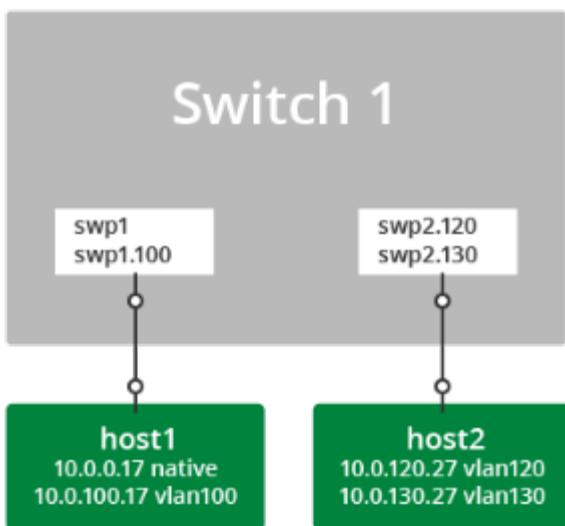
Contents

This chapter covers ...

- VLAN Tagging, a Basic Example (see page 293)
 - Persistent Configuration (see page 293)
- VLAN Tagging, an Advanced Example (see page 294)
 - Persistent Configuration (see page 295)
 - VLAN Translation (see page 299)

VLAN Tagging, a Basic Example

A simple configuration demonstrating VLAN tagging involves two hosts connected to a switch.



- *host1* connects to *swp1* with both untagged frames and with 802.1Q frames tagged for *vlan100*.
- *host2* connects to *swp2* with 802.1Q frames tagged for *vlan120* and *vlan130*.

Persistent Configuration

To configure the above example persistently, edit `/etc/network/interfaces` like this:

```
# Config for host1

auto swp1
iface swp1

auto swp1.100
iface swp1.100

# Config for host2
# swp2 must exist to create the .1Q subinterfaces, but it is not
assigned an address
```

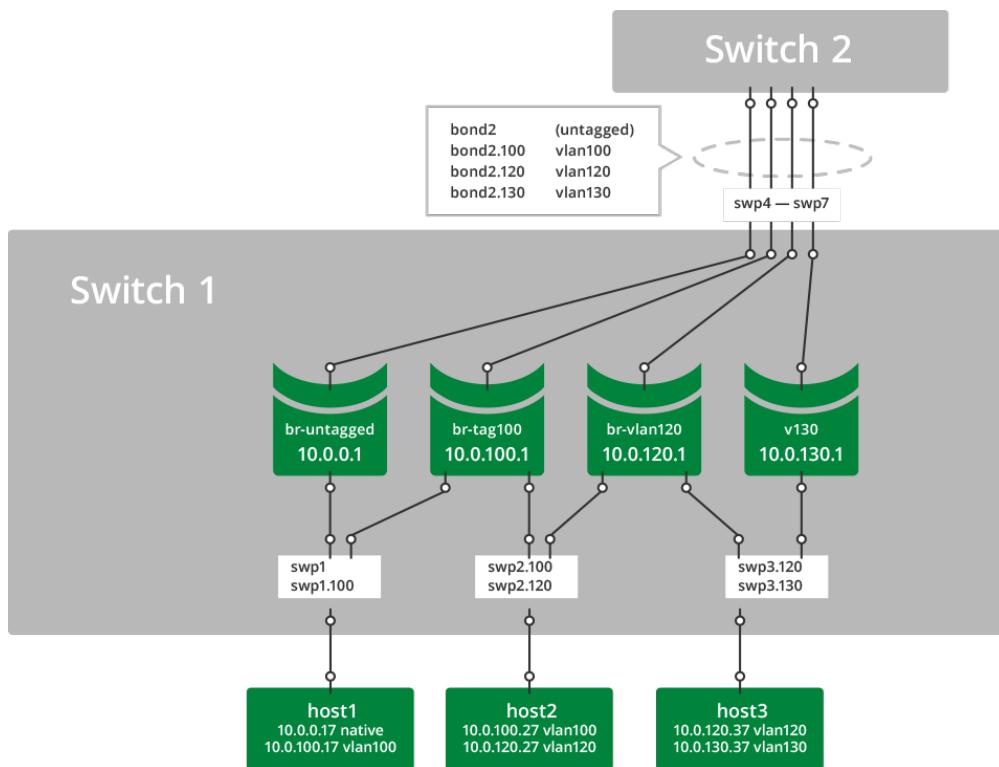
```
auto swp2
iface swp2

auto swp2.120
iface swp2.120

auto swp2.130
iface swp2.130
```

VLAN Tagging, an Advanced Example

This example of VLAN tagging is more complex, involving three hosts and two switches, with a number of bridges and a bond connecting them all.



- *host1* connects to bridge *br-untagged* with bare Ethernet frames and to bridge *br-tag100* with 802.1q frames tagged for *vlan100*.
- *host2* connects to bridge *br-tag100* with 802.1q frames tagged for *vlan100* and to bridge *br-vlan120* with 802.1q frames tagged for *vlan120*.
- *host3* connects to bridge *br-vlan120* with 802.1q frames tagged for *vlan120* and to bridge *v130* with 802.1q frames tagged for *vlan130*.
- *bond2* carries tagged and untagged frames in this example.

Although not explicitly designated, the bridge member ports function as 802.1Q *access ports* and *trunk ports*. In the example above, comparing Cumulus Linux with a traditional Cisco device:

- *swp1* is equivalent to a trunk port with untagged and *vlan100*.
- *swp2* is equivalent to a trunk port with *vlan100* and *vlan120*.
- *swp3* is equivalent to a trunk port with *vlan120* and *vlan130*.



- *bond2* is equivalent to an EtherChannel in trunk mode with untagged, *vlan100*, *vlan120*, and *vlan130*.
- Bridges *br-untagged*, *br-tag100*, *br-vlan120*, and *v130* are equivalent to SVIs (switched virtual interfaces).

Persistent Configuration

From /etc/network/interfaces:

```
# Config for host1 - - - - -
- - - - -
# swp1 does not need an iface section unless it has a specific
setting,
# it will be picked up as a dependent of swp1.100.
# And swp1 must exist in the system to create the .1q subinterfaces..
# but it is not applied to any bridge..or assigned an address.

auto swp1.100
iface swp1.100

# Config for host2
# swp2 does not need an iface section unless it has a specific
setting,
# it will be picked up as a dependent of swp2.100 and swp2.120.
# And swp2 must exist in the system to create the .1q subinterfaces..
# but it is not applied to any bridge..or assigned an address.

auto swp2.100
iface swp2.100

auto swp2.120
iface swp2.120

# Config for host3
# swp3 does not need an iface section unless it has a specific
setting,
# it will be picked up as a dependent of swp3.120 and swp3.130.
# And swp3 must exist in the system to create the .1q subinterfaces..
# but it is not applied to any bridge..or assigned an address.

auto swp3.120
iface swp3.120

auto swp3.130
iface swp3.130

# Configure the bond - - - - -
- - - - -
auto bond2
```



```
iface bond2
    bond-slaves glob swp4-7

# configure the bridges - - - - -
- - - - -

auto br-untagged
iface br-untagged
    address 10.0.0.1/24
    bridge-ports swp1 bond2
    bridge-stp on

auto br-tag100
iface br-tag100
    address 10.0.100.1/24
    bridge-ports swp1.100 swp2.100 bond2.100
    bridge-stp on

auto br-vlan120
iface br-vlan120
    address 10.0.120.1/24
    bridge-ports swp2.120 swp3.120 bond2.120
    bridge-stp on

auto v130
iface v130
    address 10.0.130.1/24
    bridge-ports swp3.130 bond2.130
    bridge-stp on

# - - - - -
```

To verify:

```
cumulus@switch:~$ sudo mstpcctl showbridge br-tag100
br-tag100 CIST info
  enabled      yes
  bridge id    8.000.44:38:39:00:32:8B
  designated root 8.000.44:38:39:00:32:8B
  regional root 8.000.44:38:39:00:32:8B
  root port    none
  path cost    0          internal path cost  0
  max age      20         bridge max age     20
  forward delay 15        bridge forward delay 15
  tx hold count 6          max hops           20
  hello time    2          ageing time       300
  force protocol version   rstp
  time since topology change 333040s
  topology change count    1
  topology change          no
```



```
topology change port      swp2.100
last topology change port None
```

```
cumulus@switch:~$ sudo mstptctl showportdetail br-tag100 | grep -B 2
state
br-tag100:bond2.100 CIST info
    enabled          yes           role
Designated
    port id        8.003         state
forwarding
--
br-tag100:swp1.100 CIST info
    enabled          yes           role
Designated
    port id        8.001         state
forwarding
--
br-tag100:swp2.100 CIST info
    enabled          yes           role
Designated
    port id        8.002         state
forwarding
```

```
cumulus@switch:~$ cat /proc/net/vlan/config
VLAN Dev name | VLAN ID
Name-Type: VLAN_NAME_TYPE_RAW_PLUS_VID_NO_PAD
bond2.100      | 100   | bond2
bond2.120      | 120   | bond2
bond2.130      | 130   | bond2
swp1.100       | 100   | swp1
swp2.100       | 100   | swp2
swp2.120       | 120   | swp2
swp3.120       | 120   | swp3
swp3.130       | 130   | swp3
```

```
cumulus@switch:~$ cat /proc/net/bonding/bond2
Ethernet Channel Bonding Driver: v3.7.1 (April 27, 2011)

Bonding Mode: IEEE 802.3ad Dynamic link aggregation
Transmit Hash Policy: layer3+4 (1)
MII Status: up
MII Polling Interval (ms): 100
Up Delay (ms): 0
Down Delay (ms): 0

802.3ad info
```



```
LACP rate: fast
Min links: 0
Aggregator selection policy (ad_select): stable
Active Aggregator Info:
    Aggregator ID: 3
    Number of ports: 4
    Actor Key: 33
    Partner Key: 33
    Partner Mac Address: 44:38:39:00:32:cf

Slave Interface: swp4
MII Status: up
Speed: 10000 Mbps
Duplex: full
Link Failure Count: 0
Permanent HW addr: 44:38:39:00:32:8e
Aggregator ID: 3
Slave queue ID: 0

Slave Interface: swp5
MII Status: up
Speed: 10000 Mbps
Duplex: full
Link Failure Count: 0
Permanent HW addr: 44:38:39:00:32:8f
Aggregator ID: 3
Slave queue ID: 0

Slave Interface: swp6
MII Status: up
Speed: 10000 Mbps
Duplex: full
Link Failure Count: 0
Permanent HW addr: 44:38:39:00:32:90
Aggregator ID: 3
Slave queue ID: 0

Slave Interface: swp7
MII Status: up
Speed: 10000 Mbps
Duplex: full
Link Failure Count: 0
Permanent HW addr: 44:38:39:00:32:91
Aggregator ID: 3
Slave queue ID: 0
```



A single bridge cannot contain multiple subinterfaces of the **same** port as members. Attempting to apply such a configuration will result in an error:



```
cumulus@switch:~$ sudo brctl addbr another_bridge
cumulus@switch:~$ sudo brctl addif another_bridge swp9 swp9.100
0
bridge cannot contain multiple subinterfaces of the same port:
swp9, swp9.100
```

VLAN Translation

By default, Cumulus Linux does not allow VLAN subinterfaces associated with different VLAN IDs to be part of the same bridge. Base interfaces are not explicitly associated with any VLAN IDs and are exempt from this restriction:

```
cumulus@switch:~$ sudo brctl addbr br_mix

cumulus@switch:~$ sudo ip link add link swp10 name swp10.100 type
vlan id 100
cumulus@switch:~$ sudo ip link add link swp11 name swp11.200 type
vlan id 200

cumulus@switch:~$ sudo brctl addif br_mix swp10.100 swp11.200
can't add swp11.200 to bridge br_mix: Invalid argument
```

In some cases, it may be useful to relax this restriction. For example, two servers may be connected to the switch using VLAN trunks, but the VLAN numbering provisioned on the two servers are not consistent. You can choose to just bridge two VLAN subinterfaces of different VLAN IDs from the servers. You do this by enabling the `sysctl net.bridge.bridge-allow-multiple-vlans`. Packets entering a bridge from a member VLAN subinterface will egress another member VLAN subinterface with the VLAN ID translated.



A bridge in [VLAN-aware mode](#) (see page 277) cannot have VLAN translation enabled for it; only bridges configured in traditional mode can utilize VLAN translation.

The following example enables the VLAN translation `sysctl`:

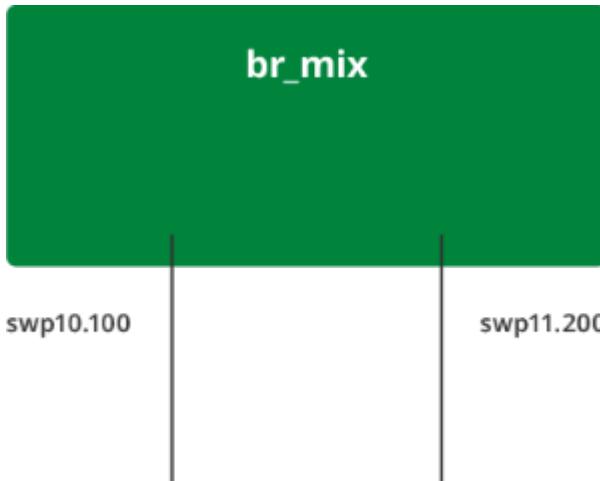
```
cumulus@switch:~$ echo net.bridge.bridge-allow-multiple-vlans = 1 | 
sudo tee /etc/sysctl.d/multiple_vlans.conf
net.bridge.bridge-allow-multiple-vlans = 1
cumulus@switch:~$ sudo sysctl -p /etc/sysctl.d/multiple_vlans.conf
net.bridge.bridge-allow-multiple-vlans = 1
```

If the `sysctl` is enabled and you want to disable it, run the above example, setting the `sysctl net.bridge.bridge-allow-multiple-vlans` to 0.

Once the `sysctl` is enabled, ports with different VLAN IDs can be added to the same bridge. In the following example, packets entering the bridge `br-mix` from `swp10.100` will be bridged to `swp11.200` with the VLAN ID translated from 100 to 200:

```
cumulus@switch:~$ sudo brctl addif br_mix swp10.100 swp11.200
```

```
cumulus@switch:~$ sudo brctl show br_mix
bridge name      bridge id          STP enabled    interfaces
br_mix          8000.4438390032bd    yes           swp10.100
                                         swp11.200
```



Multi-Chassis Link Aggregation - MLAG

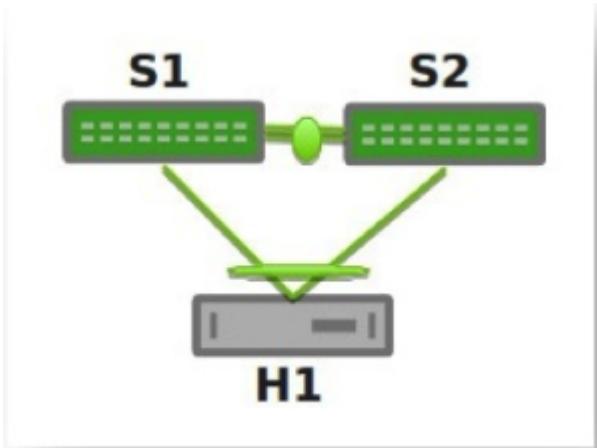
Multi-Chassis Link Aggregation, or MLAG, enables a server or switch with a two-port bond (such as a link aggregation group/LAG, EtherChannel, port group or trunk) to connect those ports to different switches and operate as if they are connected to a single, logical switch. This provides greater redundancy and greater system throughput.

i MLAG or CLAG?

The Cumulus Linux implementation of MLAG is referred to by other vendors as CLAG, MC-LAG or VPC. You will even see references to CLAG in Cumulus Linux, including the management daemon, named `c1agd`, and other options in the code, such as `c1ag-id`, which exist for historical purposes. But rest assured that the Cumulus Linux implementation is truly a multi-chassis link aggregation protocol, so we call it MLAG.

Dual-connected devices can create LACP bonds that contain links to each physical switch. Thus, active-active links from the dual-connected devices are supported even though they are connected to two different physical switches.

A basic setup looks like this:



You can see an example of how to set up this configuration by running:

```
cumulus@switch:~$ net example clag basic-clag
```

The two switches, S1 and S2, known as *peer switches*, cooperate so that they appear as a single device to host H1's bond. H1 distributes traffic between the two links to S1 and S2 in any manner that you configure on the host. Similarly, traffic inbound to H1 can traverse S1 or S2 and arrive at H1.

Contents

This chapter covers ...

- MLAG Requirements (see page 302)
- LACP and Dual-Connectedness (see page 303)
- Configuring MLAG (see page 304)
 - Reserved MAC Address Range (see page 307)
 - Configuring the Host or Switch (see page 307)
 - Configuring the Interfaces (see page 307)
 - Understanding Switch Roles and Setting Priority (see page 308)
- Example MLAG Configuration (see page 309)
 - Disabling clagd on an Interface (see page 320)
- Checking the MLAG Configuration Status (see page 320)
 - Using the clagd Command Line Interface (see page 321)
- Configuring MLAG with a Traditional Mode Bridge (see page 321)
- Peer Link Interfaces and the protodown State (see page 322)
 - Specifying a Backup Link (see page 322)
- Monitoring Dual-Connected Peers (see page 325)
- Configuring Layer 3 Routed Uplinks (see page 326)

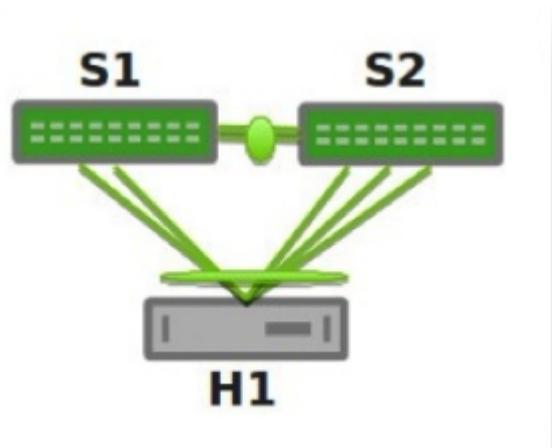
- IGMP Snooping with MLAG (see page 327)
- Monitoring the Status of the clagd Service (see page 327)
- MLAG Best Practices (see page 328)
 - Understanding MTU in an MLAG Configuration (see page 329)
 - Sizing the Peerlink (see page 330)
- STP Interoperability with MLAG (see page 331)
 - Best Practices for STP with MLAG (see page 332)
- Troubleshooting MLAG (see page 332)
 - Viewing the MLAG Log File (see page 332)
 - Large Packet Drops on the Peerlink Interface (see page 332)
- Caveats and Errata (see page 333)

MLAG Requirements

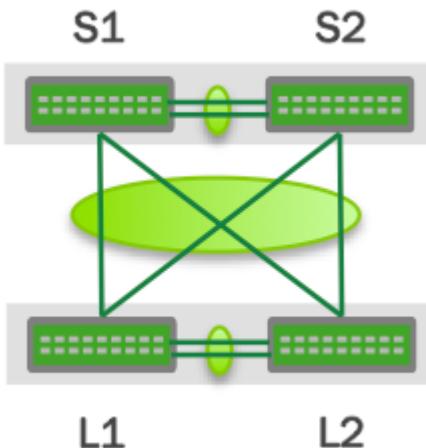
MLAG has these requirements:

- There must be a direct connection between the two peer switches implementing MLAG (S1 and S2). This is typically a bond for increased reliability and bandwidth.
- There must be only two peer switches in one MLAG configuration, but you can have multiple configurations in a network for *switch-to-switch MLAG* (see below).
- The peer switches implementing MLAG must be running Cumulus Linux version 2.5 or later.
- You must specify a unique `clag-id` for every dual-connected bond on each peer switch; the value must be between 1 and 65535 and must be the same on both peer switches in order for the bond to be considered *dual-connected*.
- The dual-connected devices (servers or switches) must use LACP (IEEE 802.3ad/802.1ax) to form the bond. The peer switches must also use LACP.

More elaborate configurations are also possible. The number of links between the host and the switches can be greater than two, and does not have to be symmetrical:



Additionally, since S1 and S2 appear as a single switch to other bonding devices, pairs of MLAG switches can also be connected to each other in a switch-to-switch MLAG setup:

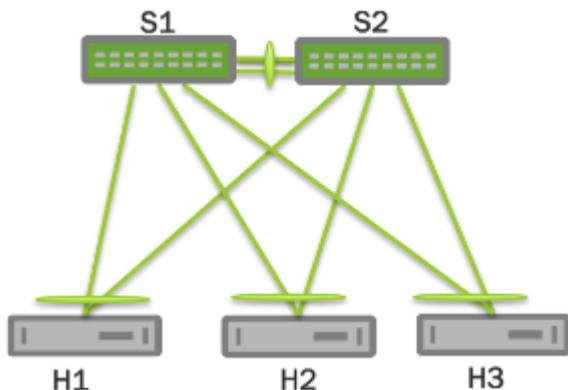


In this case, L1 and L2 are also MLAG peer switches, and thus present a two-port bond from a single logical system to S1 and S2. S1 and S2 do the same as far as L1 and L2 are concerned. For a switch-to-switch MLAG configuration, each switch pair must have a unique system MAC address. In the above example, switches L1 and L2 each have the same system MAC address configured. Switch pair S1 and S2 each have the same system MAC address configured; however, it is a different system MAC address than the one used by the switch pair L1 and L2.

LACP and Dual-Connectedness

In order for MLAG to operate correctly, the peer switches must know which links are *dual-connected*, or are connected to the same host or switch. To do this, specify a `cflag-id` for every dual-connected bond on each peer switch; the `cflag-id` must be the same for the corresponding bonds on both peer switches. [Link Aggregation Control Protocol \(LACP\)](#), the IEEE standard protocol for managing bonds, is used for verifying dual-connectedness. LACP runs on the dual-connected device and on each of the peer switches. On the dual-connected device, the only configuration requirement is to create a bond that will be managed by LACP.

On each of the peer switches, the links connected to the dual-connected host or switch must be placed in the bond. This is true even if the links are a single port on each peer switch, where each port is placed into a bond, as shown below:



All of the dual-connected bonds on the peer switches have their system ID set to the MLAG system ID. Therefore, from the point of view of the hosts, each of the links in its bond is connected to the same system, and so the host will use both links.

Each peer switch periodically makes a list of the LACP partner MAC addresses for all of their bonds and sends that list to its peer (using the `clagd` service; see below). The LACP partner MAC address is the MAC address of the system at the other end of a bond, which in the figure above would be hosts H1, H2 and H3. When a switch receives this list from its peer, it compares the list to the LACP partner MAC addresses on its switch. If any matches are found and the `clag-id` for those bonds match, then that bond is a dual-connected bond. You can also find the LACP partner MAC address by running `net show bridge macs`, or by looking in the `/sys/class/net/<bondname>/bonding/ad_partner_mac sysfs` file for each bond.

Configuring MLAG

Configuring MLAG involves:

- On the dual-connected devices, creating a bond that uses LACP.
- On each peer switch, configuring the interfaces, including bonds, VLANs, bridges and peer links.

Keep MLAG Configurations in Sync

MLAG synchronizes the dynamic state between the two peer switches, but it does not synchronize the switch configurations. After modifying the configuration of one peer switch, you must make the same changes to the configuration on the other peer switch. This applies to all configuration changes, including:

- Port configuration: For example, VLAN membership, [MTU \(see page 329\)](#) and bonding parameters.
- Bridge configuration: For example, spanning tree parameters or bridge properties.
- Static address entries: For example, static FDB entries and static IGMP entries.
- QoS configuration: For example, ACL entries.

You can verify the configuration of VLAN membership using the `net show clag verify-vlans` command.

Click to see the output ...

```
cumulus@leaf01:~$ net show clag verify-vlans
Our Bond Interface    VlanId    Peer Bond Interface
-----      -----
server01           1        server01
server01           10       server01
server01          20        server01
server01          30        server01
server01          40        server01
server01          50        server01
uplink            1        uplink
uplink            10       uplink
uplink            20       uplink
uplink            30       uplink
uplink            40       uplink
uplink            50       uplink
uplink           100      uplink
uplink           101      uplink
```



uplink	102	uplink
uplink	103	uplink
uplink	104	uplink
uplink	105	uplink
uplink	106	uplink
uplink	107	uplink
uplink	108	uplink
uplink	109	uplink
uplink	110	uplink
uplink	111	uplink
uplink	112	uplink
uplink	113	uplink
uplink	114	uplink
uplink	115	uplink
uplink	116	uplink
uplink	117	uplink
uplink	118	uplink
uplink	119	uplink
uplink	120	uplink
uplink	121	uplink
uplink	122	uplink
uplink	123	uplink
uplink	124	uplink
uplink	125	uplink
uplink	126	uplink
uplink	127	uplink
uplink	128	uplink
uplink	129	uplink
uplink	130	uplink
uplink	131	uplink
uplink	132	uplink
uplink	133	uplink
uplink	134	uplink
uplink	135	uplink
uplink	136	uplink
uplink	137	uplink
uplink	138	uplink
uplink	139	uplink
uplink	140	uplink
uplink	141	uplink
uplink	142	uplink
uplink	143	uplink
uplink	144	uplink
uplink	145	uplink
uplink	146	uplink
uplink	147	uplink
uplink	148	uplink
uplink	149	uplink
uplink	150	uplink
uplink	151	uplink
uplink	152	uplink
uplink	153	uplink

uplink	154	uplink
uplink	155	uplink
uplink	156	uplink
uplink	157	uplink
uplink	158	uplink
uplink	159	uplink
uplink	160	uplink
uplink	161	uplink
uplink	162	uplink
uplink	163	uplink
uplink	164	uplink
uplink	165	uplink
uplink	166	uplink
uplink	167	uplink
uplink	168	uplink
uplink	169	uplink
uplink	170	uplink
uplink	171	uplink
uplink	172	uplink
uplink	173	uplink
uplink	174	uplink
uplink	175	uplink
uplink	176	uplink
uplink	177	uplink
uplink	178	uplink
uplink	179	uplink
uplink	180	uplink
uplink	181	uplink
uplink	182	uplink
uplink	183	uplink
uplink	184	uplink
uplink	185	uplink
uplink	186	uplink
uplink	187	uplink
uplink	188	uplink
uplink	189	uplink
uplink	190	uplink
uplink	191	uplink
uplink	192	uplink
uplink	193	uplink
uplink	194	uplink
uplink	195	uplink
uplink	196	uplink
uplink	197	uplink
uplink	198	uplink
uplink	199	uplink
uplink	200	uplink

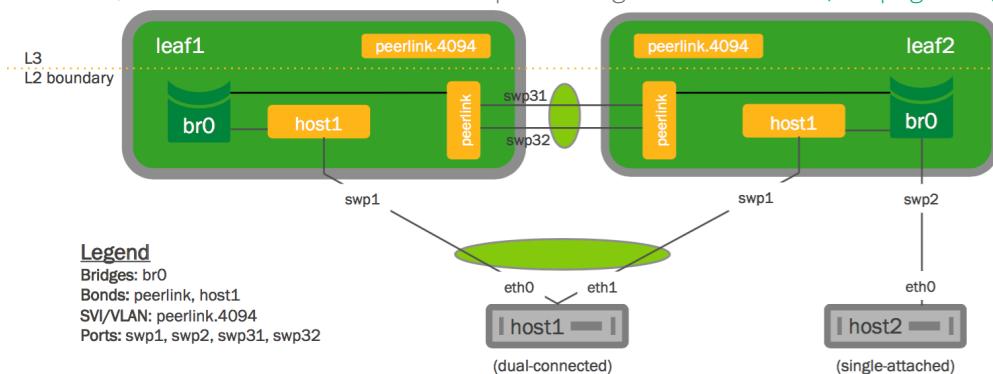
Reserved MAC Address Range

In order to prevent MAC address conflicts with other interfaces in the same bridged network, Cumulus Networks has [reserved a range of MAC addresses](#) specifically to use with MLAG. This range of MAC addresses is 44:38:39:ff:00:00 to 44:38:39:ff:ff:ff.

Cumulus Networks recommends you use this range of MAC addresses when configuring MLAG.

Configuring the Host or Switch

On your dual-connected device, create a bond that uses LACP. The method you use varies with the type of device you are configuring. The following image is a basic MLAG configuration, showing all the essential elements; a more detailed two-leaf/two-spine configuration is [below \(see page 309\)](#).



Configuring the Interfaces

Every interface that connects to the MLAG pair from a dual-connected device should be placed into a [bond](#) ([see page 268](#)), even if the bond contains only a single link on a single physical switch (even though the MLAG pair contains two or more links). Layer 2 data travels over this bond. In the examples throughout this chapter, *peerlink* is the name of the bond.

Single-attached hosts, also known as *orphan ports*, can be just a member of the bridge.

Additionally, the fast mode of LACP should be configured on the bond to allow more timely updates of the LACP state. These bonds are then placed in a bridge, which must include the peer link between the switches.

In order to enable communication between the `c1agd` services on the peer switches, you should choose an unused VLAN (also known as a *switched virtual interface* or *SVI* here) and assign an unrouteable link-local address to give the peer switches layer 3 connectivity between each other. To ensure that the VLAN is completely independent of the bridge and spanning tree forwarding decisions, configure the VLAN as a VLAN subinterface on the peer link bond rather than the VLAN-aware bridge. Cumulus Networks recommends you use 4094 for the peer link VLAN (*peerlink.4094* below) if possible. In addition, to avoid issues with STP, make sure you include untagged traffic on the peer link.

You can also specify a backup interface, which is any layer 3 backup interface for your peer links in the event that the peer link goes down. [See below \(see page 322\)](#) for more information about the backup link.

For example, if *peerlink* is the inter-chassis bond, and VLAN 4094 is the peer link VLAN, configure *peerlink.4094* as follows:

Configuring the peerlink Interface

```
cumulus@leaf01:~$ net add interface peerlink.4094 peerlink.4094
cumulus@leaf01:~$ net add interface peerlink.4094 address 169.2
54.1.1/30
cumulus@leaf01:~$ net add interface peerlink.4094 clagd-peer-
ip 169.254.1.2
cumulus@leaf01:~$ net add interface peerlink.4094 clagd-backup-
ip 192.0.2.50
cumulus@leaf01:~$ net add interface peerlink.4094 clagd-sys-
mac 44:38:39:FF:40:94
cumulus@leaf01:~$ net pending
cumulus@leaf01:~$ net commit
```

There is no need to add VLAN 4094 to the bridge VLAN list, as it is unnecessary there.

The above commands produce the following configuration in the /etc/network/interfaces file:

```
auto peerlink.4094
iface peerlink.4094
    address 169.254.1.1/30
    clagd-peer-ip 169.254.1.2
    clagd-backup-ip 192.0.2.50
    clagd-sys-mac 44:38:39:FF:40:94
```



Keep in mind that if you change the MLAG configuration by editing the `interfaces` file, the changes take effect when you bring the peer link interface up with `ifup`. Do **not** use `systemctl restart clagd.service` to apply the new configuration.

① Don't Use 169.254.0.1

Do not use 169.254.0.1 as the MLAG peerlink IP address, as Cumulus Linux uses this address exclusively for [BGP unnumbered \(see page 516\)](#) interfaces.

Understanding Switch Roles and Setting Priority

Each MLAG-enabled switch in the pair has a *role*. When the peering relationship is established between the two switches, one switch is put into the *primary* role, and the other one is put into the *secondary* role. When an MLAG-enabled switch is in the secondary role, it does not send STP BPDUs on dual-connected links; it only sends BPDUs on single-connected links. The switch in the primary role sends STP BPDUs on all single- and dual-connected links.

Sends BPDUs Via	Primary	Secondary
Single-connected links	Yes	Yes

Sends BPDUs Via	Primary	Secondary
Dual-connected links	Yes	No

By default, the role is determined by comparing the MAC addresses of the two sides of the peering link; the switch with the lower MAC address assumes the primary role. You can override this by setting the `clagd-priority` option for the peer link:

```
cumulus@leaf01:~$ net add interface peerlink.4094 clagd-priority 2048
cumulus@leaf01:~$ net pending
cumulus@leaf01:~$ net commit
```

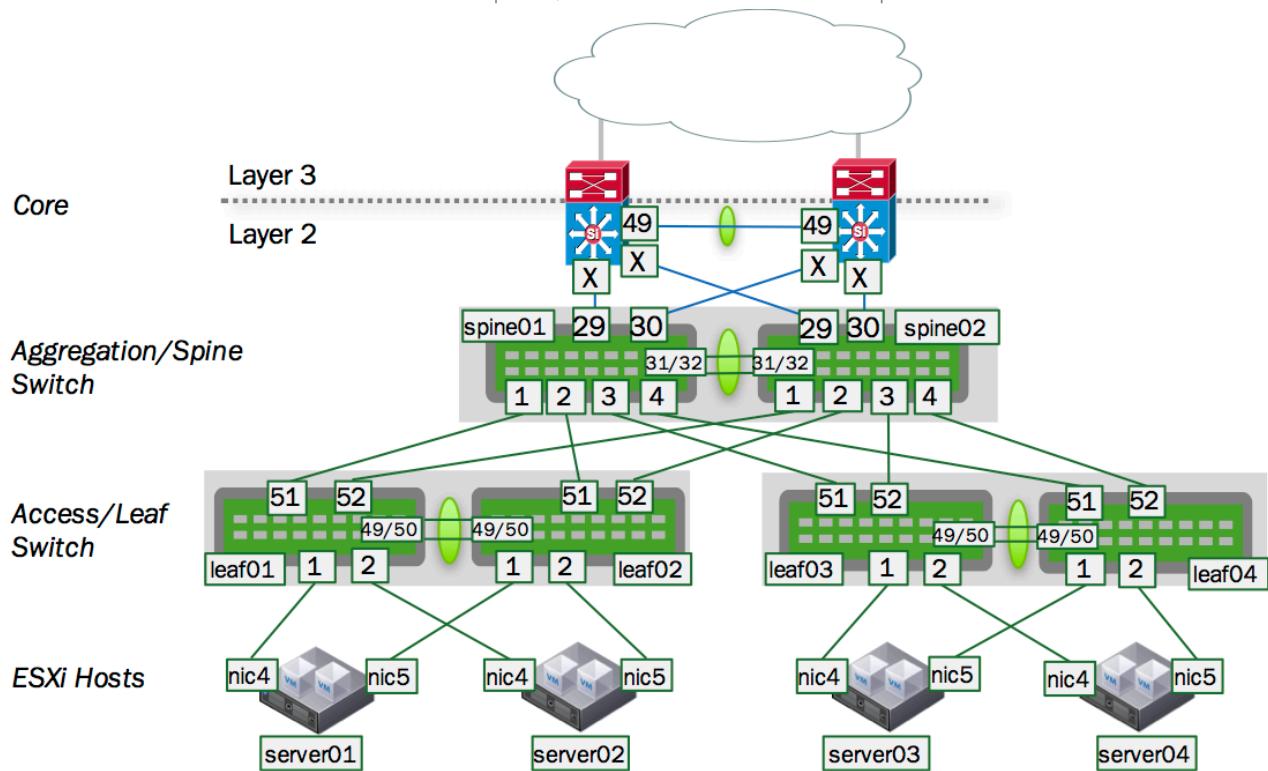
The switch with the lower priority value is given the primary role; the default value is 32768, and the range is 0 to 65535. Read the `clagd(8)` and `clagctl(8)` man pages for more information.

When the `clagd` service is exited during switch reboot or the service is stopped in the primary switch, the peer switch that is in the secondary role becomes the primary.

However, if the primary switch goes down without stopping the `clagd` service for any reason, or if the peer link goes down, the secondary switch will **not** change its role. In case the peer switch is determined to be not alive, the switch in the secondary role will roll back the LACP system ID to be the bond interface MAC address instead of the `clagd-sys-mac` and the switch in primary role uses the `clagd-sys-mac` as the LACP system ID on the bonds.

Example MLAG Configuration

An example configuration is included below. It configures two bonds for MLAG, each with a single port, a peer link that is a bond with two member ports, and three VLANs on each port.





You configure these interfaces using [NCLU \(see page 80\)](#), so the bridges are in [VLAN-aware mode \(see page 277\)](#). The bridges use these Cumulus Linux-specific keywords:

- `bridge-vids`, which defines the allowed list of tagged 802.1q VLAN IDs for all bridge member interfaces. You can specify non-contiguous ranges with a space-separated list, like
`bridge-vids 100-200 300 400-500.`
- `bridge-pvid`, which defines the untagged VLAN ID for each port. This is commonly referred to as the *native VLAN*.

The bridge configurations below indicate that each bond carries tagged frames on VLANs 10, 20, 30, 40, 50 and 100 to 200 (as specified by `bridge-vids`), but untagged frames on VLAN 1 (as specified by `bridge-pvid`). Also, take note on how you configure the VLAN subinterfaces used for `cldg` communication (`peerlink.4090` and `peerlink.4094` in the sample configuration below). Finally, the host configurations for server01 through server04 are not shown here. The configurations for each corresponding node are almost identical, except for the IP addresses used for managing the `cldg` service.

VLAN Precautions

At minimum, this VLAN subinterface should not be in your layer 2 domain, and you should give it a very high VLAN ID (up to 4094). Read more about the [range of VLAN IDs you can use \(see page 277\)](#).

The commands to create the configurations for both spines should look like the following. Note that the `cldg-id` and `cldg-sys-mac` must be the same for the corresponding bonds on spine01 and spine02:

spine01

```
cumulus@spine01:~$ net show
configuration commands
echo 'auto lo' > /etc/network
/interfaces
echo 'iface lo inet loopback' >>
/etc/network/interfaces
echo 'auto eth0' >> /etc/network
/interfaces
echo 'iface eth0 inet dhcp' >> /etc
/network/interfaces
echo 'zebra=yes' > /etc/quagga
/daemons
echo '' > /etc/quagga/Quagga.conf
net abort
net add hostname spine01
net add interface swp1-4,29-30 mtu
9216
net add bond exit01-02 bond slaves
swp29-30
net add bond leaf01-02 bond slaves
swp1-2
net add bond leaf03-04 bond slaves
swp3-4
```

spine02

```
cumulus@spine02:~$ net show
configuration commands
echo 'auto lo' > /etc
/network/interfaces
echo 'iface lo inet
loopback' >> /etc/network
/interfaces
echo 'auto eth0' >> /etc
/network/interfaces
echo 'iface eth0 inet dhcp'
>> /etc/network/interfaces
echo 'zebra=yes' > /etc
/quagga/daemons
echo '' > /etc/quagga
/Quagga.conf
net abort
net add hostname spine02
net add interface swp1-4,29-
30 mtu 9216
net add bond exit01-02 bond
slaves swp29-30
net add bond leaf01-02 bond
slaves swp1-2
```



```
net add bond peerlink bond slaves  
swp31-32  
net add loopback lo ip address  
10.0.0.21/32  
net add interface eth0 ip address  
dhcp  
net add bridge bridge ports leaf01-  
02,leaf03-04,exit01-02,peerlink  
net add bridge bridge pvid 1  
net add bridge bridge vids  
10,20,30,40,50,100-200  
net add bridge stp treeprio 4096  
net add vlan 10 ip address  
10.1.10.2/24  
net add vlan 10 ip address-virtual  
44:39:39:FF:00:10 10.1.10.1/24  
net add vlan 20 ip address  
10.1.20.2/24  
net add vlan 20 ip address-virtual  
44:39:39:FF:00:20 10.1.20.1/24  
net add vlan 10 alias 'VM 10  
Network'  
net add vlan 20 alias 'VM 20  
Network'  
net add bond exit01-02 clag id 2930  
net add bond exit01-02 mtu 9216  
net add bond leaf01-02 mtu 9216  
net add bond leaf03-04 mtu 9216  
net add bond leaf01-02 clag id 1012  
net add bond leaf03-04 clag id 1034  
net add interface peerlink.4090 ip  
address 169.254.255.1/30  
net add interface peerlink.4090  
clag backup-ip 192.168.0.22  
net add interface peerlink.4090  
clag enable yes  
net add interface peerlink.4090  
clag peer-ip 169.254.255.2  
net add interface peerlink.4090  
clag sys-mac 44:39:39:FF:40:90  
net add routing log file /var/log  
/quagga/quagga.log  
net add routing log timestamp  
precision 6
```

These commands create the following configuration ...

```
cumulus@spine01:~$ cat /etc/network  
/interfaces  
# The loopback network interface
```

```
net add bond leaf03-04 bond  
slaves swp3-4  
net add bond peerlink bond  
slaves swp31-32  
net add loopback lo ip  
address 10.0.0.22/32  
net add interface eth0 ip  
address dhcp  
net add bridge bridge ports  
leaf01-02,leaf03-04,exit01-0  
2,peerlink  
net add bridge bridge pvid 1  
net add bridge bridge vids 1  
0,20,30,40,50,100-200  
net add bridge stp treeprio  
4096  
net add vlan 10 ip address 1  
0.1.10.3/24  
net add vlan 10 ip address-  
virtual 44:39:39:FF:00:10 10  
.1.10.1/24  
net add vlan 20 ip address 1  
0.1.20.3/24  
net add vlan 20 ip address-  
virtual 44:39:39:FF:00:20 10  
.1.20.1/24  
net add vlan 10 alias 'VM  
10 Network'  
net add vlan 20 alias 'VM  
20 Network'  
net add bond exit01-02 clag  
id 2930  
net add bond exit01-02 mtu 9  
216  
net add bond leaf01-02 mtu 9  
216  
net add bond leaf03-04 mtu 9  
216  
net add bond leaf01-02 clag  
id 1012  
net add bond leaf03-04 clag  
id 1034  
net add interface peerlink.4  
090 ip address 169.254.255.2  
/30  
net add interface peerlink.4  
090 clag backup-ip 192.168.0  
.21  
net add interface peerlink.4  
090 clag enable yes
```

```

auto lo
iface lo inet loopback
  address 10.0.0.21/32
# The primary network interface
auto eth0
iface eth0 inet dhcp
auto swp1
iface swp1
  mtu 9216
auto swp2
iface swp2
  mtu 9216
auto swp3
iface swp3
  mtu 9216
auto swp4
iface swp4
  mtu 9216
auto swp29
iface swp29
  mtu 9216
auto swp30
iface swp30
  mtu 9216
auto swp31
iface swp31
auto swp32
iface swp32
auto bridge
iface bridge
  bridge-ports exit01-02 leaf01-02 leaf03-04 peerlink
  bridge-pvid 1
  bridge-vids 10 20 30 40 50 100-200
  bridge-vlan-aware yes
  mstptctl-treeprio 4096
auto exit01-02
iface exit01-02
  bond-slaves swp29 swp30
  clag-id 2930
  mtu 9216
auto leaf01-02
iface leaf01-02
  bond-slaves swp1 swp2
  clag-id 1012
  mtu 9216
auto leaf03-04
iface leaf03-04
  bond-slaves swp3 swp4
  clag-id 1034

```

```

net add interface peerlink.4
  090 clag peer-ip 169.254.255
  .1
net add interface peerlink.4
  090 clag sys-mac 44:39:39:FF:40:90
net add interface peerlink.4
  090 alias 'Spine01-Spine02
MLAG peerlink'
net add routing log file
/var/log/quagga/quagga.log
net add routing log
timestamp precision 6

```

These commands create the following configuration ...

```

cumulus@spine02:~$ cat /etc
/network/interfaces
# The loopback network
interface
auto lo
iface lo inet loopback
  address 10.0.0.22/32
# The primary network
interface
auto eth0
iface eth0 inet dhcp
auto swp1
iface swp1
  mtu 9216
auto swp2
iface swp2
  mtu 9216
auto swp3
iface swp3
  mtu 9216
auto swp4
iface swp4
  mtu 9216
auto swp29
iface swp29
  mtu 9216
auto swp30
iface swp30
  mtu 9216
auto swp31
iface swp31
auto swp32
iface swp32

```



```
        mtu 9216
auto peerlink
iface peerlink
    bond-slaves swp31 swp32
auto peerlink.4090
iface peerlink.4090
    address 169.254.255.1/30
    clagd-backup-ip 192.168.0.22
    clagd-enable yes
    clagd-peer-ip 169.254.255.2
    clagd-sys-mac 44:39:39:FF:40:90
auto vlan10
iface vlan10
    address 10.1.10.2/24
    address-virtual 44:39:39:FF:00:
10 10.1.10.1/24
    alias VM 10 Network
    vlan-id 10
    vlan-raw-device bridge
auto vlan20
iface vlan20
    address 10.1.20.2/24
    address-virtual 44:39:39:FF:00:
20 10.1.20.1/24
    alias VM 20 Network
    vlan-id 20
    vlan-raw-device bridge
```

```
auto bridge
iface bridge
    bridge-ports exit01-02
leaf01-02 leaf03-04 peerlink
    bridge-pvid 1
    bridge-vids 10 20 30 40
50 100-200
    bridge-vlan-aware yes
    mstpctl-treepri 4096
auto exit01-02
iface exit01-02
    bond-slaves swp29 swp30
    clag-id 2930
    mtu 9216
auto leaf01-02
iface leaf01-02
    bond-slaves swp1 swp2
    clag-id 1012
    mtu 9216
auto leaf03-04
iface leaf03-04
    bond-slaves swp3 swp4
    clag-id 1034
    mtu 9216
auto peerlink
iface peerlink
    bond-slaves swp31 swp32
auto peerlink.4090
iface peerlink.4090
    address 169.254.255.2/30
    alias Spine01-Spine02
MLAG peerlink
    clagd-backup-ip
192.168.0.21
    clagd-enable yes
    clagd-peer-ip
169.254.255.1
    clagd-sys-mac 44:39:39:
FF:40:90
auto vlan10
iface vlan10
    address 10.1.10.3/24
    address-virtual 44:39:
39:FF:00:10 10.1.10.1/24
    alias VM 10 Network
    vlan-id 10
    vlan-raw-device bridge
auto vlan20
iface vlan20
    address 10.1.20.3/24
```

```

address-virtual 44:39:
39:FF:00:20 10.1.20.1/24
alias VM 20 Network
vlan-id 20
vlan-raw-device bridge

```

Here is an example configuration for the switches leaf01 through leaf04. Note that the `clag-id` and `clagd-sys-mac` must be the same for the corresponding bonds on leaf01 and leaf02 as well as leaf03 and leaf04:

leaf01

```

cumulus@leaf01:~$ net show
configuration commands
echo 'auto lo' > /etc/network
/interfaces
echo 'iface lo inet loopback'
>> /etc/network/interfaces
echo 'auto eth0' >> /etc
/network/interfaces
echo 'iface eth0 inet dhcp' >>
/etc/network/interfaces
echo 'zebra=yes' > /etc/quagga
/daemons
echo '' > /etc/quagga/Quagga.
conf
rm /var/lib/cumulus/nclu
/nclu_acl.conf
net abort
net add hostname leaf01
net add interface swp1,49-52
mtu 9216
net add bond peerlink bond
slaves swp49-50
net add bond server01 bond
slaves swp1
net add bond uplink bond
slaves swp51-52
net add loopback lo ip address
10.0.0.11/32
net add interface eth0 ip
address dhcp
net add bridge alias bridge01
net add bridge bridge ports
peerlink,uplink,server01
net add bridge bridge vids
10,20,30,40,50,100-200

```

leaf02

```

cumulus@leaf02:~$ net show
conf commands
echo 'auto lo' > /etc/network
/interfaces
echo 'iface lo inet loopback'
>> /etc/network/interfaces
echo 'auto eth0' >> /etc
/network/interfaces
echo 'iface eth0 inet dhcp' >>
/etc/network/interfaces
echo 'zebra=yes' > /etc/quagga
/daemons
echo '' > /etc/quagga/Quagga.
conf
rm /var/lib/cumulus/nclu
/nclu_acl.conf
net abort
net add hostname leaf02
net add interface swp1,49-52
mtu 9216
net add bond peerlink bond
slaves swp49-50
net add bond server01 bond
slaves swp1
net add bond uplink bond
slaves swp51-52
net add loopback lo ip address
10.0.0.12/32
net add interface eth0 ip
address dhcp
net add bridge bridge ports
peerlink,uplink,server01
net add bridge bridge vids
10,20,30,40,50,100-200
net add bridge stp treeprio
32768

```



```
net add bridge stp treeprio  
32768  
net add vlan 500 ip address  
192.168.1.252/24  
net add vlan 500 address-  
virtual 00:00:5e:00:01:01  
192.168.1.254/24  
net add bond peerlink mtu 9216  
net add bond server01 mtu 9216  
net add bond uplink mtu 9216  
net add interface peerlink.  
4094 ip address 169.254.255.1  
/30  
net add interface peerlink.  
4094 clag backup-ip  
192.168.0.12  
net add interface peerlink.  
4094 clag peer-ip 169.254.255.2  
net add interface peerlink.  
4094 clag sys-mac 44:39:39:FF:  
40:94  
net add bond server01 bridge  
vids 10,20,30,40,50  
net add bond server01 clag id 1  
net add bond uplink clag id  
1000  
net add routing log file /var  
/log/quagga/quagga.log  
net add routing log timestamp  
precision 6
```

These commands create the following configuration ...

```
cumulus@leaf01:~$ cat /etc  
/network/interfaces  
auto lo  
iface lo inet loopback  
    address 10.0.0.11/32  
auto eth0  
iface eth0 inet dhcp  
auto swp1  
iface swp1  
    mtu 9216  
auto swp49  
iface swp49  
    mtu 9216  
auto swp50  
iface swp50  
    mtu 9216
```

```
net add bond peerlink mtu 9216  
net add bond server01 mtu 9216  
net add bond uplink mtu 9216  
net add interface peerlink.  
4094 ip address 169.254.255.2  
/30  
net add interface peerlink.  
4094 clag backup-ip  
192.168.0.11  
net add interface peerlink.  
4094 clag peer-ip 169.254.255.1  
net add interface peerlink.  
4094 clag sys-mac 44:39:39:FF:  
40:94  
net add bond server01 bridge  
vids 10,20,30,40,50  
net add bond server01 clag id 1  
net add bond uplink clag id  
1000  
net add log file /var/log  
/quagga/quagga.log  
net add log timestamp  
precision 6
```

These commands create the following configuration ...

```
cumulus@leaf02:~$ cat /etc  
/network/interfaces  
auto lo  
iface lo inet loopback  
    address 10.0.0.12/32  
auto eth0  
iface eth0 inet dhcp  
auto swp1  
iface swp1  
    mtu 9216  
auto swp49  
iface swp49  
    mtu 9216  
auto swp50  
iface swp50  
    mtu 9216  
auto swp51  
iface swp51  
    mtu 9216  
auto swp52  
iface swp52  
    mtu 9216  
auto bridge
```

```

auto swp51
iface swp51
    mtu 9216
auto swp52
iface swp52
    mtu 9216
auto bridge
iface bridge
    alias bridge01
    bridge-ports peerlink
server01 uplink
    bridge-vids 10 20 30 40 50
100-200
    bridge-vlan-aware yes
    mstpcctl-treepriority 32768
auto peerlink
iface peerlink
    bond-slaves swp49 swp50
    mtu 9216
auto peerlink.4094
iface peerlink.4094
    address 169.254.255.1/30
    clagd-backup-ip
192.168.0.12
    clagd-peer-ip 169.254.255.2
    clagd-sys-mac 44:39:39:FF:40:94
auto server01
iface server01
    bond-slaves swp1
    clag-id 1
    bridge-vids 10 20 30 40 50
    mtu 9216
auto uplink
iface uplink
    bond-slaves swp51 swp52
    clag-id 1000
    mtu 9216
auto vlan500
iface vlan500
    address 192.168.1.252/24
    address-virtual 00:00:5e:00:01:01 192.168.1.254/24
    vlan-id 500
    vlan-raw-device bridge

```

leaf03

```

iface bridge
    bridge-vlan-aware yes
    bridge-ports peerlink
server01 uplink
    bridge-vids 10 20 30 40 50
100-200
    mstpcctl-treepriority 32768
auto peerlink
iface peerlink
    bond-slaves swp49 swp50
    mtu 9216
auto peerlink.4094
iface peerlink.4094
    clagd-backup-ip
192.168.0.11
    clagd-peer-ip 169.254.255.1
    clagd-sys-mac 44:39:39:FF:40:94
    address 169.254.255.2/30
auto server01
iface server01
    bond-slaves swp1
    clag-id 1
    bridge-vids 10 20 30 40 50
    mtu 9216
auto uplink
iface uplink
    bond-slaves swp51 swp52
    clag-id 1000
    mtu 9216

```

leaf04



```
cumulus@leaf03:~$ net show
conf commands
echo 'auto lo' > /etc/network
/interfaces
echo 'iface lo inet loopback'
>> /etc/network/interfaces
echo 'auto eth0' >> /etc
/network/interfaces
echo 'iface eth0 inet dhcp' >>
/etc/network/interfaces
echo 'zebra=yes' > /etc/quagga
/daemons
echo '' > /etc/quagga/Quagga.
conf
rm /var/lib/cumulus/nclu
/nclu_acl.conf
net abort
net add hostname leaf03
net add interface swp1-2,51-52
mtu 9216
net add bond peerlink bond
slaves swp49-50
net add bond server03 bond
slaves swp1
net add bond server04 bond
slaves swp2
net add bond uplink bond
slaves swp51-52
net add loopback lo ip address
10.0.0.13/32
net add interface eth0 ip
address dhcp
net add bridge bridge ports
server03,server04,uplink,
peerlink
net add bridge bridge pvid 1
net add bridge bridge vids
10,20,30,40,50,100-200
net add bridge stp treeprio
32768
net add interface peerlink.
4093 ip address 169.254.255.1
/30
net add interface peerlink.
4093 alias 'Leaf03-Leaf04 MLAG
peerlink'
net add interface peerlink.
4093 clag backup-ip
192.168.0.14
net add interface peerlink.
4093 clag enable yes
```

```
cumulus@leaf04:~$ net show
configuration commands
echo 'auto lo' > /etc/network
/interfaces
echo 'iface lo inet loopback'
>> /etc/network/interfaces
echo 'auto eth0' >> /etc
/network/interfaces
echo 'iface eth0 inet dhcp' >>
/etc/network/interfaces
echo 'zebra=yes' > /etc/quagga
/daemons
echo '' > /etc/quagga/Quagga.
conf
rm /var/lib/cumulus/nclu
/nclu_acl.conf
net abort
net add hostname leaf04
net add interface swp1-2,51-52
mtu 9216
net add bond peerlink bond
slaves swp49-50
net add bond server03 bond
slaves swp1
net add bond server04 bond
slaves swp2
net add bond uplink bond
slaves swp51-52
net add loopback lo ip address
10.0.0.14/32
net add interface eth0 ip
address dhcp
net add bridge bridge ports
server03,server04,uplink,
peerlink
net add bridge bridge pvid 1
net add bridge bridge vids
10,20,30,40,50,100-200
net add bridge stp treeprio
32768
net add interface peerlink.
4093 ip address 169.254.255.2
/30
net add interface peerlink.
4093 alias 'Leaf03-Leaf04 MLAG
peerlink'
net add interface peerlink.
4093 clag backup-ip
192.168.0.13
net add interface peerlink.
4093 clag enable yes
```

```

net add interface peerlink.
4093 clag peer-ip 169.254.255.2
net add interface peerlink.
4093 clag sys-mac 44:39:39:FF:
40:93
net add bond server03 bridge
pvid 1
net add bond server04 bridge
pvid 1
net add bond server03 bridge
vids 10,20,30,40,50
net add bond server04 bridge
vids 10,20,30,40,50
net add bond server03 clag id 3
net add bond server03 mtu 9216
net add bond server04 mtu 9216
net add bond uplink mtu 9216
net add bond server04 clag id 4
net add bond uplink clag id
1000
net add log file /var/log
/quagga/quagga.log
net add log timestamp
precision 6
  
```

These commands create the following configuration ...

```

cumulus@leaf03:~$ cat /etc
/network/interfaces
# This file describes the
network interfaces available
on your system
# and how to activate them.
For more information, see
interfaces(5), ifup(8)
#
# Please see /usr/share/doc
/python-ifupdown2/examples/
for examples
#
#
# The loopback network
interface
auto lo
iface lo inet loopback
    address 10.0.0.13/32
# The primary network interface
auto eth0
iface eth0 inet dhcp
  
```

```

net add interface peerlink.
4093 clag peer-ip 169.254.255.1
net add interface peerlink.
4093 clag sys-mac 44:39:39:FF:
40:93
net add bond server03 bridge
pvid 1
net add bond server04 bridge
pvid 1
net add bond server03 bridge
vids 10,20,30,40,50
net add bond server04 bridge
vids 10,20,30,40,50
net add bond server03 clag id 3
net add bond server03 mtu 9216
net add bond server04 mtu 9216
net add bond uplink mtu 9216
net add bond server04 clag id 4
net add bond uplink clag id
1000
net add log file /var/log
/quagga/quagga.log
net add log timestamp
precision 6
  
```

These commands create the following configuration ...

```

cumulus@leaf04:~$ cat /etc
/network/interfaces
# This file describes the
network interfaces available
on your system
# and how to activate them.
For more information, see
interfaces(5), ifup(8)
#
# Please see /usr/share/doc
/python-ifupdown2/examples/
for examples
#
#
# The loopback network
interface
auto lo
iface lo inet loopback
    address 10.0.0.14/32
# The primary network interface
auto eth0
iface eth0 inet dhcp
  
```



```
auto swp1
iface swp1
    mtu 9216
auto swp2
iface swp2
    mtu 9216
auto swp49
iface swp49
auto swp50
iface swp50
auto swp51
iface swp51
    mtu 9216
auto swp52
iface swp52
    mtu 9216
auto bridge
iface bridge
    bridge-ports peerlink
server03 server04 uplink
    bridge-pvid 1
    bridge-vids 10 20 30 40 50
100-200
    bridge-vlan-aware yes
    mstptctl-treepprio 32768
auto peerlink
iface peerlink
    bond-slaves swp49 swp50
auto peerlink.4093
iface peerlink.4093
    address 169.254.255.1/30
    alias Leaf03-Leaf04 MLAG
peerlink
    clagd-backup-ip
192.168.0.14
    clagd-enable yes
    clagd-peer-ip 169.254.255.2
    clagd-sys-mac 44:39:39:FF:
40:93
auto server03
iface server03
    bond-slaves swp1
    bridge-pvid 1
    bridge-vids 10 20 30 40 50
    clag-id 3
    mtu 9216
auto server04
iface server04
    bond-slaves swp2
    bridge-pvid 1
    bridge-vids 10 20 30 40 50
```

```
auto swp1
iface swp1
    mtu 9216
auto swp2
iface swp2
    mtu 9216
auto swp49
iface swp49
auto swp50
iface swp50
auto swp51
iface swp51
    mtu 9216
auto swp52
iface swp52
    mtu 9216
auto bridge
iface bridge
    bridge-ports peerlink
server03 server04 uplink
    bridge-pvid 1
    bridge-vids 10 20 30 40 50
100-200
    bridge-vlan-aware yes
    mstptctl-treepprio 32768
auto peerlink
iface peerlink
    bond-slaves swp49 swp50
auto peerlink.4093
iface peerlink.4093
    address 169.254.255.2/30
    alias Leaf03-Leaf04 MLAG
peerlink
    clagd-backup-ip
192.168.0.13
    clagd-enable yes
    clagd-peer-ip 169.254.255.1
    clagd-sys-mac 44:39:39:FF:
40:93
auto server03
iface server03
    bond-slaves swp1
    bridge-pvid 1
    bridge-vids 10 20 30 40 50
    clag-id 3
    mtu 9216
auto server04
iface server04
    bond-slaves swp2
    bridge-pvid 1
    bridge-vids 10 20 30 40 50
```



```
clag-id 4
mtu 9216
auto uplink
iface uplink
bond-slaves swp51 swp52
clag-id 1000
mtu 9216
```

```
clag-id 4
mtu 9216
auto uplink
iface uplink
bond-slaves swp51 swp52
clag-id 1000
mtu 9216
```

Disabling clagd on an Interface

In the configurations above, the `clagd-peer-ip` and `clagd-sys-mac` parameters are mandatory, while the rest are optional. When mandatory `clagd` commands are present under a peer link subinterface, by default `clagd-enable` is set to `yes` and doesn't need to be specified; to disable `clagd` on the subinterface, set `clagd-enable` to `no`:

```
cumulus@spine01:~$ net add interface peerlink.4090 clag enable no
cumulus@spine01:~$ net pending
cumulus@spine01:~$ net commit
```

Use `clagd-priority` to set the role of the MLAG peer switch to primary or secondary. Each peer switch in an MLAG pair must have the same `clagd-sys-mac` setting. Each `clagd-sys-mac` setting should be unique to each MLAG pair in the network. For more details, refer to `man clagd`.

Checking the MLAG Configuration Status

You can easily check the status of your MLAG configuration using `net show clag`:

```
cumulus@spine01:~$ net show clag
The peer is alive
    Our Priority, ID, and Role: 32768 44:38:39:00:00:41 primary
    Peer Priority, ID, and Role: 32768 44:38:39:00:00:42 secondary
        Peer Interface and IP: peerlink.4090 169.254.255.2
                        Backup IP: 192.168.0.22 (active)
                        System MAC: 44:39:39:ff:40:90
```

CLAG Interfaces	Our Interface	Peer Interface	CLAG Id
Conflicts		Proto-Down Reason	
-	leaf03-04	leaf03-04	1034
-	exit01-02	-	2930
-	leaf01-02	leaf01-02	1012
-			-

Using the clagd Command Line Interface

A command line utility called `clagctl` is available for interacting with a running `clagd` service to get status or alter operational behavior. For detailed explanation of the utility, please refer to the `clagctl(8)` man page.

See the `clagctl` Output ...

The following is a sample output of the MLAG operational status displayed by `clagctl`:

```
The peer is alive
    Our Priority, ID, and Role: 32768 44:38:39:00:00:11 primary
    Peer Priority, ID, and Role: 32768 44:38:39:00:00:12 secondary
        Peer Interface and IP: peerlink.4094 169.254.255.2
            Backup IP: 192.168.0.12 (active)
            System MAC: 44:39:39:ff:40:94

CLAG Interfaces
Our Interface      Peer Interface      CLAG Id
Conflicts          Proto-Down Reason
-----           -----
server01          server01          1
-
    uplink         uplink          1000
-
```

Configuring MLAG with a Traditional Mode Bridge

It's possible to configure MLAG with a bridge in [traditional mode](#) (see page 288) instead of [VLAN-aware mode](#) (see page 277).

⚠ Traditional Mode Limitation

You cannot configure a traditional mode bridge using [NCLU](#) (see page 80); you must configure it manually in the `/etc/network/interfaces` file.

In order to configure MLAG with a traditional mode bridge, the peer link and all dual-connected links must be configured as [untagged/native](#) (see page 288) ports on a bridge (note the absence of any VLANs in the `bridge-ports` line and the lack of the `bridge-vlan-aware` parameter below):

✔ MLAG with a Traditional Mode Bridge

```
auto br0
iface br0
    bridge-ports peerlink spine1-2 host1 host2
```



For a deeper comparison of traditional versus VLAN-aware bridge modes, read this [knowledge base article](#).

Peer Link Interfaces and the protodown State

In addition to the standard UP and DOWN administrative states, an interface that is a member of an MLAG bond can also be in a `protodown` state. When MLAG detects a problem that could result in connectivity issues such as traffic black-holing or a network meltdown if the link carrier was left in an UP state, it can put that interface into `protodown` state. Such connectivity issues include:

- When the peer link goes down but the peer switch is up (that is, the backup link is active).
- When the bond is configured with an MLAG ID, but the `c1agd` service is not running (whether it was deliberately stopped or simply died).
- When an MLAG-enabled node is booted or rebooted, the MLAG bonds are placed in a `protodown` state until the node establishes a connection to its peer switch, or five minutes have elapsed.

When an interface goes into a `protodown` state, it results in a local OPER DOWN (carrier down) on the interface. As of Cumulus Linux 2.5.5, the `protodown` state can be manipulated with the `ip link set` command. Given its use in preventing network meltdowns, manually manipulating `protodown` is not recommended outside the scope of interaction with the Cumulus Networks support team.

The following `ip link show` command output shows an interface in `protodown` state. Notice that the link carrier is down (NO-CARRIER):

```
cumulus@switch:~$ net show bridge link swp1
3: swp1 state DOWN: <NO-CARRIER,BROADCAST,MULTICAST,MASTER,UP> mtu 921
6 master pfifo_fast master host-bond1 state DOWN mode DEFAULT qlen 500
  protodown on
    link/ether 44:38:39:00:69:84 brd ff:ff:ff:ff:ff:ff
```

Specifying a Backup Link

You can specify a backup link for your peer links in the event that the peer link goes down. When this happens, the `c1agd` service uses the backup link to check the health of the peer switch. To configure this, add `c1agd-backup-ip <ADDRESS>` to the peer link configuration:

Specifying a Backup Link

```
cumulus@spine01:~$ net add interface peerlink.4094 clag backup-
  ip 192.0.2.50
cumulus@spine01:~$ net pending
cumulus@spine01:~$ net commit
```



The backup IP address must be different than the peer link IP address (`c1agd-peer-ip`). It must be reachable by a route that doesn't use the peer link and it must be in the same network namespace as the peer link IP address.



Cumulus Networks recommends you use the switch's management IP address for this purpose.

You can also specify the backup UDP port. The port defaults to 5342, but you can configure it as an argument in `clagd-args` using `--backupPort <PORT>`.

```
cumulus@spine01:~$ net add interface peerlink.4094 clag args --  
backupPort 5400  
cumulus@spine01:~$ net pending  
cumulus@spine01:~$ net commit
```

You can see the backup IP address if you run `net show clag`:

```
cumulus@spine01:~$ net show clag  
The peer is alive  
    Our Priority, ID, and Role: 32768 44:38:39:00:00:41  
primary  
    Peer Priority, ID, and Role: 32768 44:38:39:00:00:42  
secondary  
        Peer Interface and IP: peerlink.4090 169.254.255.2  
        Backup IP: 192.168.0.22 (active)  
        System MAC: 44:39:39:ff:40:90  
  
CLAG Interfaces  
Our Interface      Peer Interface      CLAG Id  
Conflicts          Proto-Down Reason  
-----  
-----  
      leaf03-04      leaf03-04      1034  
-  
      exit01-02      -            2930  
-  
      leaf01-02      leaf01-02      1012  
-
```

Specifying a Backup Link to a VRF

You can configure the backup link to a [VRF \(see page 574\)](#) or [management VRF \(see page 593\)](#). Include the name of the VRF or management VRF when you specify `clagd-backup-ip <ADDRESS> vrf <VRF name>`. Here is a sample configuration:

Specifying a Backup Link to a VRF

```
cumulus@spine01:~$ net add interface peerlink.4090 clag backup-ip 192.168.0.22 vrf mgmt
cumulus@spine01:~$ net pending
cumulus@spine01:~$ net commit
```

 You cannot use the VRF on a peer link subinterface.

Verify the backup link by running `net show clag backup-ip`:

```
cumulus@leaf01:~$ net show clag backup-ip
Backup info:
IP: 192.168.0.12; State: active; Role: primary
Peer priority and id: 32768 44:38:39:00:00:12; Peer role: secondary
```

⌚ Comparing VRF and Management VRF Configurations

The configuration for both a VRF and management VRF is exactly the same. A sample configuration where the backup interface is in a VRF could be as follows:

```
cumulus@leaf01:~$ net show configuration
...
auto swp52s0
iface swp52s0
    address 192.0.2.1/24
    vrf green

auto green
iface green
    vrf-table auto

auto peer5.4000
iface peer5.4000
    address 192.0.2.15/24
    clagd-peer-ip 192.0.2.16
    clagd-backup-ip 192.0.2.2 vrf green
    clagd-sys-mac 44:38:39:01:01:01
...
```

Which you can verify with `net show clag status verbose`:



```
cumulus@leaf01:~$ net show clag status verbose
The peer is alive
    Peer Priority, ID, and Role: 32768 00:02:00:00:00:13
primary
    Our Priority, ID, and Role: 32768 c4:54:44:f6:44:5a
secondary
    Peer Interface and IP: peer5.4000 192.0.2.16
                            Backup IP: 192.0.2.2 vrf green (active)
                            System MAC: 44:38:39:01:01:01

CLAG Interfaces
Our Interface      Peer Interface      CLAG Id
Conflicts          Proto-Down Reason
-----            -----
-----            -----
        bond4      bond4              4
-
        bond1      bond1              1
-
        bond2      bond2              2
-
        bond3      bond3              3
-
...

```

Monitoring Dual-Connected Peers

Upon receipt of a valid message from its peer, the switch knows that `clagd` is alive and executing on that peer. This causes `clagd` to change the system ID of each bond that was assigned a `clag-id` from the default value (the MAC address of the bond) to the system ID assigned to both peer switches. This makes the hosts connected to each switch act as if they are connected to the same system so that they will use all ports within their bond. Additionally, `clagd` determines which bonds are dual-connected and modifies the forwarding and learning behavior to accommodate these dual-connected bonds.

If the peer does not receive any messages for three update intervals, then that peer switch is assumed to no longer be acting as an MLAG peer. In this case, the switch reverts all configuration changes so that it operates as a standard non-MLAG switch. This includes removing all statically assigned MAC addresses, clearing the egress forwarding mask, and allowing addresses to move from any port to the peer port. Once a message is again received from the peer, MLAG operation starts again as described earlier. You can configure a custom timeout setting by adding `--peerTimeout <VALUE>` to `clagd-args`, like this:

```
cumulus@spine01:~$ net add interface peerlink.4094 clag args --
peerTimeout 900
cumulus@spine01:~$ net pending
cumulus@spine01:~$ net commit
```

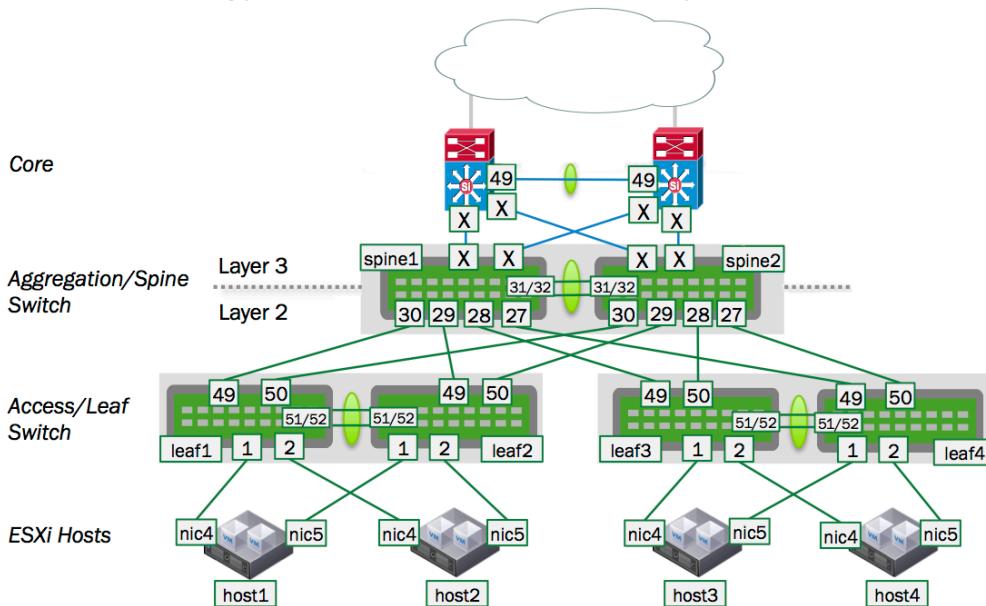
Once bonds are identified as dual-connected, `c1agd` sends more information to the peer switch for those bonds. The MAC addresses (and VLANs) that have been dynamically learned on those ports are sent along with the LACP partner MAC address for each bond. When a switch receives MAC address information from its peer, it adds MAC address entries on the corresponding ports. As the switch learns and ages out MAC addresses, it informs the peer switch of these changes to its MAC address table so that the peer can keep its table synchronized. Periodically, at 45% of the bridge ageing time, a switch will send its entire MAC address table to the peer, so that peer switch can verify that its MAC address table is properly synchronized.

The switch sends an update frequency value in the messages to its peer, which tells `c1agd` how often the peer will send these messages. You can configure a different frequency by adding `--lacpPoll <SECONDS>` to `c1agd-args`:

```
cumulus@spine01:~$ net add interface peerlink.4094 clag args --
lacpPoll 900
cumulus@spine01:~$ net pending
cumulus@spine01:~$ net commit
```

Configuring Layer 3 Routed Uplinks

In this scenario, the spine switches connect at layer 3, as shown in the image below. Alternatively, the spine switches can be singly connected to each core switch at layer 3 (not shown below).



In this design, the spine switches route traffic between the server hosts in the layer 2 domains and the core. The servers (host1 - host4) each have a layer 2 connection up to the spine layer where the default gateway for the host subnets resides. However, since the spine switches act as gateway devices communicate at layer 3, you need to configure a protocol such as [VRR \(see page 337\)](#) (Virtual Router Redundancy) between the spine switch pair to support active/active forwarding.



Then, to connect the spine switches to the core switches, you need to determine whether the routing is static or dynamic. If it's dynamic, you must choose which protocol — [OSPF \(see page 500\)](#) or [BGP \(see page 516\)](#) — to use. When enabling a routing protocol in an MLAG environment it is also necessary to manage the uplinks, because by default MLAG is not aware of layer 3 uplink interfaces. In the event of a peer link failure MLAG does not remove static routes or bring down a BGP or OSPF adjacency unless a separate link state daemon such as `ifplugd` is used.

IGMP Snooping with MLAG

IGMP snooping processes IGMP reports received on a bridge port in a bridge to identify hosts that are configured to receive multicast traffic destined to that group. An IGMP query message received on a port is used to identify the port that is connected to a router and configured to receive multicast traffic.

IGMP snooping is enabled by default on the bridge. IGMP snooping multicast database entries and router port entries are synced to the peer MLAG switch. If there is no multicast router in the VLAN, the IGMP querier can be configured on the switch to generate IGMP query messages by adding a configuration like the following:

✓ Configuring IGMP Snooping

```
cumulus@spine01:~$ net add bridge bridge mcsnoop yes
cumulus@spine01:~$ net add bridge bridge igmp-querier-src 123.1
    .1.1
cumulus@spine01:~$ net pending
cumulus@spine01:~$ net commit
```

To display multicast group and router port information, use the `net show bridge mdb` command:

```
cumulus@switch:~$ net show bridge mdb
dev br port bond0 vlan 100 grp 234.1.1.1 temp
router ports on br: bond0
```

Monitoring the Status of the `clagd` Service

Due to the critical nature of the `clagd` service, `systemd` continuously monitors the status of `clagd`. `systemd` monitors the `clagd` service through the use of notify messages every 30 seconds. If the `clagd` service dies or becomes unresponsive for any reason and `systemd` receives no messages after 60 seconds, `systemd` restarts `clagd`. `systemd` logs these failures in `/var/log/syslog`, and, on the first failure, generates a `cl-support` file as well.

This monitoring is automatically configured and enabled as long as `clagd` is enabled (that is, `clagd-peer-ip` and `clagd-sys-mac` are configured for an interface) and the `clagd` service is running. When `clagd` is explicitly stopped, for example with the `systemctl stop clagd.service` command, monitoring of `clagd` is also stopped.



Checking clagd Status

You can check the status of clagd monitoring by using the `cl-service-summary` command:

```
cumulus@switch:~$ sudo cl-service-summary summary
The systemctl daemon 5.4 uptime: 15m

...
Service clagd           enabled      active
```

Or the `systemctl status` command:

```
cumulus@switch:~$ sudo systemctl status clagd.service
  clagd.service - Cumulus Linux Multi-Chassis LACP Bonding
Daemon
   Loaded: loaded (/lib/systemd/system/clagd.service; enabled)
   Active: active (running) since Mon 2016-10-03 20:31:50 UTC;
4 days ago
     Docs: man:clagd(8)
 Main PID: 1235 (clagd)
    CGroup: /system.slice/clagd.service
            1235 /usr/bin/python /usr/sbin/clagd --daemon 169.25
4.255.2 peerlink.4090 44:39:39:FF:40:90 --prior...
            1307 /sbin/bridge monitor fdb

Feb 01 23:19:30 leaf01 clagd[1717]: Cleanup is executing.
Feb 01 23:19:31 leaf01 clagd[1717]: Cleanup is finished
Feb 01 23:19:31 leaf01 clagd[1717]: Beginning execution of
clagd version 1.3.0
Feb 01 23:19:31 leaf01 clagd[1717]: Invoked with: /usr/sbin
/clagd --daemon 169.254.255.2 peerlink.4094 44:39:39:FF:40:94
--pri...168.0.12
Feb 01 23:19:31 leaf01 clagd[1717]: Role is now secondary
Feb 01 23:19:31 leaf01 clagd[1717]: Initial config loaded
Feb 01 23:19:31 leaf01 systemd[1]: Started Cumulus Linux Multi-
Chassis LACP Bonding Daemon.
Feb 01 23:24:31 leaf01 clagd[1717]: HealthCheck: reload
timeout.
Feb 01 23:24:31 leaf01 clagd[1717]: Role is now primary;
Reload timeout
Hint: Some lines were ellipsized, use -l to show in full.
```

MLAG Best Practices

For MLAG to function properly, the dual-connected hosts' interfaces should be configured identically on the pair of peering switches. See the note above in the [Configuring MLAG \(see page 304\)](#) section.

Understanding MTU in an MLAG Configuration

Note that the [MTU \(see page 201\)](#) in MLAG traffic is determined by the bridge MTU. Bridge MTU is determined by the lowest MTU setting of an interface that is a member of the bridge. If an MTU other than the default of 1500 bytes is desired, you must configure the MTU on each physical interface and bond interface that are members of the MLAG bridges in the entire bridged domain.

For example, if an MTU of 9216 is desired through the MLAG domain in the example shown above:

On all four leaf switches, [configure mtu 9216 \(see page 201\)](#) for each of the following interfaces, since they are members of bridge *bridge*: peerlink, uplink, server01.

Configuring MTU

```
cumulus@leaf01:~$ net add bond peerlink mtu 9216
cumulus@leaf01:~$ net add bond uplink mtu 9216
cumulus@leaf01:~$ net add bond server01 mtu 9216
cumulus@leaf01:~$ net pending
cumulus@leaf01:~$ net commit
```

The above commands produce the following configuration in the `/etc/network/interfaces` file:

```
auto bridge
iface bridge
    bridge-ports peerlink uplink server01
    bridge-vids 10 20 30 40 50 100-200
    bridge-vlan-aware yes
    mstpcctl-treepri 32768
```

Likewise, to ensure the MTU 9216 path is respected through the spine switches above, also change the MTU setting for bridge *bridge* by configuring `mtu 9216` for each of the following members of bridge *bridge* on both spine01 and spine02: leaf01-02, leaf03-04, exit01-02, peerlink.

```
cumulus@spine01:~$ net add bond leaf01-02 mtu 9216
cumulus@spine01:~$ net add bond leaf03-04 mtu 9216
cumulus@spine01:~$ net add bond leaf01-02 mtu 9216
cumulus@spine01:~$ net add bond peerlink mtu 9216
cumulus@spine01:~$ net pending
cumulus@spine01:~$ net commit
```

The above commands produce the following configuration in the `/etc/network/interfaces` file:

```

auto bridge
iface bridge
    bridge-ports leaf01-02 leaf03-04 exit01-02 peerlink
    bridge-pvid 1
    bridge-vids 10 20 30 40 50 100-200
    bridge-vlan-aware yes
    mstptctl-treeprio 4096

```

Sizing the Peerlink

What's the best size for a peerlink? Before we answer that, let's talk a little bit about the peerlink itself.

The peerlink tends to carry very little traffic when compared to the bandwidth consumed by dataplane traffic. In a typical MLAG configuration, most every connection between the two switches in the MLAG pair is dual-connected, so the only traffic going across the peerlink is traffic from the `c1agd` process and some LLDP or LACP traffic; the traffic received on the peerlink is not forwarded out of the dual-connected bonds.

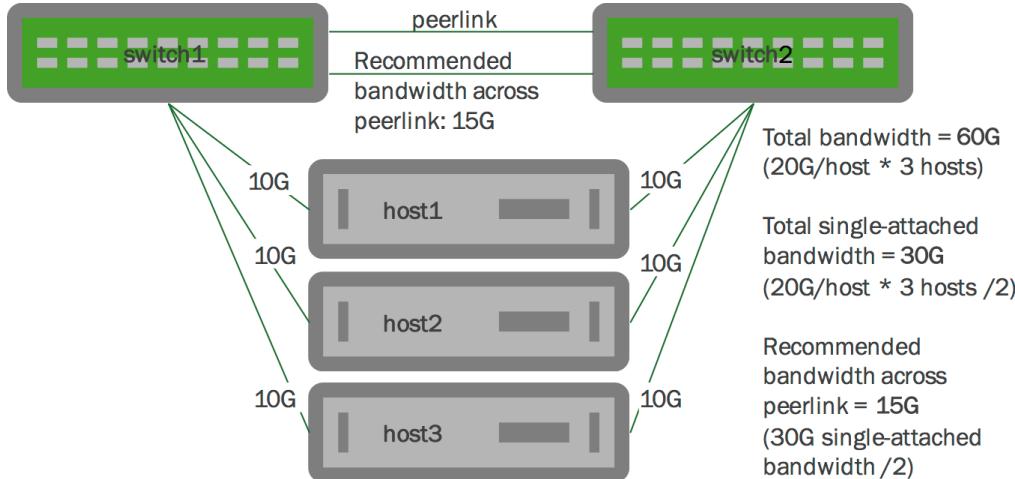
However, there are some instances where a host is connected to only one switch in the MLAG pair; these include:

- You have a hardware limitation on the host where there is only one PCIE slot, and thus, one NIC on the system, so the host is only single-connected across that interface.
- The host doesn't support 802.3ad and you can't create a bond on it.
- You are accounting for a link failure, where the host may become single connected until the failure is rectified.

So, in terms of sizing the peerlink, in general, you need to determine how much bandwidth is traveling across the single-connected interfaces, and allocate half of that bandwidth to the peerlink. We recommend half of the single-connected bandwidth because, on average, one half of the traffic destined to the single-connected host will arrive on the switch directly connected to the single-connected host and the other half will arrive on the switch that is not directly connected to the single-connected host. When this happens, only the traffic that arrives on the switch that is not directly connected to the single-connected host needs to traverse the peerlink, which is how you calculate 50% of the traffic.

In addition, you may want to add extra links to the peerlink bond to handle link failures in the peerlink bond itself.

In illustration below, each host has 2 10G links, with each 10G link going to each switch in the MLAG pair. Each host has 20G of dual-connected bandwidth, so all three hosts have a total of 60G of dual-connected bandwidth. We recommend you allocate at least 15G of bandwidth to each peerlink bond, which represents half of the single-connected bandwidth.



Scaling this example out to a full rack, when planning for link failures, you need only allocate enough bandwidth to meet your site's strategy for handling failure scenarios. Imagine a full rack with 40 servers and two switches in it. You may plan for, say, 4 to 6 servers to lose connectivity to a single switch and become single connected before you respond to the event. So expanding upon our previous example, if you have 40 hosts each with 20G of bandwidth dual-connected to the MLAG pair, you might allocate 20G to 30G of bandwidth to the peerlink — which accounts for half of the single-connected bandwidth for 4 to 6 hosts.

STP Interoperability with MLAG

Cumulus Networks recommends that you always enable STP in your layer 2 network.

Further, with MLAG, Cumulus Networks recommends you enable BPDU guard on the host-facing bond interfaces. (For more information about BPDU guard, see [BPDU Guard and Bridge Assurance \(see page 244\)](#).)

⌚ Debugging STP with MLAG

Running `net show <interface> spanning-tree` displays MLAG information that can be useful when debugging:

```
cumulus@switch:~$ net show bridge spanning-tree
bridge:peerlink CIST info
  enabled          yes
  role             Designated
  port id         8.002
  state            forwarding
  .....
  bpdufilter port no
  clag ISL        yes
  UP      yes
  clag role       primary
  mac    00:00:00:00:00:00
  clag remote portID F.FFFF
  mac    44:39:39:FF:40:90
                                clag ISL Oper
                                clag dual conn
                                clag system
```

Best Practices for STP with MLAG

- The STP global configuration must be the same on both the switches.
- The STP configuration for dual-connected ports should be the same on both peer switches.
- Use [NCLU \(see page 80\)](#) (`net`) commands for all spanning tree configurations, including bridge priority, path cost and so forth. Do not use `brctl` commands for spanning tree, except for `brctl stp on/off`, as changes are not reflected to `mstpd` and can create conflicts.

Troubleshooting MLAG

Viewing the MLAG Log File

By default, when `clagd` is running, it logs its status to the `/var/log/clagd.log` file and `syslog`. Example log file output is below:

```
cumulus@spine01:~$ sudo tail /var/log/clagd.log
2016-10-03T20:31:50.471400+00:00 spine01 clagd[1235]: Initial config
loaded
2016-10-03T20:31:52.479769+00:00 spine01 clagd[1235]: The peer switch
is active.
2016-10-03T20:31:52.496490+00:00 spine01 clagd[1235]: Initial data
sync to peer done.
2016-10-03T20:31:52.540186+00:00 spine01 clagd[1235]: Role is now
primary; elected
2016-10-03T20:31:54.250572+00:00 spine01 clagd[1235]: HealthCheck:
role via backup is primary
2016-10-03T20:31:54.252642+00:00 spine01 clagd[1235]: HealthCheck:
backup active
2016-10-03T20:31:54.537967+00:00 spine01 clagd[1235]: Initial data
sync from peer done.
2016-10-03T20:31:54.538435+00:00 spine01 clagd[1235]: Initial
handshake done.
2016-10-03T20:31:58.527464+00:00 spine01 clagd[1235]: leaf03-04 is
now dual connected.
2016-10-03T22:47:35.255317+00:00 spine01 clagd[1235]: leaf01-02 is
now dual connected.
```

Large Packet Drops on the Peerlink Interface

If you notice a large volume of packet drops across one of the peerlink interfaces, this could be expected behavior. These drops serve to prevent looping of BUM (broadcast, unknown unicast, multicast) packets. When a packet is received across the peerlink, if the destination lookup results in an egress interface that is a dual-connected bond, the switch does not forward the packet to prevent loops. This results in a drop being recorded on the peerlink.

You can detect this issue by running `net show counters` or `ethtool -S <interface>`.



Using [NCLU \(see page 80\)](#), the number of dropped packets is displayed in the RX_DRP column when you run `net show counters`:

```
cumulus@switch:~$ net show counters

Kernel Interface table
Iface           MTU     Met    RX_OK    RX_ERR    RX_DRP
RX_OVR      TX_OK    TX_ERR    TX_DRP    TX_OVR   Flg
-----  -----  -----  -----  -----  -----
peerlink      1500      0  19226721      0  2952460
0      55115330      0      364      0  BMmRU
peerlink.4094 1500      0      0      0  BMRU
0      5379243       0      0      0  BMRU
swp51        1500      0  6587220      0  2129676
0      38957769      0      202      0  BMsRU
swp52        1500      0  12639501      0  822784
0      16157561      0      162      0  BMsRU
```

When you run `ethtool -S` on a peerlink interface, the drops are indicated by the HwIfInDiscards counter:

```
cumulus@switch:~$ sudo ethtool -S swp51
NIC statistics:
HwIfInOctets: 669507330
HwIfInUcastPkts: 658871
HwIfInBcastPkts: 2231559
HwIfInMcastPkts: 3696790
HwIfOutOctets: 2752224343
HwIfOutUcastPkts: 1001632
HwIfOutMcastPkts: 3743199
HwIfOutBcastPkts: 34212938
HwIfInDiscards: 2129675
```

Caveats and Errata

If both the backup and peer connectivity are lost within a 30-second window, the switch in the secondary role misinterprets the event sequence, believing the peer switch is down, so it takes over as the primary.

LACP Bypass

On Cumulus Linux, *LACP Bypass* is a feature that allows a [bond \(see page 268\)](#) configured in 802.3ad mode to become active and forward traffic even when there is no LACP partner. A typical use case for this feature is to enable a host, without the capability to run LACP, to PXE boot while connected to a switch on a bond configured in 802.3ad mode. Once the pre-boot process finishes and the host is capable of running LACP, the normal 802.3ad link aggregation operation takes over.

Contents

This chapter covers ...

- Understanding the LACP Bypass All-active Mode (see page 334)
 - LACP Bypass and MLAG Deployments (see page 334)
- Configuring LACP Bypass (see page 334)
 - Traditional Bridge Mode Configuration (see page 336)

Understanding the LACP Bypass All-active Mode

When a bond has multiple slave interfaces, each bond slave interface operates as an active link while the bond is in bypass mode. This is known as *all-active mode*. This is useful during PXE boot of a server with multiple NICs, when the user cannot determine beforehand which port needs to be active.

Keep in the mind the following caveats with all-active mode:

- All-active mode is not supported on bonds that are not specified as bridge ports on the switch.
- Spanning tree protocol (STP) does not run on the individual bond slave interfaces when the LACP bond is in all-active mode. Therefore, only use all-active mode on host-facing LACP bonds. Cumulus Networks highly recommends you configure [STP BPDU guard \(see page 244\)](#) along with all-active mode.



The following features are not supported:

- priority mode
- bond-lacp-bypass-period
- bond-lacp-bypass-priority
- bond-lacp-bypass-all-active

LACP Bypass and MLAG Deployments

In an [MLAG deployment \(see page 300\)](#) where bond slaves of a host are connected to two switches and the bond is in all-active mode, all the slaves of bond are active on both the primary and secondary MLAG nodes.

Configuring LACP Bypass

To enable LACP bypass on the host-facing bond, set `bond-lacp-bypass-allow` to yes.

Example VLAN-aware Bridge Mode Configuration

The following commands create a VLAN-aware bridge with LACP bypass enabled:

```
cumulus@switch:~$ net add bond bond1 bond slaves swp51s2,swp51s3
cumulus@switch:~$ net add bond bond1 clag id 1
cumulus@switch:~$ net add bond bond1 bond lacp-bypass-allow
cumulus@switch:~$ net add bond bond1 stp bpduguard
```



```
cumulus@switch:~$ net add bridge bridge ports bond1,bond2,bond3,  
bond4,peer5  
cumulus@switch:~$ net add bridge bridge vids 100-105  
cumulus@switch:~$ net pending  
cumulus@switch:~$ net commit
```

These commands create the following stanzas in /etc/network/interfaces:

```
auto bond1  
iface bond1  
    bond-lacp-bypass-allow yes  
    bond-slaves swp51s2 swp51s3  
    clag-id 1  
    mstpctl-bpduguard yes  
  
...  
  
auto bridge  
iface bridge  
    bridge-ports bond1 bond2 bond3 bond4 peer5  
    bridge-vids 100-105  
    bridge-vlan-aware yes
```

You can check the status of the configuration by running `net show interface <bond>` on the bond and its slave interfaces:

```
cumulus@switch:~$ net show interface bond1  
  
      Name      MAC                Speed     MTU     Mode  
--- -----  
UP  bond1  44:38:39:00:00:5b  1G        1500  Bond/Trunk  
  
Bond Details  
-----  
Bond Mode:          LACP  
Load Balancing:    Layer3+4  
Minimum Links:     1  
In CLAG:           CLAG Active  
LACP Sys Priority:  
LACP Rate:         Fast Timeout  
LACP Bypass:       LACP Bypass Not Supported  
  
      Port      Speed     TX     RX     Err   Link Failures  
--- -----  
UP  swp51s2(P)  1G        0      0      0            0
```

```
UP swp51s3(P) 1G      0   0   0      0
All VLANs on L2 Port
-----
100-105

Untagged
-----
1

Vlans in disabled State
-----
100-105

LLDP
-----
swp51s2(P) === swp1(spine01)
swp51s3(P) === swp1(spine02)
```

Use the `cat` command to verify that LACP bypass is enabled on a bond and its slave interfaces:

```
cumulus@switch:~$ cat /sys/class/net/bond1/bonding/lacp_bypass
on 1
cumulus@switch:~$ cat /sys/class/net/bond1/bonding/slaves
swp51 swp52
cumulus@switch:~$ cat /sys/class/net/swp52/bonding_slave/ad_rx_bypass
1
cumulus@switch:~$ cat /sys/class/net/swp51/bonding_slave/ad_rx_bypass
1
```

Traditional Bridge Mode Configuration

The following configuration shows LACP bypass enabled for multiple active interfaces (all-active mode) with a bridge in [traditional bridge mode](#) (see page 288):

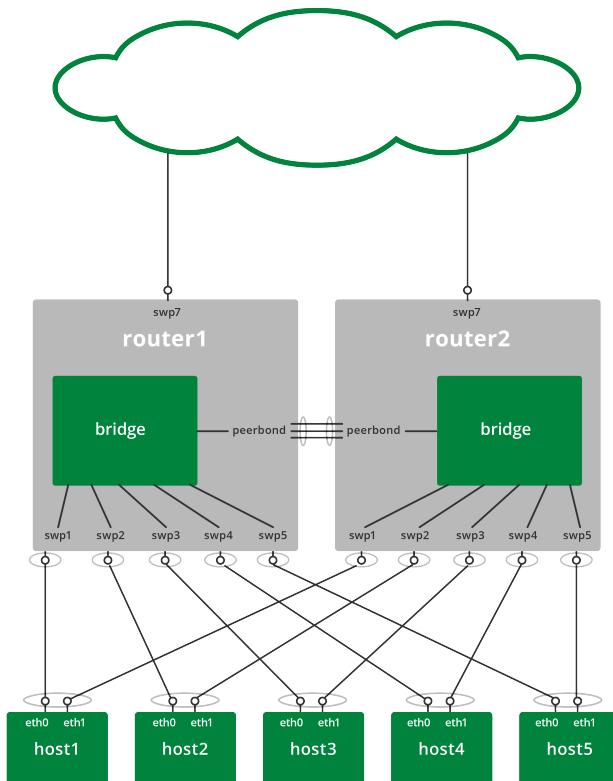
```
auto bond1
iface bond1
    bond-slaves swp3 swp4
    bond-lacp-bypass-allow 1

auto br0
iface br0
    bridge-ports bond1 bond2 bond3 bond4 peer5
    mstpctl-bpduguard bond1=yes
```

Virtual Router Redundancy - VRR

Virtual Router Redundancy (VRR) enables hosts to communicate with any redundant router without reconfiguration, running dynamic router protocols, or running router redundancy protocols. This means that redundant routers will respond to Address Resolution Protocol (ARP) requests from hosts. Routers are configured to respond in an identical manner, but if one fails, the other redundant routers will continue to respond, leaving the hosts with the impression that nothing has changed.

The diagram below illustrates a basic VRR-enabled network configuration. The network includes several hosts, and two routers running Cumulus Linux configured with [Multi-chassis Link Aggregation \(see page 300 \)](#) (MLAG):



A production implementation will have many more server hosts and network connections than are shown here. However, this basic configuration provides a complete description of the important aspects of the VRR setup.

As the bridges in each of the redundant routers are connected, they will each receive and reply to ARP requests for the virtual router IP address.



Multiple ARP Replies



Each ARP request made by a host will receive replies from each router; these replies will be identical, and so the host receiving the replies will either ignore replies after the first, or accept them and overwrite the previous identical reply, rather than being confused over which response is correct.

① Reserved MAC Address Range

A range of MAC addresses is reserved for use with VRR, in order to prevent MAC address conflicts with other interfaces in the same bridged network. The reserved range is `00:00:5E:00:01:00` to `00:00:5E:00:01:ff`.



Cumulus Networks recommends using MAC addresses from the reserved range when configuring VRR.



The reserved MAC address range for VRR is the same as for the Virtual Router Redundancy Protocol (VRRP), as they serve similar purposes.

Contents

This chapter covers ...

- Configuring a VRR-enabled Network (see page 338)
 - Configuring the Routers (see page 338)
 - Configuring the Hosts (see page 339)

Configuring a VRR-enabled Network

Configuring the Routers

The routers implement the layer 2 network interconnecting the hosts and the redundant routers. To configure the routers, add a bridge with the following interfaces to each router:

- One bond interface or switch port interface to each host.



For networks using MLAG, use bond interfaces. Otherwise, use switch port interfaces.

- One or more interfaces to each peer router.



Multiple inter-peer links are typically bonded interfaces, in order to accommodate higher bandwidth between the routers, and to offer link redundancy.



Example VLAN-aware Bridge Configuration

The example NCLU commands below create a VLAN-aware bridge interface for a VRR-enabled network:

NCLU Commands

```
cumulus@switch:~$ net add bridge
cumulus@switch:~$ net add vlan 500 ip address 192.168.0.252/24
cumulus@switch:~$ net add vlan 500 ip address-virtual 00:00:5e:00:
01:01 192.168.0.254/24
cumulus@switch:~$ net add vlan 500 ipv6 address 2001:aa::1/48
cumulus@switch:~$ net add vlan 500 ipv6 address-virtual 00:00:5e:0
0:01:01 2001:aa::1/48
cumulus@switch:~$ net pending
cumulus@switch:~$ net commit
```

The NCLU commands above produce the following /etc/network/interfaces snippet:

/etc/network/interfaces Snippet

```
auto bridge
iface bridge
    bridge-vids 500
    bridge-vlan-aware yes

auto vlan500
iface vlan500
    address 192.168.0.252/24
    address 2001:aa::1/48
    address-virtual 00:00:5e:00:01:01 2001:aa::1/48 192.168.0.254/
24
    vlan-id 500
    vlan-raw-device bridge
```

Configuring the Hosts

Each host should have two network interfaces. The routers configure the interfaces as bonds running LACP; the hosts should also configure its two interfaces using teaming, port aggregation, port group, or EtherChannel running LACP. Configure the hosts, either statically or via DHCP, with a gateway address that is the IP address of the virtual router; this default gateway address never changes.

Configure the links between the hosts and the routers in *active-active* mode for First Hop Redundancy Protocol.



ifplugged

`ifplugged` is an Ethernet link-state monitoring daemon, that can execute user-specified scripts to configure an Ethernet device when a cable is plugged in, or automatically unconfigure it when a cable is removed.

Follow the steps below to install and configure the `ifplugged` daemon.

Install ifplugged

1. Update the switch before installing the daemon:

```
cumulus@switch:~$ sudo apt-get update
```

2. Install the `ifplugged` package:

```
cumulus@switch:~$ sudo apt-get install ifplugged
```

Configure ifplugged

Once `ifplugged` is installed, two configuration files must be edited to set up `ifplugged`:

- `/etc/default/ifplugged`
- `/etc/ifplugged/action.d/ifupdown`

Example ifplugged Configuration

The example `ifplugged` configuration below show that `ifplugged` has been configured to bring down all uplinks when the peerbond goes down in an MLAG environment.



`ifplugged` is configured on both both the primary and secondary [MLAG](#) (see page 300) switches in this example.

1. Open `/etc/default/ifplugged` in a text editor.
2. Configure the file as appropriate, and add the peerbond name, before saving:

```
INTERFACES="peerbond"
HOTPLUG_INTERFACES=""
ARGS="-q -f -u0 -d1 -w -I"
SUSPEND_ACTION="stop"
```

3. Open `/etc/ifplugged/action.d/ifupdown` in a text editor.
4. Configure the script, and save the file.

```

#!/bin/sh
set -e
case "$2" in
up)
    clagrole=$(clagctl | grep "Our Priority" | awk '{print $8}')
    if [ "$clagrole" = "secondary" ]
    then
        #List all the interfaces below to bring up when
        #clag peerbond comes up.
        for interface in swp1 bond1 bond3 bond4
        do
            echo "bringing up : $interface"
            ip link set $interface up
        done
    fi
    ;;
down)
    clagrole=$(clagctl | grep "Our Priority" | awk '{print $8}')
    if [ "$clagrole" = "secondary" ]
    then
        #List all the interfaces below to bring down
        #when clag peerbond goes down.
        for interface in swp1 bond1 bond3 bond4
        do
            echo "bringing down : $interface"
            ip link set $interface down
        done
    fi
    ;;
esac

```

5. Restart `ifplugd` to implement the changes:

```
cumulus@switch:$ sudo systemctl restart ifplugd.service
```

Caveats and Errata

The default shell for `ifplugd` is `dash` (`/bin/sh`), rather than `bash`, as it provides a faster and more nimble shell. However, it contains less features than `bash` (such as being unable to handle multiple uplinks).

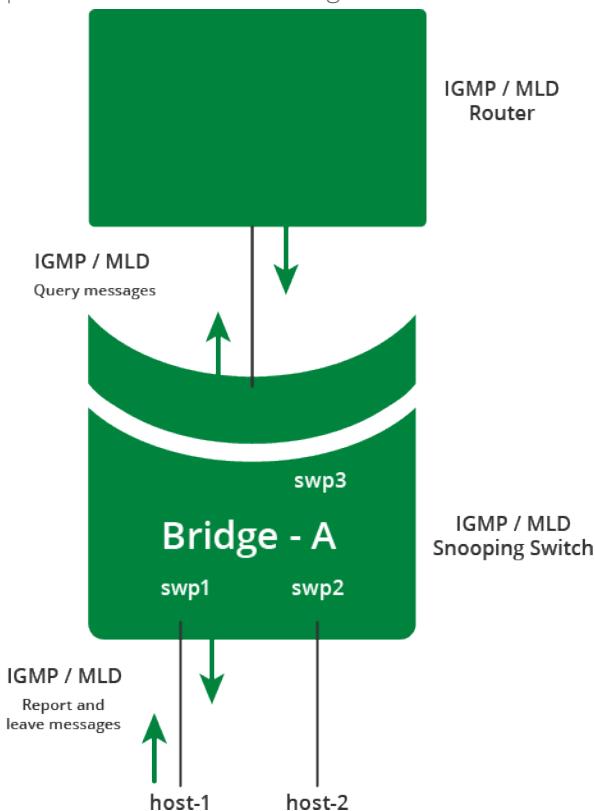
IGMP and MLD Snooping

IGMP (Internet Group Management Protocol) and MLD (Multicast Listener Discovery) snooping functionality is implemented in the bridge driver in the Cumulus Linux kernel and are enabled by default. IGMP snooping processes IGMP v1/v2/v3 reports received on a bridge port in a bridge to identify the hosts which would like to receive multicast traffic destined to that group.

When an IGMPv2 leave message is received, a group specific query is sent to identify if there are any other hosts interested in that group, before the group is deleted.

An IGMP query message received on a port is used to identify the port that is connected to a router and is interested in receiving multicast traffic.

MLD snooping processes MLD v1/v2 reports, queries and v1 done messages for IPv6 groups. If IGMP or MLD snooping is disabled, multicast traffic will be flooded to all the bridge ports in the bridge. The multicast group IP address is mapped to a multicast MAC address and a forwarding entry is created with a list of ports interested in receiving multicast traffic destined to that group.



Contents

This chapter covers ...

- Configuring IGMP/MLD Querier (see page 342)
- Disable IGMP and MLD Snooping (see page 343)
- Debugging IGMP/MLD Snooping (see page 344)
- Related Information (see page 346)



Configuring IGMP/MLD Querier

If no multicast router is sending queries to configure IGMP/MLD querier on the switch, you can add a configuration similar to the following in `/etc/network/interfaces`. To enable IGMP and MLD snooping for a bridge, set `bridge-mcquerier` to 1 in the bridge stanza. By default, the source IP address of IGMP queries is 0.0.0.0. To set the source IP address of the queries to be the bridge IP address, configure `bridge-mcqifaddr` 1.

For an explanation of the relevant parameters, see the `ifupdown-addons-interfaces` man page.

For a [VLAN-aware bridge \(see page 277\)](#), use a configuration like the following:

```
auto br0.100
vlan br0.100
  bridge-igmp-querier-src 123.1.1.1

auto br0
iface br0
  bridge-ports swp1 swp2 swp3
  bridge-vlan-aware yes
  bridge-vids 100 200
  bridge-pvid 1
  bridge-mcquerier 1
```

For a VLAN-aware bridge, like `br0` in the above example, to enable querier functionality for VLAN 100 in the bridge, set `bridge-mcquerier` to 1 in the bridge stanza and set `bridge-igmp-querier-src` to `123.1.1.1` in the `br0.100` stanza.

You can specify a range of VLANs as well. For example:

```
auto bridge.[1-200]
vlan bridge.[1-200]
  bridge-igmp-querier-src 123.1.1.1
```

For a bridge in [traditional mode \(see page 272\)](#), use a configuration like the following:

```
auto br0
iface br0
  address 192.0.2.10/24
  bridge-ports swp1 swp2 swp3
  bridge-vlan-aware no
  bridge-mcquerier 1
  bridge-mcqifaddr 1
```

Disable IGMP and MLD Snooping

To disable IGMP and MLD snooping:

- Set the `bridge-mcsnoop` value to 0.

ⓘ Example Disable IGMP MLD Snooping Configuration

The example NCLU commands below create a VLAN-aware bridge interface for a VRR-enabled network:

```
cumulus@switch:~$ net add bridge bridge mcsnoop no
cumulus@switch:~$ net pending
cumulus@switch:~$ net commit
```

The commands above add the `bridge-mcsnoop` line to the following example bridge in `/etc/network/interfaces`:

```
auto br0
iface br0
  bridge-ports swp1 swp2 swp3
  bridge-mcsnoop 0
  bridge-vlan-aware yes
  bridge-vids 100 200
  bridge-pvid 1
  bridge-mcquerier 1
```

Debugging IGMP/MLD Snooping

To get the IGMP/MLD snooping bridge state, run `brctl showstp <bridge>`:

```
cumulus@switch:~$ sudo brctl showstp br0
br0
bridge id          8000.7072cf8c272c
designated root    8000.7072cf8c272c
root port          0                                path
cost              0
max age           20.00                            bridge max
age               20.00
hello time        2.00                             bridge hello
time              2.00
forward delay     15.00                            bridge forward
delay             15.00
ageing time       300.00
hello timer       0.00                                tcn
timer             0.00
topology change timer 0.00
timer             263.70
hash elasticity   4096                            hash
max               4096
```



mc last member count	2	mc init query
count	2	
mc router	1	mc
snooping	1	
mc last member timer	1.00	mc membership
timer	260.00	
mc querier timer	255.00	mc query
interval	125.00	
mc response interval	10.00	mc init query
interval	31.25	
mc querier	0	mc query
ifaddr	0	
flags		
 swp1 (1)		
port id	8001	state
forwarding		
designated root	8000.7072cf8c272c	path
cost	2	
designated bridge	8000.7072cf8c272c	message age
timer	0.00	
designated port	8001	forward delay
timer	0.00	
designated cost	0	hold
timer	0.00	
mc router	1	mc fast
leave	0	
flags		
 swp2 (2)		
port id	8002	state
forwarding		
designated root	8000.7072cf8c272c	path
cost	2	
designated bridge	8000.7072cf8c272c	message age
timer	0.00	
designated port	8002	forward delay
timer	0.00	
designated cost	0	hold
timer	0.00	
mc router	1	mc fast
leave	0	
flags		
 swp3 (3)		
port id	8003	state
forwarding		
designated root	8000.7072cf8c272c	path
cost	2	
designated bridge	8000.7072cf8c272c	message age
timer	0.00	



designated port	8003	forward delay
timer	8.98	
designated cost	0	hold
timer	0.00	
mc router	1	mc fast
leave	0	
flags		

To get the groups and bridge port state, use the `bridge mdb show` command. To display router ports and group information use the `bridge -d -s mdb show` command:

```
cumulus@switch:~$ sudo bridge -d -s mdb show
dev br0 port swp2 grp 234.10.10.10 temp 241.67
dev br0 port swp1 grp 238.39.20.86 permanent 0.00
dev br0 port swp1 grp 234.1.1.1 temp 235.43
dev br0 port swp2 grp ff1a::9 permanent 0.00
router ports on br0: swp3
```

Related Information

- www.linuxfoundation.org/collaborate/workgroups/networking/bridge#Snooping
- tools.ietf.org/html/rfc4541
- en.wikipedia.org/wiki/IGMP_snooping
- tools.ietf.org/rfc/rfc2236.txt
- tools.ietf.org/html/rfc3376
- tools.ietf.org/search/rfc2710
- tools.ietf.org/html/rfc3810



Network Virtualization

Cumulus Linux supports these forms of [network virtualization](#):

VXLAN (Virtual Extensible LAN) is a standard overlay protocol that abstracts logical virtual networks from the physical network underneath. You can deploy simple and scalable layer 3 Clos architectures while extending layer 2 segments over that layer 3 network.

VXLAN uses a VLAN-like encapsulation technique to encapsulate MAC-based layer 2 Ethernet frames within layer 3 UDP packets. Each virtual network is a VXLAN logical L2 segment. VXLAN scales to 16 million segments – a 24-bit VXLAN network identifier (VNI ID) in the VXLAN header – for multi-tenancy.

Hosts on a given virtual network are joined together through an overlay protocol that initiates and terminates tunnels at the edge of the multi-tenant network, typically the hypervisor vSwitch or top of rack. These edge points are the VXLAN tunnel end points (VTEP).

Cumulus Linux can initiate and terminate VTEPs in hardware and supports wire-rate VXLAN. VXLAN provides an efficient hashing scheme across IP fabric during the encapsulation process; the source UDP port is unique, with the hash based on L2-L4 information from the original frame. The UDP destination port is the standard port 4789.

Cumulus Linux includes the native Linux VXLAN kernel support and integrates with controller-based overlay solutions like [VMware NSX](#) (see page 348) and [Midokura MidoNet](#) (see page 361).

VXLAN is supported only on switches in the [Cumulus Linux HCL](#) using the Broadcom Tomahawk, Trident II+ and Trident II chipsets as well as the Mellanox Spectrum chipset.



VXLAN encapsulation over layer 3 subinterfaces (for example, swp3.111) is not supported. Therefore, VXLAN uplinks should be only configured as layer 3 interfaces without any subinterfaces (for example, swp3).

Furthermore the VXLAN tunnel endpoints cannot share a common subnet; there must be at least one layer 3 hop between the VXLAN source and destination.

Caveats/Errata

Cut-through Mode

Cut-through mode is disabled in Cumulus Linux by default. With cut-through mode enabled and link pause is asserted, Cumulus Linux generates a TOVR and TUFL ERROR; certain error counters increment on a given physical port.

```
cumulus@switch:~$ sudo ethtool -S swp49 | grep Error
HwIfInDot3LengthErrors: 0
HwIfInErrors: 0
HwIfInDot3FrameErrors: 0
SoftInErrors: 0
SoftInFrameErrors: 0
HwIfOutErrors: 35495749
SoftOutErrors: 0
```



```
cumulus@switch:~$ sudo ethtool -S swp50 | grep Error
HwIfInDot3LengthErrors: 3038098
HwIfInErrors: 297595762
HwIfInDot3FrameErrors: 293710518
```

To work around this issue, disable link pause or disable cut-through in `/etc/cumulus/datapath/traffic.conf`:

To disable link pause, comment out the `link_pause*` section in `/etc/cumulus/datapath/traffic.conf`:

```
cumulus@switch:~$ sudo nano /etc/cumulus/datapath/traffic.conf
#link_pause.port_group_list = [port_group_0]
#link_pause.port_group_0.port_set = swp45-swp54
#link_pause.port_group_0.rx_enable = true
#link_pause.port_group_0.tx_enable = true
```

To enable store and forward switching, set `cut_through_enable` to false in `/etc/cumulus/datapath/traffic.conf`:

```
cumulus@switch:~$ sudo nano /etc/cumulus/datapath/traffic.conf
cut_through_enable = false
```

MTU Size for Virtual Network Interfaces

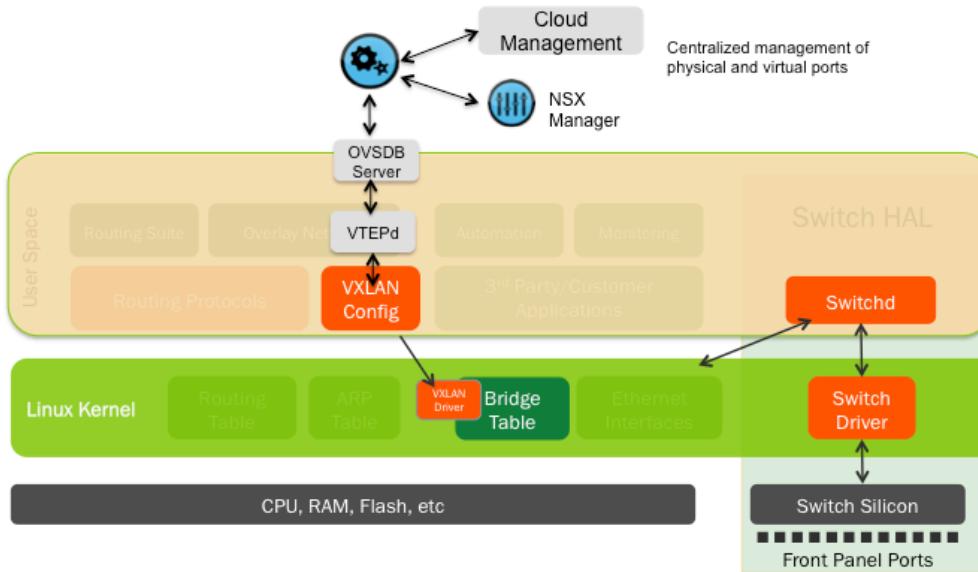
The maximum transmission unit (MTU) size for a virtual network interface should be 50 bytes smaller than the MTU for the physical interfaces on the switch. For more information, read [Layer 1 and Switch Port Attributes](#) (see page 211).

Useful Links

- VXLAN IETF draft
- ovsdb-server

Integrating with VMware NSX

Switches running Cumulus Linux can integrate with VMware NSX to act as VTEP gateways. The VMware NSX controller provides consistent provisioning across virtual and physical server infrastructures.



Contents

This chapter covers ...

- Getting Started (see page 349)
 - Caveats and Errata (see page 350)
- Bootstrapping the NSX Integration (see page 350)
 - Enabling the openvswitch-vtep Package (see page 350)
 - Using the Bootstrapping Script (see page 351)
 - Manually Bootstrapping the NSX Integration (see page 352)
 - Generating the Credentials Certificate (see page 352)
 - Configuring the Switch as a VTEP Gateway (see page 353)
- Configuring the Transport Layer (see page 355)
- Configuring the Logical Layer (see page 356)
 - Defining Logical Switches (see page 357)
 - Defining Logical Switch Ports (see page 358)
- Verifying the VXLAN Configuration (see page 360)
- Troubleshooting VXLANs in NSX (see page 360)

Getting Started

Before you integrate VXLANs with NSX, make sure you have the following components:

- A switch (L2 gateway) with a Tomahawk, Trident II+ or Trident II chipset running Cumulus Linux
- OVSDB server (ovsdb-server), included in Cumulus Linux
- VTEPd (ovs-vtep), included in Cumulus Linux

Integrating a VXLAN with NSX involves:

- Bootstrapping the NSX Integration



- Configuring the Transport Layer
- Configuring the Logical Layer
- Verifying the VXLAN Configuration

Caveats and Errata

- As mentioned in [Network Virtualization \(see page 346\)](#), the switches with the source and destination VTEPs cannot reside on the same subnet; there must be at least one layer 3 hop between the VXLAN source and destination.
- There is no support for VXLAN routing in the Tomahawk, Trident II+ and Trident II chips; use a loopback interface or external router.
- The `ovsdb-server` cannot select the loopback interface as the source IP address, causing TOR registration to the controller to fail. To work around this issue, run:

```
cumulus@switch:~$ net add bgp redistribute connected  
cumulus@switch:~$ net pending  
cumulus@switch:~$ net commit
```

- Do not use 0 or 16777215 as the VNI ID, as they are reserved values under Cumulus Linux.
- For more information about NSX, see the VMware NSX User Guide, version 4.0.0 or later.

Bootstrapping the NSX Integration

Before you start configuring the gateway service and logical switches and ports that comprise the VXLAN, you need to complete some steps to bootstrap the process. You need to do the bootstrapping just once, before you begin the integration.

Enabling the `openvswitch-vtep` Package

Before you start bootstrapping the integration, you need to enable the `openvswitch-vtep` package, as it is disabled by default in Cumulus Linux.

1. In `/etc/default/openvswitch-vtep`, change the `START` option from `no` to `yes`:

```
cumulus@switch$ cat /etc/default/openvswitch-vtep  
# This is a POSIX shell fragment -*- sh -*-  
  
# Start openvswitch at boot ? yes/no  
START=yes  
  
# FORCE_COREFILES: If 'yes' then core files will be enabled.  
# FORCE_COREFILES=yes  
  
# BRCOMPAT: If 'yes' and the openvswitch-brcompat package is  
installed, then  
# Linux bridge compatibility will be enabled.  
# BRCOMPAT=no
```



2. Start the daemon:

```
cumulus@switch$ sudo systemctl start openvswitch-vtep.service
```

Using the Bootstrapping Script

A script is available so you can do the bootstrapping automatically. For information, read `man vtep-bootstrap`. The output of the script is displayed here:

```
cumulus@vtep7: ~
cumulus@vtep7$ sudo vtep-bootstrap --credentials-path /var/lib/openvswitch vtep7 192.168.100.17 172.1
6.20.157 192.168.100.157
Executed:
  create certificate on a switch, to be used for authentication with controller
  (),
Executed:
  sign certificate
    (vtep7-red.pem      Fri Jan 17 18:04:33 UTC 2014
     fingerprint 6f443eb8445317b545d8564c2ae9636ea0a9104a).
Executed:
  define physical switch
  (),
Executed:
  define NSX controller IP address in OVSDB
  (),
Executed:
  define local tunnel IP address on the switch
  (),
Executed:
  define management IP address on the switch
  (),
Executed:
  restart a service
    (Killing ovs-vtep (4973).
Killing ovsdb-server (4969).
Starting ovsdb-server.
Starting ovs-vtep (4973).
cumulus@vtep7$)
cumulus@vtep7$ ps xa | grep ovsdb-server
5184 pts/0    S+    0:00 ovsdb-server: monitoring pid 5185 (healthy)

5185 ?      S+    0:00 ovsdb-server /var/lib/openvswitch/conf.db -vANY:CONSOLE:EMER -vANY:SYSLOG:
ERR -vANY:FILE:INFO --remote=punix:/var/run/openvswitch/db.sock --remote=db:Global_managers --remote=
ptcp:6633: --private-key=/var/lib/openvswitch/vtep7-prvkey.pem --certificate=/var/lib/openvswitch/vt
ep7-cert.pem --bootstrap-ca-cert=/var/lib/openvswitch/controller.cacert --no-chdir --log-file=/var/lo
g/openvswitch/ovsdb-server.log --pidfile=/var/run/openvswitch/ovsdb-server.pid --detach --monitor
5436 pts/0    S+    0:00 grep ovsdb-server
cumulus@vtep7$
```

In the above example, the following information was passed to the `vtep-bootstrap` script:

- `--credentials-path /var/lib/openvswitch`: Is the path to where the certificate and key pairs for authenticating with the NSX controller are stored.
- `vtep7`: is the ID for the VTEP.
- `192.168.100.17`: is the IP address of the NSX controller.
- `172.16.20.157`: is the datapath IP address of the VTEP.
- `192.168.100.157`: is the IP address of the management interface on the switch.

These IP addresses will be used throughout the rest of the examples below.



Manually Bootstrapping the NSX Integration

If you don't use the script, then you must:

- Initialize the OVS database instance
- Generate a certificate and key pair for authentication by NSX
- Configure a switch as a VTEP gateway

These steps are described next.

Generating the Credentials Certificate

First, in Cumulus Linux, you must generate a certificate that the NSX controller uses for authentication.

1. In a terminal session connected to the switch, run the following commands:

```
cumulus@switch:~$ sudo ovs-pki init
Creating controllerca...
Creating switchca...
cumulus@switch:~$ sudo ovs-pki req+sign cumulus

cumulus@switch:~$ ls -l
total 12
-rw-r--r-- 1 root root 4028 Oct 23 05:32 cumulus-cert.pem
-rw----- 1 root root 1679 Oct 23 05:32 cumulus-privkey.pem
-rw-r--r-- 1 root root 3585 Oct 23 05:32 cumulus-req.pem
```

2. In /usr/share/openvswitch/scripts/ovs-ctl-vtep, make sure the lines containing **private-key**, **certificate** and **bootstrap-ca-cert** point to the correct files; **bootstrap-ca-cert** is obtained dynamically the first time the switch talks to the controller:

```
# Start ovsdb-server.
set ovsdb-server "$DB_FILE"
set "$@" -vANY:CONSOLE:EMER -vANY:SYSLOG:ERR -vANY:FILE:INFO
set "$@" --remote=unix:"$DB_SOCK"
set "$@" --remote=db:Global,managers
set "$@" --remote=ptcp:$LOCALIP:6633
set "$@" --private-key=/root/cumulus-privkey.pem
set "$@" --certificate=/root/cumulus-cert.pem
set "$@" --bootstrap-ca-cert=/root/controller.cacert
```

If files have been moved or regenerated, restart the OVSDB server and vtep daemon:

```
cumulus@switch:~$ sudo systemctl restart openvswitch-vtep.service
```



3. Define the NSX controller cluster IP address in OVSDB. This causes the OVSDB server to start contacting the NSX controller:

```
cumulus@switch:~$ sudo vtep-ctl set-manager ssl:192.168.100.17:66  
32
```

4. Define the local IP address on the VTEP for VXLAN tunnel termination. First, find the physical switch name as recorded in OVSDB:

```
cumulus@switch:~$ sudo vtep-ctl list-ps  
vtep7
```

Then set the tunnel source IP address of the VTEP. This is the datapath address of the VTEP, which is typically an address on a loopback interface on the switch that is reachable from the underlying L3 network:

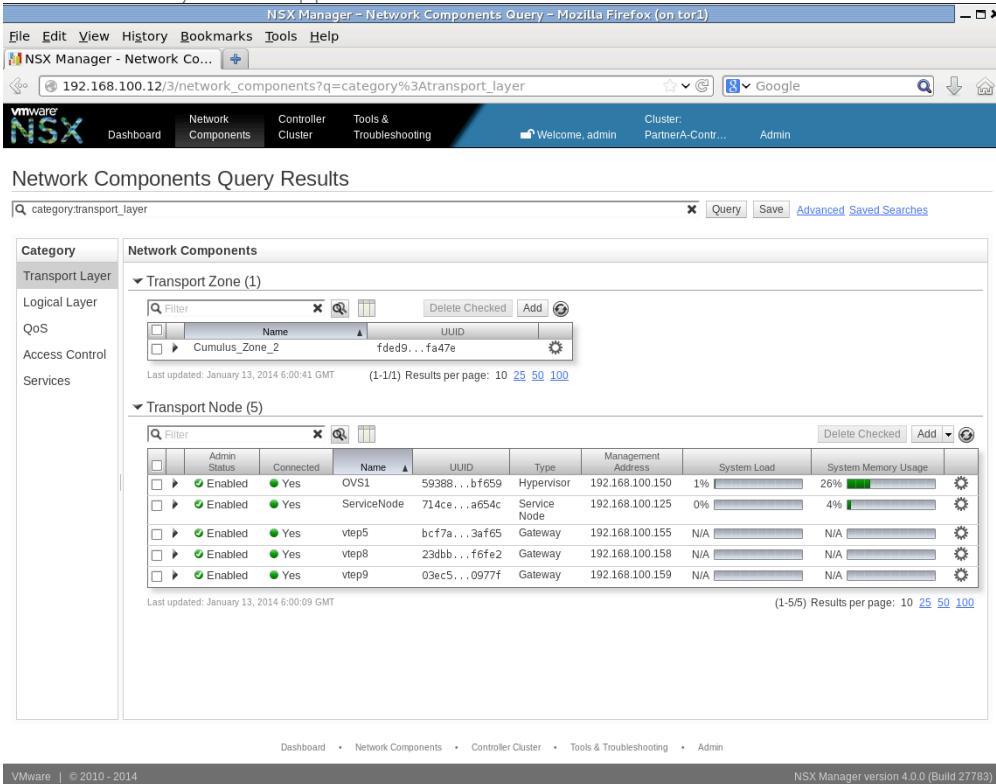
```
cumulus@switch:~$ sudo vtep-ctl set Physical_Switch vtep7  
tunnel_ips=172.16.20.157
```

Once you finish generating the certificate, keep the terminal session active, as you need to paste the certificate into NSX Manager when you configure the VTEP gateway.

Configuring the Switch as a VTEP Gateway

After you create a certificate, connect to NSX Manager in a browser to configure a Cumulus Linux switch as a VTEP gateway. In this example, the IP address of the NSX manager is 192.168.100.12.

- In NSX Manager, add a new gateway. Click the **Network Components** tab, then the **Transport Layer** category. Under **Transport Node**, click **Add**, then select **Manually Enter All Fields**. The Create Gateway wizard appears.



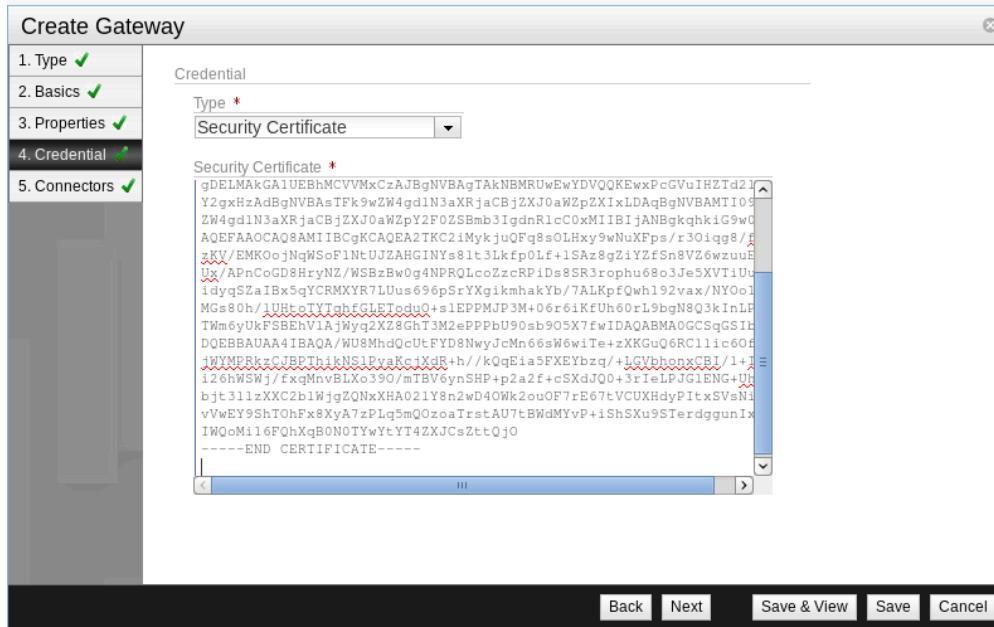
The screenshot shows the NSX Manager interface with the following details:

- Network Components Query Results**
- Category:** Transport Layer
- Transport Zone (1):**
 - Filter: Name, UUID
 - Items: Cumulus_Zone_2 (fded9...fa47e)
 - Last updated: January 13, 2014 6:00:41 GMT
 - (1-1) Results per page: 10, 25, 50, 100
- Transport Node (5):**
 - Filter: Admin Status, Connected, Name, UUID, Type, Management Address, System Load, System Memory Usage
 - Items:
 - OVS1 (5938e...bf659): Hypervisor, 192.168.100.150, 1%, 26%
 - ServiceNode (714ce...a654c): Service Node, 192.168.100.125, 0%, 4%
 - vtep5 (bcf7a...3af65): Gateway, 192.168.100.155, N/A, N/A
 - vtep8 (23dbb...f6fe2): Gateway, 192.168.100.158, N/A, N/A
 - vtep9 (03ec5...0977f): Gateway, 192.168.100.159, N/A, N/A
 - Last updated: January 13, 2014 6:00:09 GMT
 - (1-5) Results per page: 10, 25, 50, 100

- In the Create Gateway dialog, select **Gateway** for the **Transport Node Type**, then click **Next**.
- In the **Display Name** field, give the gateway a name, then click **Next**.
- Enable the VTEP service. Select the **VTEP Enabled** checkbox, then click **Next**.
- From the terminal session connected to the switch where you generated the certificate, copy the certificate and paste it into the **Security Certificate** text field. Copy only the bottom portion, including the `BEGIN CERTIFICATE` and `END CERTIFICATE` lines. For example, copy all the highlighted text in the terminal:

```
ubuntu@tor1: ~
87:3f:ea;76:8b:67:fe:71:25:dd:25:0d:3e:de:b2:1e:2c:f2;
46:94:43:46:f9:48:43:6e:3b:77:96:5e:d7:5c:2d:9b:95:68;
e0:65:03:71:5c:70:34:da:56:3c:9f:6e:03:e0:4e:da:8b;
8e:17:ba:c4:eb:bb:59:09:45:c7:77:23:c8:b7:14:95:b0:d8;
ba:bd:5c:04:63:41:a1:c4:e8:45:c7:c5:f2:03:b0:c7:2e:ae;
66:40:ec:e8:69:3a:ec:b4:05:3b:b4:15:9d:31:b8:c7:fa:24;
a1:49:7b:bd:49:37:ab:76:08:2e:9c:8c:44:21:64:28:32:2d;
7a:19:08:57:a8:1d:0d:0d:36:30:02:db:13:e1:95:c9:0a:c6;
6d:b5:08:ce
-----BEGIN CERTIFICATE-----
MIIBdCCA18CAQYxDQYJKoZIhvNA0gEBOAwgExCzA1BgNVBAYTA1VTM0swCQD
M0Q1EwJJOtEVMBGALUECHMT3B1b1B2U3dpdNoMREwDwYDVQQLEuhzd210V2hj
YtE7MDkgA1UEAxMt1ZT1HN3aXRjaGhNIEBEN1cnRpZnlyXR11CsgyIDEzER1
YgAuMyAxNzowOByNCkuUhcmNTThxjIzTcxMzAxJwhNM7Q:MjIzMtCxLxzhdJCB
gDELMAgCA1UEBhMCVWhCcxBJBgNVBAsTkhkNBMRUwEYDVQKExwPcGVwH2Td210
Y2gxHzAdBgNVBAsTFk9uZl44gdIN3aXrJaC8ZXJ0alZpZXIxLxDqBgwVBANT109w
Zl44gdIN3aXrJaC8ZXJ0alZpZl44gdIN3aXrJaC8ZXJ0alZpZXIxLxDqBgwVBANT109w
Zl44gdIN3aXrJaC8ZXJ0alZpZl44gdIN3aXrJaC8ZXJ0alZpZXIxLxDqBgwVBANT109w
90EFa0DCAQ8M1IBCaKCAQEA2TKC2iMyjQFq8s0LHxy3wNuXfps/r30iq98/FP
zKV/EMK0oJNdsf1htUJZAHGINy81t3LkfplF+1Sa28g21YzFSn8VZ6zuuE8
Jx/APnCoGd8hNZ/WSBzBw0g4NPnQlco2zcRP1DsS93r0phu6803jeXVt1UuZ
jdqg52a1bx5qYCRMXYR7LUsos96p5TYgimkahYb/7HLkpQ0uh192vax/YNy0li
MG80h1UHt0TYTghGLETodu0+sLEPPNJP3M+06:61kFuhs6L9bgwB3kInLPN
TlW6UkFSEhV1Ajjuqg2XZ8GhT3M2ePPBbJ9Osbs05X7fuID9QBMhAGGCSgGS1b3
DOEBBaUa44IBa09/wU8MhddeUfYD8NwJcm6s6lwjTe+zXGu6RC11ic60f9
jUyYMPrkzCJBPThiKNSIPyaKcjXdr-/h/KQeia5FXEYbzq/+LGvbhnxCB1/1+1G
i26hUsUjfxMmvlBLko390/wTBw6unSHP+2a2f+c5xdJ00+3r1eLPJG1ENG+UnD
bJt311zXc2b1wJg2UN:XHA021Y8w2u040lk2uoF7rEB7tVCUxHdyItxvSvN16
vUyEY9ShTOhfXuA7zPLq5m02ozaTrstAU7tBldMyvP+ShSXu95STerDggunIxE
IWQoM16F0hkaBONOTyYtYT4ZJC0s2ztQj0
-----END CERTIFICATE-----
root@vtep-1:~#
```

And paste it into NSX Manager:



Then click **Next**.

6. In the Connectors dialog, click **Add Connector** to add a transport connector. This defines the tunnel endpoint that terminates the VXLAN tunnel and connects NSX to the physical gateway. You must choose a tunnel **Transport Type** of **VXLAN**. Choose an existing transport zone for the connector, or click **Create** to create a new transport zone.
7. Define the connector's IP address (that is, the underlay IP address on the switch for tunnel termination).
8. Click **OK** to save the connector, then click **Save** to save the gateway.

Once communication is established between the switch and the controller, a `controller.cacert` file will be downloaded onto the switch.

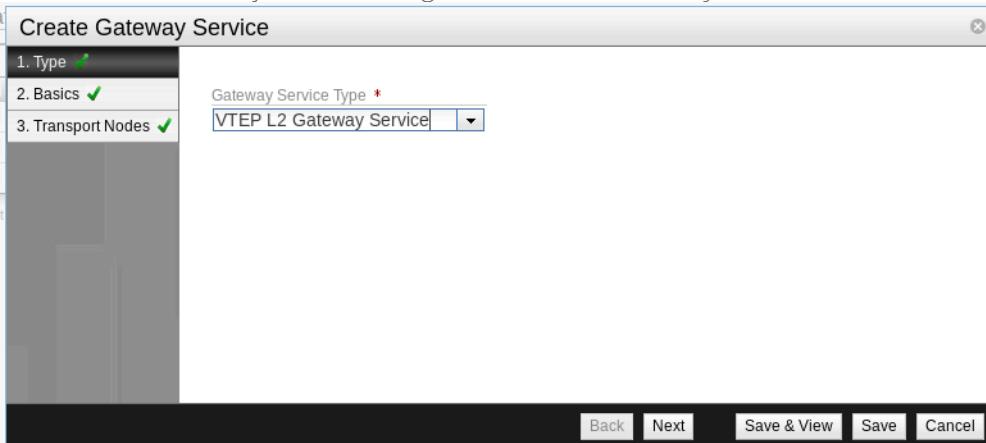
Verify the controller and switch handshake is successful. In a terminal connected to the switch, run this command:

```
cumulus@switch:~$ sudo ovswitch-client dump -f list | grep -A 7 "Manager"
Manager table
_uuid : 505f32af-9acb-4182-a315-022e405aa479
_inactivity_probe : 30000
_is_connected : true
_max_backoff : []
_other_config : {}
_status : {sec_since_connect="18223",
sec_since_disconnect="18225", state=ACTIVE}
_target : "ssl:192.168.100.17:6632"
```

Configuring the Transport Layer

After you finish bootstrapping the NSX integration, you need to configure the transport layer. For each host-facing switch port that is to be associated with a VXLAN instance, define a **Gateway Service** for the port.

1. In NSX Manager, add a new gateway service. Click the **Network Components** tab, then the **Services** category. Under **Gateway Service**, click **Add**. The Create Gateway Service wizard appears.
2. In the Create Gateway Service dialog, select **VTEP L2 Gateway Service** as the **Gateway Service Type**.



3. Give the service a **Display Name** to represent the VTEP in NSX.
4. Click **Add Gateway** to associate the service with the gateway you created earlier.
5. In the **Transport Node** field, choose the name of the gateway you created earlier.
6. In the **Port ID** field, choose the physical port on the gateway (for example, swp10) that will connect to a logical L2 segment and carry data traffic.
7. Click **OK** to save this gateway in the service, then click **Save** to save the gateway service.

The gateway service shows up as type **VTEP L2** in NSX.

Name	UUID	Type	# Transport Nodes
vtep-1-swp2s0	9db3e...d3ea3	VTEP L2	1
vtep-1-swp2s1	343a7...00ce6	VTEP L2	1
vtep5-swp1	958f1...a7ddc	VTEP L2	1
vtep8-swp1	f02ba...29ffb	VTEP L2	1
vtep9-swp1	c9e1...aed98	VTEP L2	1

Next, you will configure the logical layer on NSX.

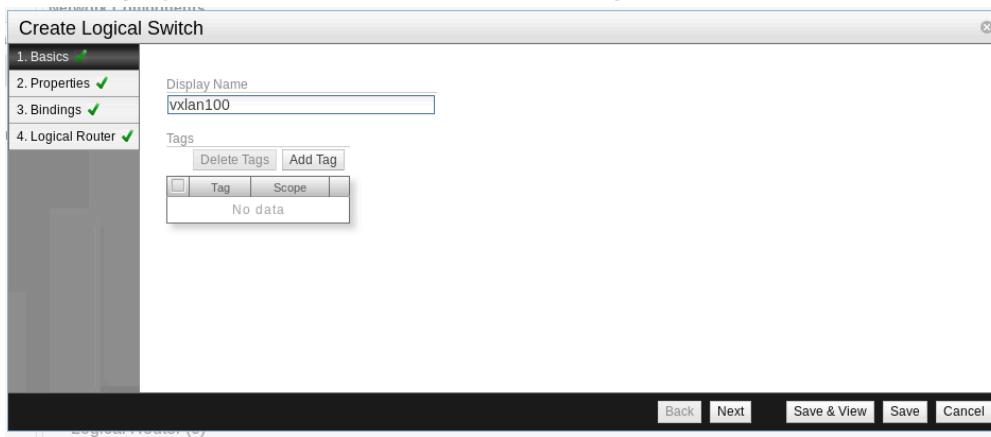
Configuring the Logical Layer

To complete the integration with NSX, you need to configure the logical layer, which requires defining a logical switch (the VXLAN instance) and all the logical ports needed.

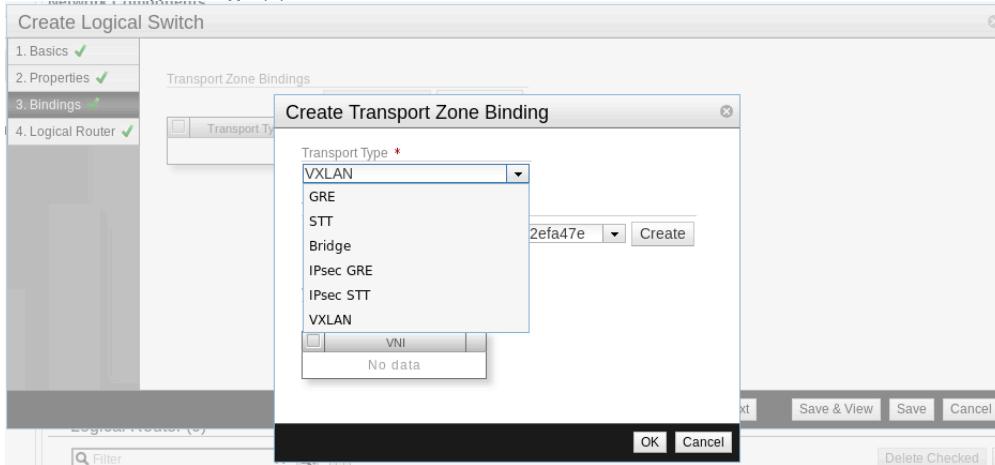
Defining Logical Switches

To define the logical switch, do the following:

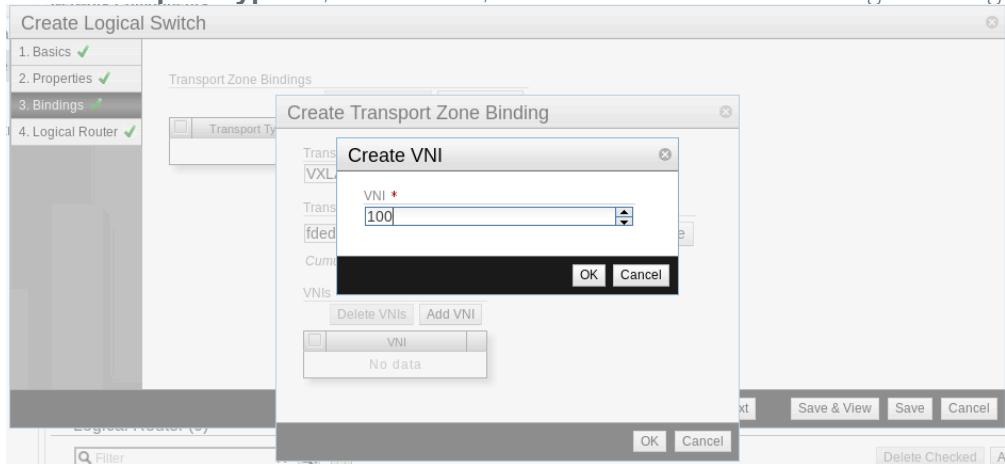
1. In NSX Manager, add a new logical switch. Click the **Network Components** tab, then the **Logical Layer** category. Under **Logical Switch**, click **Add**. The Create Logical Switch wizard appears.
2. In the **Display Name** field, enter a name for the logical switch, then click **Next**.



3. Under **Replication Mode**, select **Service Nodes**, then click **Next**.
4. Specify the transport zone bindings for the logical switch. Click **Add Binding**. The Create Transport Zone Binding dialog appears.



5. In the **Transport Type** list, select **VXLAN**, then click **OK** to add the binding to the logical switch.



6. In the **VNI** field, assign the switch a VNI ID, then click **OK**.



Do not use 0 or 16777215 as the VNI ID, as they are reserved values under Cumulus Linux.

7. Click **Save** to save the logical switch configuration.

Category	Name	UUID	Logical Switch Ports	Logical Router	Transport Zone Bindings	Replication Mode	Port Isolation
Logical Layer	vxian100	47a37...b1790	0	-	1	Service Node	Disabled
	LS-B	80b3e...36454	1	-	1	Service Node	Disabled
	LS-A	c1be3...fb689	1	-	1	Service Node	Disabled

Category	Admin	Link	Fabric	Message	Name	Port	Switch Name	UUID	Attachment	Attached MAC
Logical Layer	Up	Up	Up	-	v1	1	LS-A	3db01...e2afb	VIF:5e690...42dfb	52:54:00:15:44:34
	Up	Up	Up	-	v2	3	LS-B	a080d...b5588	VIF:0afff...a8571	52:54:00:92:25:bd

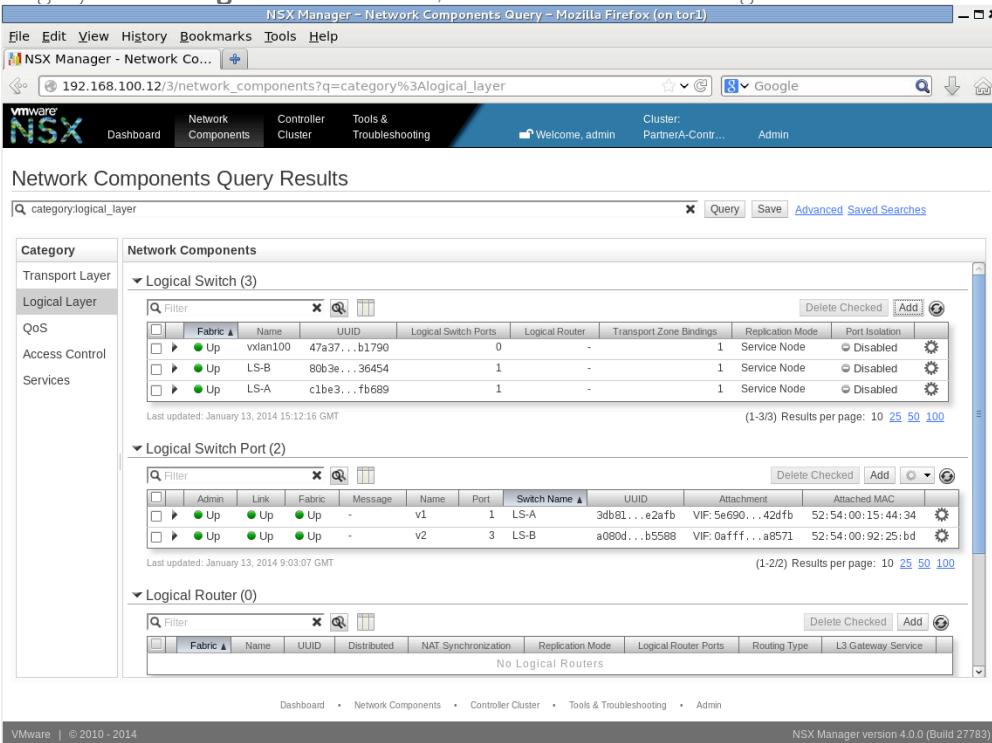
Category	Name	UUID	Distributed	NAT Synchronization	Replication Mode	Logical Router Ports	Routing Type	L3 Gateway Service
Logical Router								No Logical Routers

Defining Logical Switch Ports

As the final step, define the logical switch ports. They can be virtual machine VIF interfaces from a registered OVS, or a VTEP gateway service instance on this switch, as defined above in the Configuring the Transport Layer. A VLAN binding can be defined for each VTEP gateway service associated with the particular logical switch.

To define the logical switch ports, do the following:

- In NSX Manager, add a new logical switch port. Click the **Network Components** tab, then the **Logical Layer** category. Under **Logical Switch Port**, click **Add**. The Create Logical Switch Port



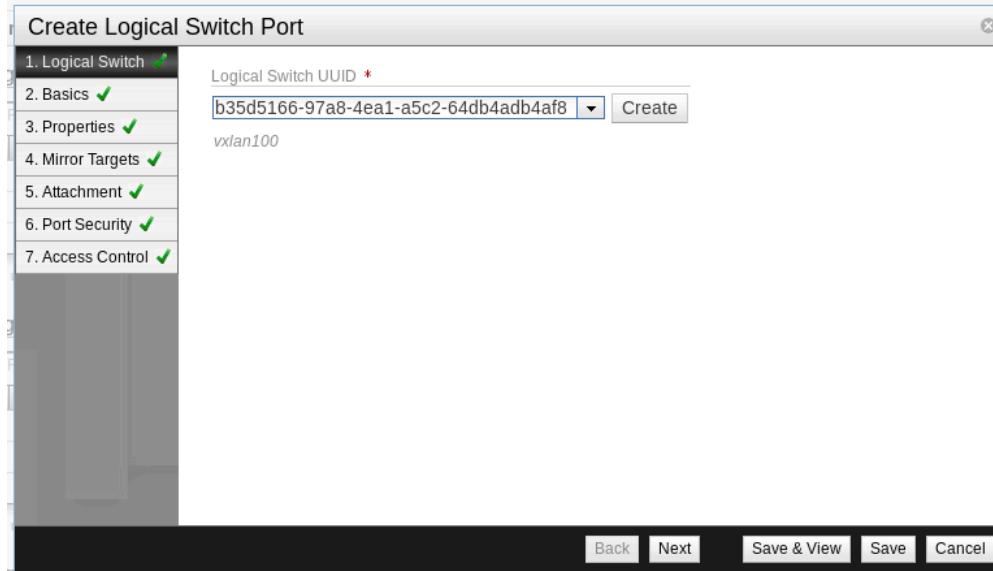
The screenshot shows the NSX Manager interface with the Network Components Query Results page open. The URL is 192.168.100.12/3/network_components?q=category%3Alogical_layer. The page displays two tables: Logical Switch (3) and Logical Switch Port (2). The Logical Switch table has three entries: vxlan100, LS-B, and LS-A. The Logical Switch Port table has two entries: v1 and v2. Both tables include columns for Fabric, Name, UUID, Logical Switch Ports, Logical Router, Transport Zone Bindings, Replication Mode, and Port Isolation.

Fabric	Name	UUID	Logical Switch Ports	Logical Router	Transport Zone Bindings	Replication Mode	Port Isolation
vxlan100	47a37...b1790		0	-	1	Service Node	Disabled
LS-B	80b3e...36454		1	-	1	Service Node	Disabled
LS-A	c1be3...fb689		1	-	1	Service Node	Disabled

Admin	Link	Fabric	Message	Name	Port	Switch Name	UUID	Attachment	Attached MAC
Up	Up	Up	-	v1	1	LS-A	3db81...62afb	VIF: 5e690...42dfb	52:54:00:15:44:34
Up	Up	Up	-	v2	3	LS-B	a080d...b5588	VIF: 0aff...a8571	52:54:00:92:25:bd

wizard appears.

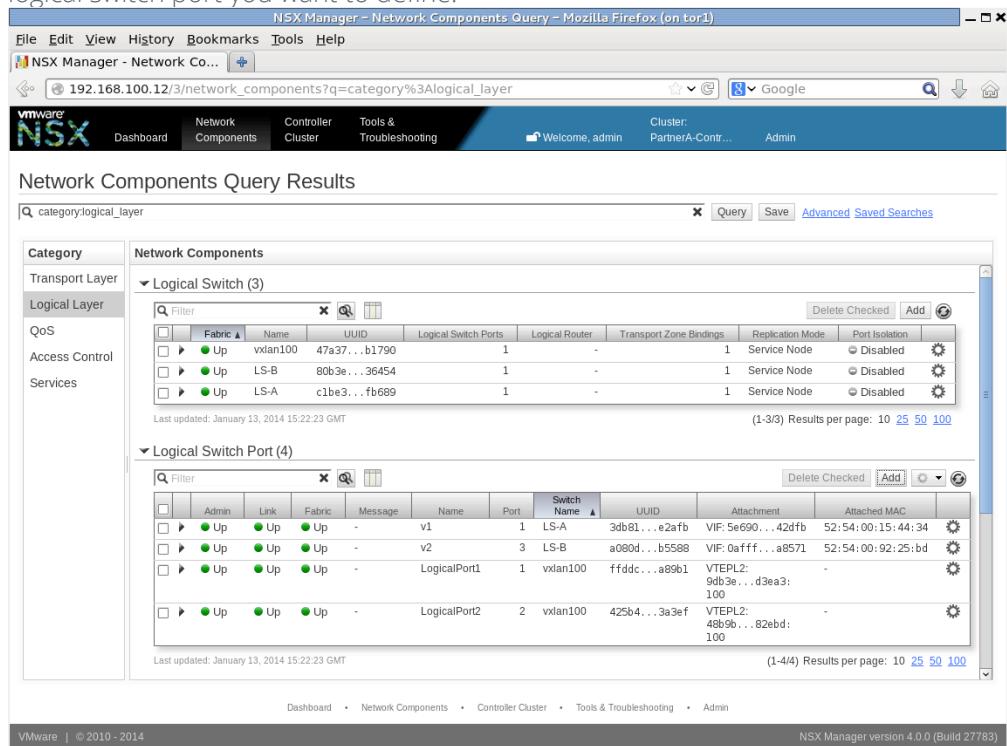
- In the **Logical Switch UUID** list, select the logical switch you created above, then click **Create**.



The screenshot shows the 'Create Logical Switch Port' wizard at step 2. Basics. The left sidebar lists steps 1 through 7. Step 2 is selected. The main area shows a dropdown for 'Logical Switch UUID' containing 'b35d5166-97a8-4ea1-a5c2-64db4adb4af8'. Below it is a 'Display Name' field with 'vxlan100'. At the bottom are 'Back', 'Next', 'Save & View', 'Save', and 'Cancel' buttons.

- In the **Display Name** field, give the port a name that indicates it is the port that connects the gateway, then click **Next**.
- In the **Attachment Type** list, select **VTEP L2 Gateway**.
- In the **VTEP L2 Gateway Service UUID** list, choose the name of the gateway service you created earlier.
- In the **VLAN** list, you can optionally choose a VLAN if you wish to connect only traffic on a specific VLAN of the physical network. Leave it blank to handle all traffic.

- Click **Save** to save the logical switch port. Connectivity is established. Repeat this procedure for each logical switch port you want to define.



The screenshot shows the NSX Manager interface for Network Components Query. The left sidebar has categories: Transport Layer, Logical Layer (selected), QoS, Access Control, and Services. The main area displays two tables:

- Logical Switch (3)**: Shows three entries: vxlan100 (Port 1, Logical Router -, Transport Zone Bindings 1, Service Node), LS-B (Port 1, Logical Router -, Transport Zone Bindings 1, Service Node), and LS-A (Port 1, Logical Router -, Transport Zone Bindings 1, Service Node). All ports are Up.
- Logical Switch Port (4)**: Shows four entries: v1 (Port 1, Logical Router LS-A, Attachment 3db81...e2af, Attached MAC 52:54:00:15:44:34), v2 (Port 3, Logical Router LS-B, Attachment a080d...b5588, Attached MAC 52:54:00:92:25:bd), LogicalPort1 (Port 1, Logical Router vxlan100, Attachment ffddc...a89b1, Attached MAC VTEPL2:9db3e...d3ea3:100), and LogicalPort2 (Port 2, Logical Router vxlan100, Attachment 425b4...3a3ef, Attached MAC VTEPL2:489b...82ebd:100). All ports are Up.

Both tables have a "Delete Checked" button and a "Add" button. The bottom of the page shows navigation links: Dashboard, Network Components, Controller Cluster, Tools & Troubleshooting, Admin, and the footer: VMware | © 2010 - 2014, NSX Manager version 4.0.0 (Build 27783).

Verifying the VXLAN Configuration

Once configured, you can verify the VXLAN configuration using these Cumulus Linux commands in a terminal connected to the switch:

```
cumulus@switch1:~$ sudo ip -d link show vxln100
71: vxln100: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue
    master br-vxln100 state UNKNOWN mode DEFAULT
        link/ether d2:ca:78:bb:7c:9b brd ff:ff:ff:ff:ff:ff
        vxlan id 100 local 172.16.20.157 port 32768 61000 nolearning
        ageing 300 svchnode 172.16.21.125
```

or

```
cumulus@switch1:~$ sudo bridge fdb show
52:54:00:ae:2a:e0 dev vxln100 dst 172.16.21.150 self permanent
d2:ca:78:bb:7c:9b dev vxln100 permanent
90:e2:ba:3f:ce:34 dev swp2s1.100
90:e2:ba:3f:ce:35 dev swp2s0.100
44:38:39:00:48:0e dev swp2s1.100 permanent
44:38:39:00:48:0d dev swp2s0.100 permanent
```

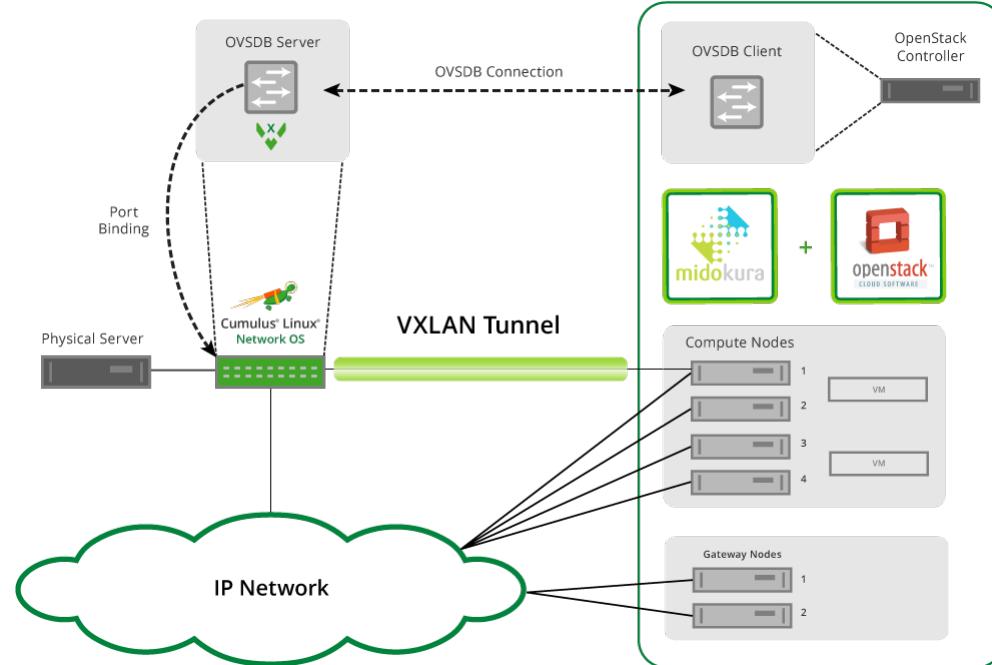
Troubleshooting VXLANs in NSX

Use `ovsdb-client dump` to troubleshoot issues on the switch. It verifies that the controller and switch handshake is successful. This command works only for VXLANs integrated with NSX:

```
cumulus@switch:~$ sudo ovsdb-client dump -f list | grep -A 7 "Manager"
Manager table
_uuid              : 505f32af-9acb-4182-a315-022e405aa479
_inactivity_probe   : 30000
_is_connected       : true
_max_backoff        : []
_other_config        :
_status             : {sec_since_connect="18223",
                     sec_since_disconnect="18225", state=ACTIVE}
_target              : "ssl:192.168.100.17:6632"
```

Integrating Hardware VTEPs with Midokura MidoNet and OpenStack

Cumulus Linux seamlessly integrates with the MidoNet OpenStack infrastructure, where the switches provide the VTEP gateway for terminating VXLAN tunnels from within MidoNet. MidoNet connects to the OVSDB server running on the Cumulus Linux switch, and exchanges information about the VTEPs and MAC addresses associated with the OpenStack Neutron networks. This provides seamless Ethernet connectivity between virtual and physical server infrastructures.



Contents

This chapter covers ...



- [Getting Started \(see page 362\)](#)
 - [Caveats and Errata \(see page 362\)](#)
 - [Preparing for the MidoNet Integration \(see page 362\)](#)
 - [Enabling the openvswitch-vtep Package \(see page 363\)](#)
 - [Bootstrapping the OVSDB Server and VTEP \(see page 363\)](#)
 - [Automating with the Bootstrap Script \(see page 363\)](#)
 - [Manually Bootstrapping \(see page 364\)](#)
 - [Configuring MidoNet VTEP and Port Bindings \(see page 364\)](#)
 - [Using the MidoNet Manager GUI \(see page 365\)](#)
 - [Using the MidoNet CLI \(see page 368\)](#)
- [Troubleshooting MidoNet and Cumulus VTEPs \(see page 370\)](#)
 - [Troubleshooting the Control Plane \(see page 370\)](#)
 - [Verifying VTEP and OVSDB Services \(see page 370\)](#)
 - [Verifying OVSDB-server Connections \(see page 371\)](#)
 - [Verifying the VXLAN Bridge and VTEP Interfaces \(see page 371\)](#)
 - [Datapath Troubleshooting \(see page 372\)](#)
 - [Verifying IP Reachability \(see page 372\)](#)
 - [MidoNet VXLAN Encapsulation \(see page 372\)](#)
 - [Inspecting the OVSDB \(see page 373\)](#)
 - [Using VTEP-CTL \(see page 373\)](#)
 - [Getting Open Vswitch Database \(OVSDB\) Data \(see page 374\)](#)

Getting Started

Before you create VXLANs with MidoNet, make sure you have the following components:

- A switch (L2 gateway) with a Tomahawk, Trident II+ or Trident II chipset running Cumulus Linux 2.0 and later
- OVSDB server (`ovsdb-server`), included in Cumulus Linux 2.0 and later
- VTEPd (`ovs-vtep`), included in Cumulus Linux 2.0 and later

Integrating a VXLAN with MidoNet involves:

- Preparing for the MidoNet integration
- Bootstrapping the OVS and VTEP
- Configuring the MidoNet VTEP binding
- Verifying the VXLAN configuration

Caveats and Errata

- There is no support for VXLAN routing in the Tomahawk, Trident II+ and Trident II chipsets; use a loopback interface or external router.
- For more information about MidoNet, see the MidoNet Operations Guide, version 1.8 or later.



Preparing for the MidoNet Integration

Before you start configuring the MidoNet tunnel zones, VTEP binding and connecting virtual ports to the VXLAN, you need to complete the bootstrap process on each switch to which you plan to build VXLAN tunnels. This creates the VTEP gateway and initializes the OVS database server. You only need to do the bootstrapping once, before you begin the MidoNet integration.

Enabling the `openvswitch-vtep` Package

Before you start bootstrapping the integration, you need to enable the `openvswitch-vtep` package, since it is disabled by default in Cumulus Linux.

1. Edit the `/etc/default/openvswitch-vtep` file, changing the `START` option from `no` to `yes`. This simple `sed` command does this, and creates a backup as well:

```
cumulus@switch:~$ sudo sed -i.bak s/START=no/START=yes/g /etc/default/openvswitch-vtep
```

2. Start the daemon:

```
cumulus@switch:~$ sudo systemctl start openvswitch-vtep.service
```

Bootstrapping the OVSDB Server and VTEP

Automating with the Bootstrap Script

The `vtep-bootstrap` script is available so you can do the bootstrapping automatically. For information, read `man vtep-bootstrap`. This script requires three parameters, in this order:

- Switch name: The name of the switch that is the VTEP gateway.
- Tunnel IP address: The datapath IP address of the VTEP.
- Management IP address: The IP address of the switch's management interface.

For example, click here ...

```
cumulus@switch:~$ sudo vtep-bootstrap sw11 10.111.1.1 10.50.20.21 --  
no_encryption  
Executed:  
define physical switch  
().  
Executed:  
define local tunnel IP address on the switch  
().  
Executed:  
define management IP address on the switch  
().  
Executed:
```



```
restart a service
(Killing ovs-vtep (28170).
Killing ovsdb-server (28146).
Starting ovsdb-server.
Starting ovs-vtep.) .
```



Since MidoNet does not have a controller, you need to use a dummy IP address (for example, 1.1.1.1) for the controller parameter in the bootstrap script. After the script completes, delete the VTEP manager, since it is not needed and will otherwise fill the logs with inconsequential error messages:

```
cumulus@switch:~$ sudo vtep-ctl del-manager
```

Manually Bootstrapping

If you don't use the bootstrap script, then you must initialize the OVS database instance manually, and create the VTEP.

Perform the following commands in order (see the automated bootstrapping example above for values):

1. Define the switch in OVSDB:

```
cumulus@switch:~$ sudo vtep-ctl add-ps <switch_name>
```

2. Define the VTEP tunnel IP address:

```
cumulus@switch:~$ sudo vtep-ctl set Physical_switch
<switch_name> tunnel_ips=<tunnel_ip>
```

3. Define the management interface IP address:

```
cumulus@switch:~$ sudo vtep-ctl set Physical_switch
<switch_name> management_ips=<management_ip>
```

4. Restart the OVSDB server and vtep daemon:

```
cumulus@switch:~$ sudo systemctl restart openvswitch-vtep.service
```

At this point, the switch is ready to connect to MidoNet. The rest of the configuration is performed in the MidoNet Manager GUI, or using the MidoNet API.

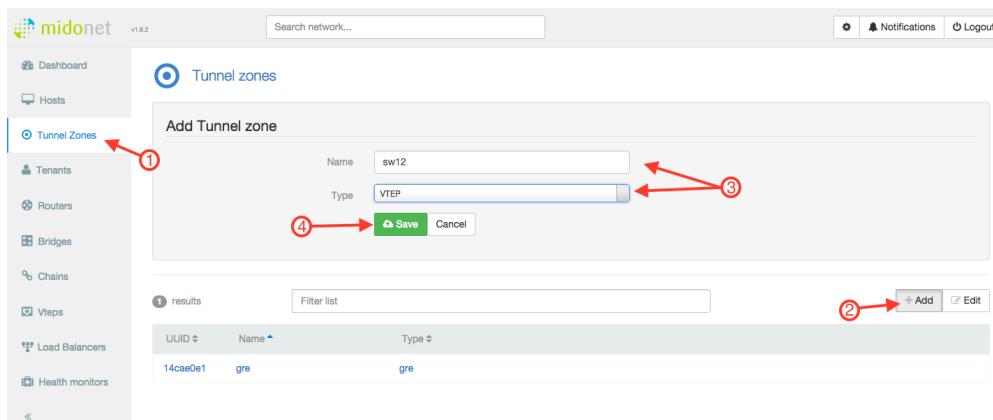
Configuring MidoNet VTEP and Port Bindings

This part of the configuration sets up MidoNet and OpenStack to connect the virtualization environment to the Cumulus Linux switch. The `midotnet-agent` is the networking component that manages the VXLAN, while the Open Virtual Switch (OVS) client on the OpenStack controller node communicates MAC address information between the `midotnet-agent` and the Cumulus Linux OVS database (OVSDB) server.

Using the MidoNet Manager GUI

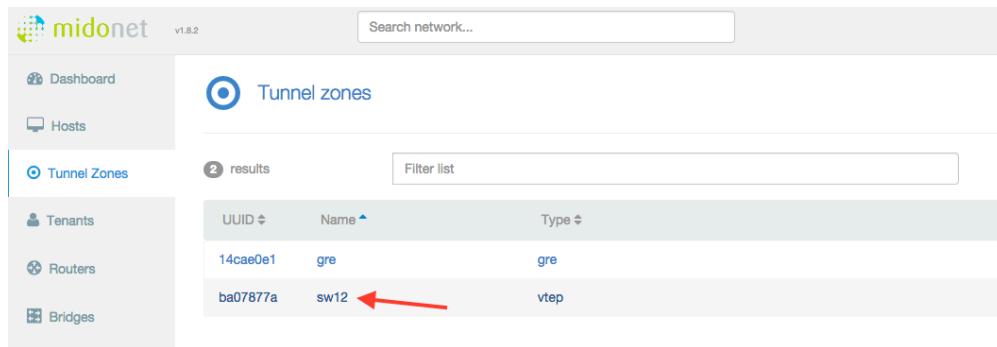
Creating a Tunnel Zone

1. Click **Tunnel Zones** in the menu on the left side.
2. Click **Add**.
3. Give the tunnel zone a **Name** and select "**VTEP**" for the **Type**.
4. Click **Save**.



Adding Hosts to a Tunnel Zone

Once the tunnel zone is created, click the name of the tunnel zone to view the hosts table.



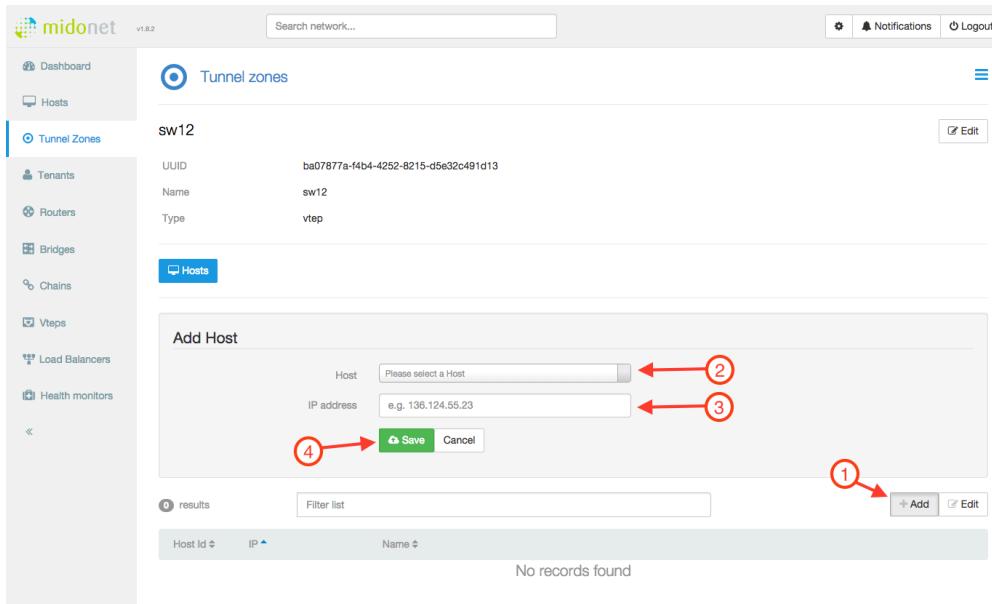
UUID	Name	Type
14cae0e1	gre	gre
ba07877a	sw12	vtep

The tunnel zone is a construct used to define the VXLAN source address used for the tunnel. This host's address is used for the source of the VXLAN encapsulation, and traffic will transit into the routing domain from this point. Thus, the host must have layer 3 reachability to the Cumulus Linux switch tunnel IP.

Next, add a host entry to the tunnel zone:

1. Click **Add**.
2. Select a host from the **Host** list.

3. Provide the tunnel source **IP Address** to use on the selected host.
4. Click **Save**.



The host list now displays the new entry:



Creating the VTEP

1. Click the **Vteps** menu on the left side.
2. Click **Add**.
3. Fill out the fields using the same information you used earlier on the switch for the bootstrap procedure:
 - **Management IP** is typically the eth0 address of the switch. This tells the OVS-client to connect to the OVSDB-server on the Cumulus Linux switch.
 - **Management Port Number** is the PTCP port you configured in the `ovs-ctl-vtep` script earlier (the example uses 6632).
 - **Tunnel Zone** is the name of the zone you created in the previous procedure.
4. Click **Save**.



The screenshot shows the 'Add VTEP' form in the MidoNet web interface. The form fields are:

- Management IP: e.g. 119.15.112.177
- Management Port Number: e.g. 6262
- Tunnel zone: Please select a Tunnel zone

Red numbered arrows indicate specific actions:

- ① Points to the 'Vteps' link in the sidebar.
- ② Points to the '+ Add' button.
- ③ Points to the 'Tunnel zone' dropdown menu.
- ④ Points to the 'Save' button.

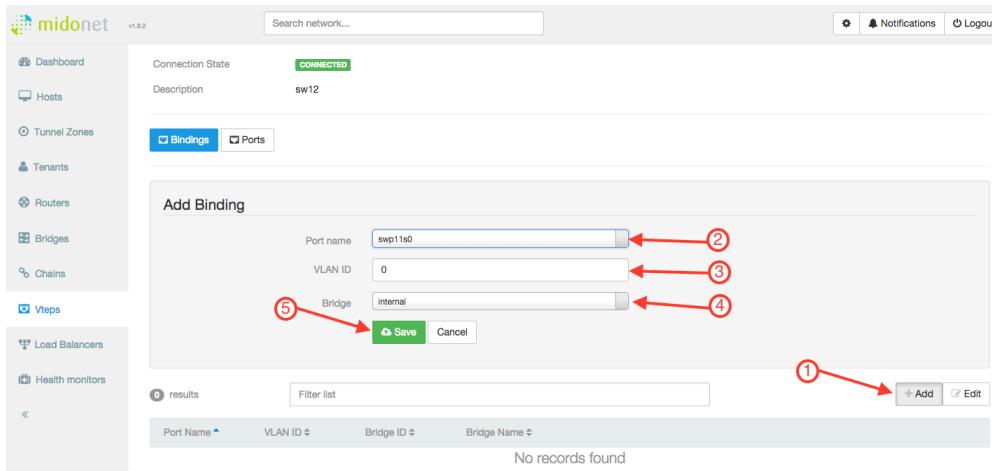
The new VTEP appears in the list below. MidoNet then initiates a connection between the OpenStack Controller and the Cumulus Linux switch. If the OVS client is successfully connected to the OVSDB server, the VTEP entry should display the switch name and VXLAN tunnel IP address, which you specified during the bootstrapping process.

State	Name	Management IP	Management Port	VXLAN Tunnel IP
CONNECTED	sw11	10.50.20.21	6632	10.111.1.1
CONNECTED	sw12	10.50.20.22	6632	10.111.1.2

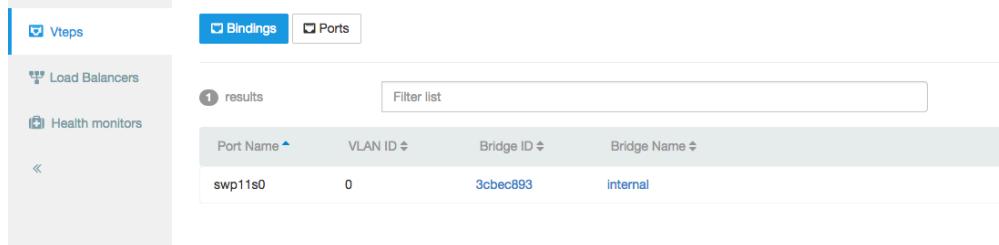
Binding Ports to the VTEP

Now that connectivity is established to the switch, you need to add a physical port binding to the VTEP on the Cumulus Linux switch:

1. Click **Add**.
2. In the **Port Name** list, select the port on the Cumulus Linux switch that you are using to connect to the VXLAN segment.
3. Specify the **VLAN ID** (enter 0 for untagged).
4. In the **Bridge** list, select the MidoNet bridge that the instances (VMs) are using in OpenStack.
5. Click **Save**.



You should see the port binding displayed in the binding table under the VTEP.



Once the port is bound, this automatically configures a VXLAN bridge interface, and includes the VTEP interface and the port bound to the bridge. Now the OpenStack instances (VMs) should be able to ping the hosts connected to the bound port on the Cumulus switch. The Troubleshooting section below demonstrates the verification of the VXLAN data and control planes.

Using the MidoNet CLI

To get started with the MidoNet CLI, you can access the CLI prompt on the OpenStack Controller:

```
root@os-controller:~# midonet-cli
midonet>
```

Now from the MidoNet CLI, the commands explained in this section perform the same operations depicted in the previous section with the MidoNet Manager GUI.

1. Create a tunnel zone with a name and type vtep:

```
midonet> tunnel-zone create name sw12 type vtep
tzone1
```

2. The tunnel zone is a construct used to define the VXLAN source address used for the tunnel. This host's address is used for the source of the VXLAN encapsulation, and traffic will transit into the routing domain from this point. Thus, the host must have layer 3 reachability to the Cumulus Linux switch tunnel IP.

- First, get the list of available hosts connected to the Neutron network and the MidoNet bridge.
- Next, get a listing of all the interfaces.
- Finally, add a host entry to the tunnel zone ID returned in the previous step, and specify which interface address to use.

```

midonet> list host
host host0 name os-compute1 alive true
host host1 name os-network alive true
midonet> host host0 list interface
iface midonet host_id host0 status 0 addresses [] mac 02:4b:38:92:dd:ce mtu 1500 type Virtual endpoint DATAPATH
iface lo host_id host0 status 3 addresses [u'127.0.0.1', u'169.254.169.254', u'0:0:0:0:0:0:0:1'] mac 00:00:00:00:00:00
mtu 65536 type Virtual endpoint LOCALHOST
iface virbr0 host_id host0 status 1 addresses [u'192.168.122.1'] mac 22:6e:63:90:1f:69 mtu 1500 type Virtual endpoint UNKNOWN
iface tap7cf84c-26 host_id host0 status 3 addresses [u'fe80:0:0:0:e822:94ff:fee2:d41b'] mac ea:22:94:e2:d4:1b mtu 6500
0 type Virtual endpoint DATAPATH
iface eth1 host_id host0 status 3 addresses [u'10.111.0.182', u'fe80:0:0:0:5054:ff:fe85:acd6'] mac 52:54:00:85:ac:d6
mtu 1500 type Physical endpoint PHYSICAL
iface tapfd4abcea-df host_id host0 status 3 addresses [u'fe80:0:0:0:14b3:45ff:fe94:5b07'] mac 16:b3:45:94:5b:07 mtu 6500
0 type Virtual endpoint DATAPATH
iface eth0 host_id host0 status 3 addresses [u'10.50.21.182', u'fe80:0:0:0:5054:ff:feef:c5dc'] mac 52:54:00:ef:c5:dc
mtu 1500 type Physical endpoint PHYSICAL
midonet> tunnel-zone tzone0 add member host host0 address 10.111.0.182
zone tzone0 host host0 address 10.111.0.182

```

Repeat this procedure for each OpenStack host connected to the Neutron network and the MidoNet bridge.

3. Create a VTEP and assign it to the tunnel zone ID returned in the previous step. The management IP address (the destination address for the VXLAN/remote VTEP) and the port must be the same ones you configured in the `vtep-bootstrap` script or the manual bootstrapping:

```

midonet> vtep add management-ip 10.50.20.22 management-port 6632
tunnel-zone tzone0
name sw12 description sw12 management-ip 10.50.20.22 management-
port 6632 tunnel-zone tzone0 connection-state CONNECTED

```



In this step, MidoNet initiates a connection between the OpenStack Controller and the Cumulus Linux switch. If the OVS client is successfully connected to the OVSDB server, the returned values should show the name and description matching the `switch-name` parameter specified in the bootstrap process.



Verify the connection-state as CONNECTED, otherwise if ERROR is returned, you must debug. Typically this only fails if the `management-ip` and/or `management-port` settings are wrong.

4. The VTEP binding uses the information provided to MidoNet from the OVSDB server, providing a list of ports that the hardware VTEP can use for layer 2 attachment. This binding virtually connects the physical interface to the overlay switch, and joins it to the Neutron bridged network.

First, get the UUID of the Neutron network behind the MidoNet bridge:

```
midonet> list bridge
bridge bridge0 name internal state up
bridge bridge1 name internal2 state up
midonet> show bridge bridge1 id
6c9826da-6655-4fe3-a826-4dcba6477d2d
```

Next, create the VTEP binding, using the UUID and the switch port being bound to the VTEP on the remote end. If there is no VLAN ID, set `vlan` to 0:

```
midonet> vtep name sw12 binding add network-id 6c9826da-6655-4
fe3-a826-4dcba6477d2d physical-port swp11s0 vlan 0
management-ip 10.50.20.22 physical-port swp11s0 vlan 0 network-
id 6c9826da-6655-4fe3-a826-4dcba6477d2d
```

At this point, the VTEP should be connected, and the layer 2 overlay should be operational. From the openstack instance (VM), you should be able to ping a physical server connected to the port bound to the hardware switch VTEP.

Troubleshooting MidoNet and Cumulus VTEPs

As with any complex system, there is a control plane and data plane.

Troubleshooting the Control Plane

In this solution, the control plane consists of the connection between the OpenStack Controller, and each Cumulus Linux switch running the `ovsdb-server` and `vtep` daemons.

Verifying VTEP and OVSDB Services

First, it is important that the OVSDB server and `ovs-vtep` daemon are running. Verify this is the case:

```
cumulus@switch12:~$ systemctl status openvswitch-vtep.service
ovsdb-server is running with pid 17440
ovs-vtep is running with pid 17444
```

Verifying OVSDB-server Connections

From the OpenStack Controller host, verify that it can connect to the ovsdb-server. Telnet to the switch IP address on port 6632:

```
root@os-controller:~# telnet 10.50.20.22 6632
Trying 10.50.20.22...
Connected to 10.50.20.22.
Escape character is '^]'.
<Ctrl+c>
Connection closed by foreign host.
```

If the connection fails, verify IP reachability from the host to the switch. If that succeeds, it is likely the bootstrap process did not set up port 6632. Redo the bootstrapping procedures above.

```
root@os-controller:~# ping -c1 10.50.20.22
PING 10.50.20.22 (10.50.20.22) 56(84) bytes of data.
64 bytes from 10.50.20.22: icmp_seq=1 ttl=63 time=0.315 ms
--- 10.50.20.22 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.315/0.315/0.315/0.000 ms
```

Verifying the VXLAN Bridge and VTEP Interfaces

After creating the VTEP in MidoNet and adding an interface binding, you should see **br-vxln** and **vxln** interfaces on the switch. You can verify that the VXLAN bridge and VTEP interface are created and UP:

```
cumulus@switch12:~$ sudo brctl show
bridge name      bridge id          STP      enabled interfaces
br-vxln10006  8000.00e0ec2749a2    no       swp11s0
                                         vxln10006
cumulus@switch12:~$ sudo ip -d link show vxln10006
55: vxln10006: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue
        master br-vxln10006 state UNKNOWN mode DEFAULT
        link/ether 72:94:eb:b6:6c:c3 brd ff:ff:ff:ff:ff:ff
        vxlan id 10006 local 10.111.1.2 port 32768 61000 nolearning ageing 3
        00 svcnode 10.111.0.182
                                         bridge_slave
```

Next, look at the bridging table for the VTEP and the forwarding entries. The bound interface and the VTEP should be listed along with the MAC addresses of those interfaces. When the hosts attached to the bound port send data, those MACs are learned, and entered into the bridging table, as well as the OVSDB.

```
cumulus@switch12:~$ brctl showmacs br-vxln10006
port name      mac addr          vlan  is local?    ageing timer
swp11s0        00:e0:ec:27:49:a2   0     yes           0.00
swp11s0        64:ae:0c:32:f1:41   0     no            0.01
vxln10006      72:94:eb:b6:6c:c3   0     yes           0.00

cumulus@switch12:~$ sudo bridge fdb show br-vxln10006
fa:16:3e:14:04:2e dev vxln10004 dst 10.111.0.182 vlan 65535 self
permanent
00:e0:ec:27:49:a2 dev swp11s0 vlan 0 master br-vxln10004 permanent
b6:71:33:3b:a7:83 dev vxln10004 vlan 0 master br-vxln10004 permanent
64:ae:0c:32:f1:41 dev swp11s0 vlan 0 master br-vxln10004
```

Datapath Troubleshooting

If you have verified the control plane is correct, and you still cannot get data between the OpenStack instances and the physical nodes on the switch, there may be something wrong with the data plane. The data plane consists of the actual VXLAN encapsulated path, between one of the OpenStack nodes running the `midolman` service. This is typically the compute nodes, but can include the MidoNet gateway nodes. If the OpenStack instances can ping the tenant router address but cannot ping the physical device connected to the switch (or vice versa), then something is wrong in the data plane.

Verifying IP Reachability

First, there must be IP reachability between the encapsulating node, and the address you bootstrapped as the tunnel IP on the switch. Verify the OpenStack host can ping the tunnel IP. If this doesn't work, check the routing design, and fix the layer 3 problem first.

```
root@os-compute1:~# ping -c1 10.111.1.2
PING 10.111.1.2 (10.111.1.2) 56(84) bytes of data.
64 bytes from 10.111.1.2: icmp_seq=1 ttl=62 time=0.649 ms
--- 10.111.1.2 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.649/0.649/0.649/0.000 ms
```

MidoNet VXLAN Encapsulation

If the instance (VM) cannot ping the physical server, or the reply is not returning, look at the packets on the OpenStack node. Initiate a ping from the OpenStack instance, then using `tcpdump`, hopefully you can see the VXLAN data. This example displays what it looks like when it is working.

```
root@os-compute1:~# tcpdump -i eth1 -l -nnn -vvv -X -e port 4789
52:54:00:85:ac:d6 > 00:e0:ec:26:50:36, ethertype IPv4 (0x0800),
length 148: (tos 0x0, ttl 255, id 7583, offset 0, flags [none], proto
UDP (17), length 134)
```



```
10.111.0.182.41568 > 10.111.1.2.4789: [no cksum] VXLAN, flags [I] (0x08), vni 10008
fa:16:3e:14:04:2e > 64:ae:0c:32:f1:41, ethertype IPv4 (0x0800),
length 98: (tos 0x0, ttl 64, id 64058, offset 0, flags [DF], proto
ICMP (1), length 84)
 10.111.102.104 > 10.111.102.2: ICMP echo request, id 15873, seq 0,
length 64
 0x0000: 4500 0086 1d9f 0000 ff11 8732 0a6f 00b6 E.....2.o..
 0x0010: 0a6f 0102 a260 12b5 0072 0000 0800 0000 .o...`....r....
 0x0020: 0027 1800 64ae 0c32 f141 fa16 3e14 042e .'..d..2.A..>...
 0x0030: 0800 4500 0054 fa3a 4000 4001 5f26 0a6f ..E..T.:@. @_&.o
 0x0040: 6668 0a6f 6602 0800 f9de 3e01 0000 4233 fh.of....>...B3
 0x0050: 7dec 0000 0000 0000 0000 0000 0000 0000 }.....
 0x0060: 0000 0000 0000 0000 0000 0000 0000 0000 .....
 0x0070: 0000 0000 0000 0000 0000 0000 0000 0000 .....
 0x0080: 0000 0000 0000 .....  
00:e0:ec:26:50:36 > 52:54:00:85:ac:d6, ethertype IPv4 (0x0800),
length 148: (tos 0x0, ttl 62, id 2689, offset 0, flags [none], proto
UDP (17), length 134)
 10.111.1.2.63385 > 10.111.0.182.4789: [no cksum] VXLAN, flags [I] (0x08), vni 10008
64:ae:0c:32:f1:41 > fa:16:3e:14:04:2e, ethertype IPv4 (0x0800),
length 98: (tos 0x0, ttl 255, id 64058, offset 0, flags [DF], proto
ICMP (1), length 84)
 10.111.102.2 > 10.111.102.104: ICMP echo reply, id 15873, seq 0,
length 64
 0x0000: 4500 0086 0a81 0000 3e11 5b51 0a6f 0102 E.....>.[Q.o..
 0x0010: 0a6f 00b6 f799 12b5 0072 0000 0800 0000 .o.....r.....
 0x0020: 0027 1800 fa16 3e14 042e 64ae 0c32 f141 .'....>..d..2.A
 0x0030: 0800 4500 0054 fa3a 4000 ff01 a025 0a6f ..E..T.:@....%o
 0x0040: 6602 0a6f 6668 0000 01df 3e01 0000 4233 f..ofh....>...B3
 0x0050: 7dec 0000 0000 0000 0000 0000 0000 0000 }.....
 0x0060: 0000 0000 0000 0000 0000 0000 0000 0000 .....
 0x0070: 0000 0000 0000 0000 0000 0000 0000 0000 .....
 0x0080: 0000 0000 0000 .....  
.....
```

Inspecting the OVSDB

Using VTEP-CTL

These commands show you the information installed in the OVSDB. This database is structured using the *physical switch ID*, with one or more *logical switch IDs* associated with it. The bootstrap process creates the physical switch, and MidoNet creates the logical switch after the control session is established.

Listing the Physical Switch

```
cumulus@switch12:~$ vtep-ctl list-ps
sw12
```



Listing the Logical Switch

```
cumulus@switch12:~$ vtep-ctl list-ls  
mn-6c9826da-6655-4fe3-a826-4dcba6477d2d
```

Listing Local or Remote MAC Addresses

These commands show the MAC addresses learned from the connected port bound to the logical switch, or the MAC addresses advertised from MidoNet. The *unknown-dst* entries are installed to satisfy the ethernet flooding of unknown unicast, and important for learning.

```
cumulus@switch12:~$ vtep-ctl list-local-macs mn-6c9826da-6655-4fe3-  
a826-4dcba6477d2d  
ucast-mac-local  
  64:ae:0c:32:f1:41 -> vxlan_over_ipv4/10.111.1.2  
mcast-mac-local  
  unknown-dst -> vxlan_over_ipv4/10.111.1.2  
  
cumulus@switch12:~$ vtep-ctl list-remote-macs mn-6c9826da-6655-4fe3-  
a826-4dcba6477d2d  
ucast-mac-remote  
  fa:16:3e:14:04:2e -> vxlan_over_ipv4/10.111.0.182  
mcast-mac-remote  
  unknown-dst -> vxlan_over_ipv4/10.111.0.182oh
```

Getting Open Vswitch Database (OVSDB) Data

The `ovsdb-client dump` command is large, but shows all of the information and tables that are used in communication between the OVS client and server.

Click to expand the output ...

```
cumulus@switch12:~$ ovsdb-client dump  
Arp_Sources_Local table  
_uuid locator src_mac  
-----  
Arp_Sources_Remote table  
_uuid locator src_mac  
-----  
Global table  
_uuid managers switches  
-----  
-----  
76672d6a-2740-4c8d-9618-9e8dfb4b0bd7 [ ] [6d459554-0c75-4170-bb3d-  
117eb4ce1f4d]
```



```
Logical_Binding_Stats table
_uuid bytes_from_local bytes_to_local packets_from_local
packets_to_local
-----
d2e378b4-61c1-4daf-9aec-a7fd352d3193 5782569 1658250 21687 14589
Logical_Router table
_uuid description name static_routes switch_binding
-----
Logical_Switch table
_uuid description name tunnel_key
-----
44d162dc-0372-4749-a802-5b153c7120ec "" "mn-6c9826da-6655-4fe3-a826-
4dcba6477d2d" 10006
Manager table
_uuid inactivity_probe is_connected max_backoff other_config status
target
-----
Mcast_Macs_Local table
MAC _uuid ipaddr locator_set logical_switch
-----
unknown-dst 25eaf29a-c540-46e3-8806-3892070a2de5 "" 7a4c000a-244e-
4b37-8f25-fd816c1a80dc 44d162dc-0372-4749-a802-5b153c7120ec
Mcast_Macs_Remote table
MAC _uuid ipaddr locator_set logical_switch
-----
unknown-dst b122b897-5746-449e-83ba-fa571a64b374 "" 6c04d477-18d0-
41df-8d52-dc7b17845ebe 44d162dc-0372-4749-a802-5b153c7120ec
Physical_Locator table
_uuid dst_ip encapsulation_type
-----
2fcf8b7e-e084-4bcb-b668-755ae7ac0bfb "10.111.0.182" "vxlan_over_ipv4"
3f78dbb0-9695-42ef-a31f-aaaf525147f1 "10.111.1.2" "vxlan_over_ipv4"
Physical_Locator_Set table
_uuid locators
-----
6c04d477-18d0-41df-8d52-dc7b17845ebe [2fcf8b7e-e084-4bcb-b668-
755ae7ac0bfb]
7a4c000a-244e-4b37-8f25-fd816c1a80dc [3f78dbb0-9695-42ef-a31f-
aaaf525147f1]
Physical_Port table
_uuid description name port_fault_status vlan_bindings vlan_stats
-----
```

```

bf69fcbb-36b3-4dbc-a90d-fc7412e57076 "swp1" "swp1" [] {} {}
bf38137d-3a14-454e-8df0-9c56e4b4e640 "swp10" "swp10" [] {} {}
69585fff-4360-4177-901d-8360ade5391b "swp11s0" "swp11s0" [] {0=44d162d
c-0372-4749-a802-5b153c7120ec} {0=d2e378b4-61c1-4daf-9aec-
a7fd352d3193}
2a2d04fa-7190-41fe-8cee-318fcbafb2ea "swp11s1" "swp11s1" [] {} {}
684f99d5-426c-45c8-b964-211489f45599 "swp11s2" "swp11s2" [] {} {}
47cc66fb-ef8a-4a9b-a497-1844b89f7d32 "swp11s3" "swp11s3" [] {} {}
5be3a052-be0f-4258-94cb-5e8be9afb896 "swp12" "swp12" [] {} {}
631b19bd-3022-4353-bb2d-f498b0c1cb17 "swp13" "swp13" [] {} {}
3001c904-b152-4dc4-9d8e-718f24ffa439 "swp14" "swp14" [] {} {}
a6f8a88a-3877-4f81-b9b4-d75394a09d2c "swp15" "swp15" [] {} {}
7cb681f4-2206-4c70-85b7-23b60963cd21 "swp16" "swp16" [] {} {}
3943fb6a-0b49-4806-a014-2bcd4d469537 "swp17" "swp17" [] {} {}
109a9911-d6c7-4142-b6c9-7c985506abb4 "swp18" "swp18" [] {} {}
93b85c31-be38-4384-8b7a-9696764f9ba9 "swp19" "swp19" [] {} {}
bcfb2920-6676-494c-9dc9-b474123b7e59 "swp2" "swp2" [] {} {}
4223559a-dalc-4c34-b8bf-bff7ced376ad "swp20" "swp20" [] {} {}
6bccda8-d7e5-4b19-b978-4ec7f5b868e0 "swp21" "swp21" [] {} {}
c6876886-8386-4e34-a307-931909fca58f "swp22" "swp22" [] {} {}
c5a88dd6-d931-4b2c-9baa-a0abfb9d41f5 "swp23" "swp23" [] {} {}
124d1e01-a187-4427-819f-21de66e76f13 "swp24" "swp24" [] {} {}
55b49814-b5c5-405e-8e9f-898f3df4f872 "swp25" "swp25" [] {} {}
b2b2cd14-662d-45a5-87c1-277acbccdf7d "swp26" "swp26" [] {} {}
c35f55f5-8ec6-4fed-bef4-49801cd0934c "swp27" "swp27" [] {} {}
a44c5402-6218-4f09-bf1e-518f41a5546e "swp28" "swp28" [] {} {}
a9294152-2b32-4058-8796-23520ff7379 "swp29" "swp29" [] {} {}
e0ee993a-8383-4701-a766-d425654dbb7f "swp3" "swp3" [] {} {}
d9db91a6-1c10-4154-9269-84877faa79b4 "swp30" "swp30" [] {} {}
b26ce4dd-b771-4d7b-8647-41fa97aa40e3 "swp31" "swp31" [] {} {}
652c6cd1-0823-4585-bb78-658e6ca2abfc "swp32" "swp32" [] {} {}
5b15372b-89f0-4e14-a50b-b6c6f937d33d "swp4" "swp4" [] {} {}
e00741f1-ba34-47c5-ae23-9269c5d1a871 "swp5" "swp5" [] {} {}
7096abaf-eebf-4ee3-b0cc-276224bc3e71 "swp6" "swp6" [] {} {}
439afb62-067e-4bbe-a0d9-ee33a23d2a9c "swp7" "swp7" [] {} {}
54f6c9df-01a1-4d96-9dcf-3035a33ffb3e "swp8" "swp8" [] {} {}
c85ed6cd-a7d4-4016-b3e9-34df592072eb "swp9s0" "swp9s0" [] {} {}
cf382ed6-60d3-43f5-8586-81f4f0f2fb28 "swp9s1" "swp9s1" [] {} {}
c32a9ff9-fd11-4399-815f-806322f26ff5 "swp9s2" "swp9s2" [] {} {}
9a7e42c4-228f-4b55-b972-7c3b8352c27d "swp9s3" "swp9s3" [] {} {}

Physical_Switch table
_uuid description management_ips name ports switch_fault_status
tunnel_ips tunnels
-----
```



```
-----  
6d459554-0c75-4170-bb3d-117eb4ce1f4d "sw12" ["10.50.20.22"] "sw12"  
[109a9911-d6c7-4142-b6c9-7c985506abb4, 124d1e01-a187-4427-819f-  
21de66e76f13, 2a2d04fa-7190-41fe-8cee-318fcbafb2ea, 3001c904-b152-  
4dc4-9d8e-718f24ffa439, 3943fb6a-0b49-4806-a014-2bcd4d469537,  
4223559a-dal1c-4c34-b8bf-bff7ced376ad, 439afb62-067e-4bbe-a0d9-  
ee33a23d2a9c, 47cc66fb-ef8a-4a9b-a497-1844b89f7d32, 54f6c9df-01a1-  
4d96-9dcf-3035a33ffb3e, 55b49814-b5c5-405e-8e9f-898f3df4f872,  
5b15372b-89f0-4e14-a50b-b6c6f937d33d, 5be3a052-be0f-4258-94cb-  
5e8be9afb896, 631b19bd-3022-4353-bb2d-f498b0c1cb17, 652c6cd1-0823-4585  
-bb78-658e6ca2abfc, 684f99d5-426c-45c8-b964-211489f45599, 69585fff-436  
0-4177-901d-8360ade5391b, 6bbccda8-d7e5-4b19-b978-4ec7f5b868e0,  
7096abaf-eebf-4ee3-b0cc-276224bc3e71, 7cb681f4-2206-4c70-85b7-  
23b60963cd21, 93b85c31-be38-4384-8b7a-9696764f9ba9, 9a7e42c4-228f-  
4b55-b972-7c3b8352c27d, a44c5402-6218-4f09-bf1e-518f41a5546e,  
a6f8a88a-3877-4f81-b9b4-d75394a09d2c, a9294152-2b32-4058-8796-23520  
ffb7379, b26ce4dd-b771-4d7b-8647-41fa97aa40e3, b2b2cd14-662d-45a5-  
87c1-277acbccdfdf, bcfb2920-6676-494c-9dcba474123b7e59, bf38137d-  
3a14-454e-8df0-9c56e4b4e640, bf69fcbb-36b3-4dbc-a90d-fc7412e57076,  
c32a9ff9-fd11-4399-815f-806322f26ff5, c35f55f5-8ec6-4fed-bef4-  
49801cd0934c, c5a88dd6-d931-4b2c-9baa-a0abfb9d41f5, c6876886-8386-4  
e34-a307-931909fc58f, c85ed6cd-a7d4-4016-b3e9-34df592072eb, cf382ed6-  
60d3-43f5-8586-81f4f0f2fb28, d9db91a6-1c10-4154-9269-84877faa79b4,  
e00741f1-ba34-47c5-ae23-9269c5d1a871, e0ee993a-8383-4701-a766-  
d425654dbb7f] [] ["10.111.1.2"] [062eaf89-9bd5-4132-8b6b-09db254325af]  
Tunnel table  
_uuid bfd_config_local bfd_config_remote bfd_params bfd_status local  
remote  
-----  
-----  
-----  
-----  
062eaf89-9bd5-4132-8b6b-09db254325af {bfd_dst_ip="169.254.1.0",  
bfd_dst_mac="00:23:20:00:00:01"} {} {} {} {} 3f78dbb0-9695-42ef-a31f-  
aaaf525147f1 2fcf8b7e-e084-4bcb-b668-755ae7ac0bfb  
Ucast_Macs_Local table  
MAC _uuid ipaddr locator logical_switch
```

```
--  
--  
--  
"64:ae:0c:32:f1:41" 47a83a7c-bd2d-4c02-9814-8222229c592f "" 3f78dbb0-9  
695-42ef-a31f-aaaf525147f1 44d162dc-0372-4749-a802-5b153c7120ec  
Ucast_Macs_Remote table  
MAC _uuid ipaddr locator logical_switch  
--  
--  
--  
"fa:16:3e:14:04:2e" 65605488-9ee5-4c8e-93e5-7b1cc15cfcc7 "" 2fcf8b7e-  
e084-4bcb-b668-755ae7ac0bfb 44d162dc-0372-4749-a802-5b153c7120ec
```

Lightweight Network Virtualization - LNV Overview

Lightweight Network Virtualization (LNV) is a technique for deploying [VXLANs](#) (see page 346) without a central controller on bare metal switches. This solution requires no external controller or software suite; it runs the VXLAN service and registration daemons on Cumulus Linux itself. The data path between bridge entities is established on top of a layer 3 fabric by means of a simple service node coupled with traditional MAC address learning.

To see an example of a full solution before reading the following background information, [please read this chapter](#) (see page 427).



LNV is a lightweight controller option. Please [contact Cumulus Networks](#) with your scale requirements so we can make sure this is the right fit for you. There are also other controller options that can work on Cumulus Linux.

Contents

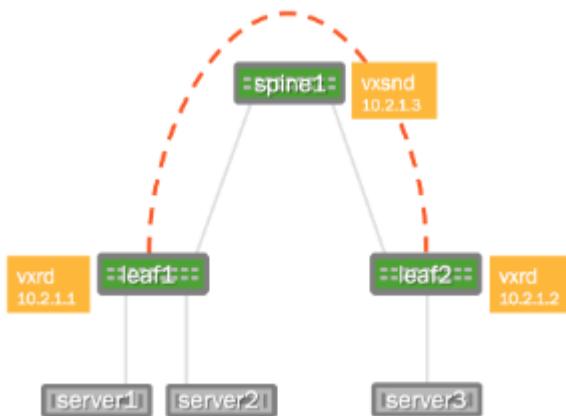
This chapter covers ...

- Understanding LNV Concepts (see page 379)
 - Acquiring the Forwarding Database at the Service Node (see page 379)
 - MAC Learning and Flooding (see page 380)
 - Handling BUM Traffic (see page 380)
- Requirements (see page 381)
 - Hardware Requirements (see page 381)
 - Configuration Requirements (see page 381)
 - Installing the LNV Packages (see page 381)
- Sample LNV Configuration (see page 382)
 - Network Connectivity (see page 382)
 - Layer 3 IP Addressing (see page 382)
 - Layer 3 Fabric (see page 385)
 - Host Configuration (see page 387)

- Configuring the VLAN to VXLAN Mapping (see page 388)
- Verifying the VLAN to VXLAN Mapping (see page 390)
- Enabling and Managing Service Node and Registration Daemons (see page 391)
 - Enabling the Service Node Daemon (see page 391)
 - Enabling the Registration Daemon (see page 391)
 - Checking the Daemon Status (see page 391)
- Configuring the Registration Node (see page 392)
- Configuring the Service Node (see page 394)
- Verification and Troubleshooting (see page 395)
 - Verifying the Registration Node Daemon (see page 395)
 - Verifying the Service Node Daemon (see page 396)
 - Verifying Traffic Flow and Checking Counters (see page 397)
 - Pinging to Test Connectivity (see page 398)
 - Troubleshooting with MAC Addresses (see page 399)
 - Checking the Service Node Configuration (see page 400)
- Advanced LNV Usage (see page 400)
 - Scaling LNV by Load Balancing with Anycast (see page 400)
 - Restarting Network Removes vxsnd Anycast IP Address from Loopback Interface (see page 405)
- Related Information (see page 406)

Understanding LNV Concepts

To best describe this feature, consider the following example deployment:



The two switches running Cumulus Linux, called leaf1 and leaf2, each have a bridge configured. These two bridges contain the physical switch port interfaces connecting to the servers as well as the logical VXLAN interface associated with the bridge. By creating a logical VXLAN interface on both leaf switches, the switches become *VTEPs* (virtual tunnel end points). The IP address associated with this VTEP is most commonly configured as its loopback address — in the image above, the loopback address is 10.2.1.1 for leaf1 and 10.2.1.2 for leaf2.



Acquiring the Forwarding Database at the Service Node

In order to connect these two VXLANs together and forward BUM (Broadcast, Unknown-unicast, Multicast) packets to members of a VXLAN, the service node needs to acquire the addresses of all the VTEPs for every VXLAN it serves. The service node daemon does this through a registration daemon running on each leaf switch that contains a VTEP participating in LNV. The registration process informs the service node of all the VXLANs to which the switch belongs.

MAC Learning and Flooding

With LNV, as with traditional bridging of physical LANs or VLANs, a bridge automatically learns the location of hosts as a side effect of receiving packets on a port.

For example, when server1 sends an L2 packet to server3, leaf2 learns that server1's MAC address is located on that particular VXLAN, and the VXLAN interface learns that the IP address of the VTEP for server1 is 10.2.1.1. So when server3 sends a packet to server1, the bridge on leaf2 forwards the packet out of the port to the VXLAN interface and the VXLAN interface sends it, encapsulated in a UDP packet, to the address 10.2.1.1.

But what if server3 sends a packet to some address that has yet to send it a packet (server2, for example)? In this case, the VXLAN interface sends the packet to the service node, which sends a copy to every other VTEP that belongs to the same VXLAN. This is called *service node replication* and is one of two techniques for handling BUM (Broadcast Unknown-unicast and Multicast) traffic.

Handling BUM Traffic

Cumulus Linux has two ways of handling BUM (Broadcast Unknown-unicast and Multicast) traffic:

- Head end replication
- Service node replication

Head end replication is enabled by default in Cumulus Linux.



You cannot have both service node and head end replication configured simultaneously, as this causes the BUM traffic to be duplicated — both the source VTEP and the service node sending their own copy of each packet to every remote VTEP.

Head End Replication

The Broadcom Tomahawk, Trident II+ and Trident II chipsets and Mellanox Spectrum chipsets are capable of head end replication (HER) — the ability to generate all the BUM traffic in hardware. The most scalable solution available with LNV is to have each VTEP (top of rack switch) generate all of its own BUM traffic rather than relying on an external service node. HER is enabled by default in Cumulus Linux.

Cumulus Linux verified support for up to 128 VTEPs with head end replication.

To disable head end replication, edit `/etc/vxrd.conf` and set `head_rep` to `False`.

Service Node Replication

Cumulus Linux also supports service node replication for VXLAN BUM packets. This is useful with LNV if you have more than 128 VTEPs. However, it is not recommended because it forces the spine switches running the `vxsnd` (service node daemon) to replicate the packets in software instead of in hardware, unlike head end replication. If you're not using a controller but have more than 128 VTEPs, contact [Cumulus Networks](#).



To enable service node replication:

1. Disable head end replication; set `head_rep` to *False* in `/etc/vxrd.conf`.
2. Configure a service node IP address for every VXLAN interface using the `vxlan-svcnodeip` parameter:

```
cumulus@switch:~$ net add vxlan VXLAN vxlan svcnodeip IP_ADDRESS
```



You only specify this parameter when head end replication is disabled. For the loopback, the parameter is still named `vxrd-svcnode-ip`.

3. Edit `/etc/vxsnd.conf`, and configure the following:

- Set the same service node IP address that you did in the previous step:

```
svcnode_ip = <>
```

- To forward VXLAN data traffic, set the following variable to *True*:

```
enable_vxlan_listen = true
```

Requirements

Hardware Requirements

- Switches with a Broadcom Tomahawk, Trident II+ or Trident II, or Mellanox Spectrum chipset running Cumulus Linux 2.5.4 or later. Please refer to the Cumulus Networks [hardware compatibility list](#) for a list of supported switch models.

Configuration Requirements

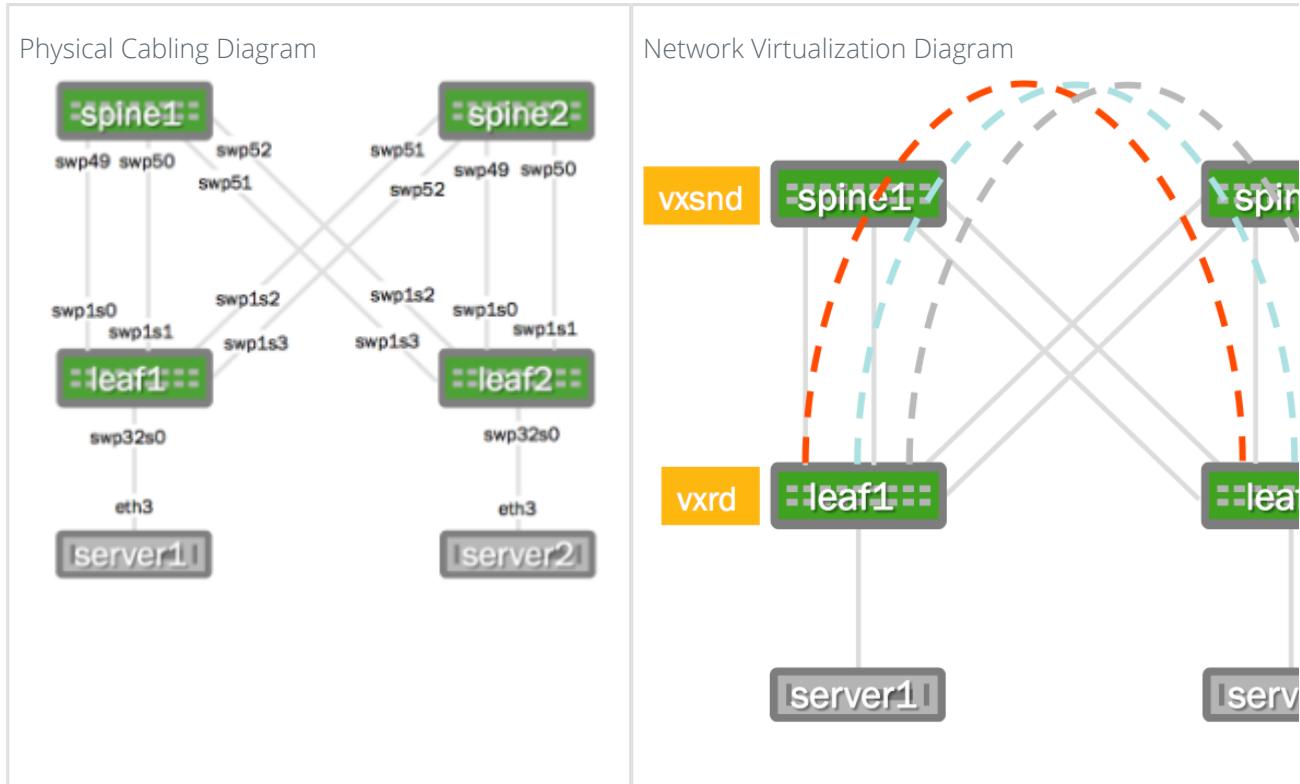
- The VXLAN has an associated **VXLAN Network Identifier (VNI)**, also interchangeably called a VXLAN ID.
- The VNI should not be 0 or 16777215, as these two numbers are reserved values under Cumulus Linux.
- The VXLAN link and physical interfaces are added to the bridge to create the association between the port, VLAN and VXLAN instance.
- Each bridge on the switch has only one VXLAN interface. Cumulus Linux does not support more than one VXLAN link in a bridge; however, a switch can have multiple bridges.
- An SVI (Switch VLAN Interface) or L3 address on the bridge is not supported. For example, you can't ping from the leaf1 SVI to the leaf2 SVI via the VXLAN tunnel; you would need to use server1 and server2 to verify.

Installing the LNV Packages

`vxfl1d` is installed by default on all new installations of Cumulus Linux 3.x. If you are upgrading from an earlier version, run `sudo apt-get install python-vxfl1d` to install the LNV package.

Sample LNV Configuration

The following images illustrate the configuration that is referenced throughout this chapter.



Want to try out configuring LNV and don't have a Cumulus Linux switch? Check out [Cumulus VX](#).

Network Connectivity

There must be full network connectivity before you can configure LNV. The layer 3 IP addressing information as well as the OSPF configuration (`/etc/quagga/Quagga.conf`) below is provided to make the LNV example easier to understand.



OSPF is not a requirement for LNV, LNV just requires L3 connectivity. With Cumulus Linux this can be achieved with static routes, OSPF or BGP.

Layer 3 IP Addressing

Here is the configuration for the IP addressing information used in this example.

**spine1:**

```
cumulus@spine1:~$ net
add interface swp49 ip address
10.1.1.2/30
cumulus@spine1:~$ net
add interface swp50 ip address
10.1.1.6/30
cumulus@spine1:~$ net
add interface swp51 ip address
10.1.1.50/30
cumulus@spine1:~$ net
add interface swp52 ip address
10.1.1.54/30
cumulus@spine1:~$ net
add loopback lo ip address
10.2.1.3/32
cumulus@spine1:~$ net pending
cumulus@spine1:~$ net commit
```

These commands create the following configuration:

```
cumulus@spine1:~$ cat /etc
/network/interfaces
auto lo
iface lo inet loopback
    address 10.2.1.3/32

auto eth0
iface eth0 inet dhcp

auto swp49
iface swp49
    address 10.1.1.2/30

auto swp50
iface swp50
    address 10.1.1.6/30

auto swp51
iface swp51
    address 10.1.1.50/30

auto swp52
iface swp52
    address 10.1.1.54/30
```

spine2:

```
cumulus@spine2:~$ net add
interface swp49 ip address
10.1.1.18/30
cumulus@spine2:~$ net add
interface swp50 ip address
10.1.1.22/30
cumulus@spine2:~$ net add
interface swp51 ip address
10.1.1.34/30
cumulus@spine2:~$ net add
interface swp52 ip address
10.1.1.38/30
cumulus@spine2:~$ net add
loopback lo ip address 10.2.1.4
/32
cumulus@spine2:~$ net pending
cumulus@spine2:~$ net commit
```

These commands create the following configuration:

```
cumulus@spine2:~$ cat /etc
/network/interfaces
auto lo
iface lo inet loopback
    address 10.2.1.4/32

auto eth0
iface eth0 inet dhcp

auto swp49
iface swp49
    address 10.1.1.18/30

auto swp50
iface swp50
    address 10.1.1.22/30

auto swp51
iface swp51
    address 10.1.1.34/30

auto swp52
iface swp52
    address 10.1.1.38/30
```

leaf1:

```
cumulus@leaf1:~$ net add
interface swp1 breakout 4x
cumulus@leaf1:~$ net add
interface swp1s0 ip address
10.1.1.1/30
cumulus@leaf1:~$ net add
interface swp1s1 ip address
10.1.1.5/30
cumulus@leaf1:~$ net add
interface swp1s2 ip address
10.1.1.33/30
cumulus@leaf1:~$ net add
interface swp1s3 ip address
10.1.1.37/30
cumulus@leaf1:~$ net add
loopback lo ip address 10.2.1.1
/32
cumulus@leaf1:~$ net pending
cumulus@leaf1:~$ net commit
```

These commands create the following configuration:

```
cumulus@leaf1:~$ cat /etc
/network/interfaces
auto lo
iface lo inet loopback
    address 10.2.1.1/32

auto eth0
iface eth0 inet dhcp

auto swp1s0
iface swp1s0
    address 10.1.1.1/30

auto swp1s1
iface swp1s1
    address 10.1.1.5/30

auto swp1s2
iface swp1s2
    address 10.1.1.33/30

auto swp1s3
iface swp1s3
```

leaf2:

```
cumulus@leaf2:~$ net add
interface swp1 breakout 4x
cumulus@leaf2:~$ net add
interface swp1s0 ip address
10.1.1.17/30
cumulus@leaf2:~$ net add
interface swp1s1 ip address
10.1.1.21/30
cumulus@leaf2:~$ net add
interface swp1s2 ip address
10.1.1.49/30
cumulus@leaf2:~$ net add
interface swp1s3 ip address
10.1.1.53/30
cumulus@leaf2:~$ net add
loopback lo ip address 10.2.1.2
/32
cumulus@leaf2:~$ net pending
cumulus@leaf2:~$ net commit
```

These commands create the following configuration:

```
cumulus@leaf2:~$ cat /etc
/network/interfaces
auto lo
iface lo inet loopback
    address 10.2.1.2/32

auto eth0
iface eth0 inet dhcp

auto swp1s0
iface swp1s0
    address 10.1.1.17/30

auto swp1s1
iface swp1s1
    address 10.1.1.21/30

auto swp1s2
iface swp1s2
    address 10.1.1.49/30

auto swp1s3
iface swp1s3
```



address 10.1.1.37/30

address 10.1.1.53/30

Layer 3 Fabric

The service nodes and registration nodes must all be routable between each other. The L3 fabric on Cumulus Linux can either be [BGP \(see page 516\)](#) or [OSPF \(see page 500\)](#). In this example, OSPF is used to demonstrate full reachability. Click to expand the Quagga configurations below.

Click to expand the OSPF configuration ...

Quagga configuration using OSPF:

spine1:

```
cumulus@spine1:~$ net add ospf
network 10.2.1.3/32 area
0.0.0.0
cumulus@spine1:~$ net add
interface swp49 ospf network
point-to-point
cumulus@spine1:~$ net add
interface swp50 ospf network
point-to-point
cumulus@spine1:~$ net add
interface swp51 ospf network
point-to-point
cumulus@spine1:~$ net add
interface swp52 ospf network
point-to-point
cumulus@spine1:~$ net add
interface swp49 ospf area
0.0.0.0
cumulus@spine1:~$ net add
interface swp50 ospf area
0.0.0.0
cumulus@spine1:~$ net add
interface swp51 ospf area
0.0.0.0
cumulus@spine1:~$ net add
interface swp52 ospf area
0.0.0.0
cumulus@spine1:~$ net add ospf
router-id 10.2.1.3
cumulus@spine1:~$ net pending
cumulus@spine1:~$ net commit
```

These commands create the following configuration:

spine2:

```
cumulus@spine2:~$ net add ospf
network 10.2.1.4/32 area
0.0.0.0
cumulus@spine2:~$ net add
interface swp49 ospf network
point-to-point
cumulus@spine2:~$ net add
interface swp50 ospf network
point-to-point
cumulus@spine2:~$ net add
interface swp51 ospf network
point-to-point
cumulus@spine2:~$ net add
interface swp52 ospf network
point-to-point
cumulus@spine2:~$ net add
interface swp49 ospf area
0.0.0.0
cumulus@spine2:~$ net add
interface swp50 ospf area
0.0.0.0
cumulus@spine2:~$ net add
interface swp51 ospf area
0.0.0.0
cumulus@spine2:~$ net add
interface swp52 ospf area
0.0.0.0
cumulus@spine2:~$ net add ospf
router-id 10.2.1.4
cumulus@spine2:~$ net pending
cumulus@spine2:~$ net commit
```

These commands create the following configuration:

```

interface swp49
  ip ospf network point-to-point
  ip ospf area 0.0.0.0
!
interface swp50
  ip ospf network point-to-point
  ip ospf area 0.0.0.0
!
interface swp51
  ip ospf network point-to-point
  ip ospf area 0.0.0.0
!
interface swp52
  ip ospf network point-to-point
  ip ospf area 0.0.0.0
!
router ospf
  ospf router-id 10.2.1.3
  network 10.2.1.3/32 area 0.0.0.0
.
```

```

interface swp49
  ip ospf network point-to-point
  ip ospf area 0.0.0.0
!
interface swp50
  ip ospf network point-to-point
  ip ospf area 0.0.0.0
!
interface swp51
  ip ospf network point-to-point
  ip ospf area 0.0.0.0
!
interface swp52
  ip ospf network point-to-point
  ip ospf area 0.0.0.0
!
router ospf
  ospf router-id 10.2.1.4
  network 10.2.1.4/32 area 0.0.0.0
.
```

leaf1:

```

cumulus@leaf1:~$ net add ospf
network 10.2.1.1/32 area
0.0.0.0
cumulus@leaf1:~$ net add
interface swpl1s0 ospf network
point-to-point
cumulus@leaf1:~$ net add
interface swpl1s1 ospf network
point-to-point
cumulus@leaf1:~$ net add
interface swpl1s2 ospf network
point-to-point
cumulus@leaf1:~$ net add
interface swpl1s3 ospf network
point-to-point
cumulus@leaf1:~$ net add
interface swpl1s0 ospf area
0.0.0.0
cumulus@leaf1:~$ net add
interface swpl1s1 ospf area
0.0.0.0
cumulus@leaf1:~$ net add
interface swpl1s2 ospf area
0.0.0.0

```

leaf2:

```

cumulus@leaf2:~$ net add ospf
network 10.2.1.2/32 area
0.0.0.0
cumulus@leaf2:~$ net add
interface swpl1s0 ospf network
point-to-point
cumulus@leaf2:~$ net add
interface swpl1s1 ospf network
point-to-point
cumulus@leaf2:~$ net add
interface swpl1s2 ospf network
point-to-point
cumulus@leaf2:~$ net add
interface swpl1s3 ospf network
point-to-point
cumulus@leaf2:~$ net add
interface swpl1s0 ospf area
0.0.0.0
cumulus@leaf2:~$ net add
interface swpl1s1 ospf area
0.0.0.0
cumulus@leaf2:~$ net add
interface swpl1s2 ospf area
0.0.0.0

```

```
cumulus@leaf1:~$ net add
interface swp1s3 ospf area
0.0.0.0
cumulus@leaf1:~$ net add ospf
router-id 10.2.1.1
cumulus@leaf1:~$ net pending
cumulus@leaf1:~$ net commit
```

These commands create the following configuration:

```
interface swp1s0
  ip ospf network point-to-point
  ip ospf area 0.0.0.0
!
interface swp1s1
  ip ospf network point-to-point
  ip ospf area 0.0.0.0
!
interface swp1s2
  ip ospf network point-to-point
  ip ospf area 0.0.0.0
!
interface swp1s3
  ip ospf network point-to-point
  ip ospf area 0.0.0.0
!
router ospf
  ospf router-id 10.2.1.1
  network 10.2.1.1/32 area
  0.0.0.0
```

```
cumulus@leaf2:~$ net add
interface swp1s3 ospf area
0.0.0.0
cumulus@leaf2:~$ net add ospf
router-id 10.2.1.2
cumulus@leaf2:~$ net pending
cumulus@leaf2:~$ net commit
```

These commands create the following configuration:

```
interface swp1s0
  ip ospf network point-to-point
  ip ospf area 0.0.0.0
!
interface swp1s1
  ip ospf network point-to-point
  ip ospf area 0.0.0.0
!
interface swp1s2
  ip ospf network point-to-point
  ip ospf area 0.0.0.0
!
interface swp1s3
  ip ospf network point-to-point
  ip ospf area 0.0.0.0
!
router ospf
  ospf router-id 10.2.1.2
  network 10.2.1.2/32 area
  0.0.0.0
```

Host Configuration

In this example, the servers are running Ubuntu 14.04. There needs to be a trunk mapped from server1 and server2 to the respective switch. In Ubuntu this is done with subinterfaces. You can expand the configurations below.

Click to expand the host configurations ...

server1:

```
auto eth3.10
iface eth3.10 inet
static
  address 10.10.10.1/24
auto eth3.20
```

server2:

```
auto eth3.10
iface eth3.10 inet
static
  address 10.10.10.2/24
auto eth3.20
```

```
iface eth3.20 inet
static
    address 10.10.20.1/24
auto eth3.30
iface eth3.30 inet
static
    address 10.10.30.1/24
```

```
iface eth3.20 inet
static
    address 10.10.20.2/24
auto eth3.30
iface eth3.30 inet
static
    address 10.10.30.2/24
```

On Ubuntu it is more reliable to use `ifup` and `if down` to bring the interfaces up and down individually, rather than restarting networking entirely (that is, there is no equivalent to `if reload` like there is in Cumulus Linux):

```
cumulus@server1:~$ sudo ifup eth3.10
Set name-type for VLAN subsystem. Should be visible in /proc/net/vlan
/config
Added VLAN with VID == 10 to IF -:eth3:-
cumulus@server1:~$ sudo ifup eth3.20
Set name-type for VLAN subsystem. Should be visible in /proc/net/vlan
/config
Added VLAN with VID == 20 to IF -:eth3:-
cumulus@server1:~$ sudo ifup eth3.30
Set name-type for VLAN subsystem. Should be visible in /proc/net/vlan
/config
Added VLAN with VID == 30 to IF -:eth3:-
```

Configuring the VLAN to VXLAN Mapping

Configure the VLANs and associated VXLANs. In this example, there are 3 VLANs and 3 VXLAN IDs (VNIs). VLANs 10, 20 and 30 are used and associated with VNIs 10, 2000 and 30 respectively. The loopback address, used as the `vxlan-local-tunnelip`, is the only difference between leaf1 and leaf2 for this demonstration.

leaf1:

```
cumulus@leaf1:~$ net add
loopback lo ip address 10.2.1.1
/32
cumulus@leaf1:~$ net add
loopback lo vxrd-src-ip
10.2.1.1
cumulus@leaf1:~$ net add
loopback lo vxrd-svcnode-ip
10.2.1.3
cumulus@leaf1:~$ net add vxlan
vni-10 vxlan id 10
```

leaf2:

```
cumulus@leaf2:~$ net add
loopback lo ip address 10.2.1.2
/32
cumulus@leaf2:~$ net add
loopback lo vxrd-src-ip
10.2.1.2
cumulus@leaf2:~$ net add
loopback lo vxrd-svcnode-ip
10.2.1.3
cumulus@leaf2:~$ net add vxlan
vni-10 vxlan id 10
```



```
cumulus@leaf1:~$ net add vxlan  
vni-10 vxlan local-tunnelip  
10.2.1.1  
cumulus@leaf1:~$ net add vxlan  
vni-10 bridge access 10  
cumulus@leaf1:~$ net add vxlan  
vni-2000 vxlan id 2000  
cumulus@leaf1:~$ net add vxlan  
vni-2000 vxlan local-tunnelip  
10.2.1.1  
cumulus@leaf1:~$ net add vxlan  
vni-2000 bridge access 20  
cumulus@leaf1:~$ net add vxlan  
vni-30 vxlan id 30  
cumulus@leaf1:~$ net add vxlan  
vni-30 vxlan local-tunnelip  
10.2.1.1  
cumulus@leaf1:~$ net add vxlan  
vni-30 bridge access 30  
cumulus@leaf1:~$ net pending  
cumulus@leaf1:~$ net commit
```

These commands create the following configuration in the /etc/network/interfaces file:

```
auto lo  
iface lo  
    address 10.2.1.1/32  
    vxrd-src-ip 10.2.1.1  
  
auto bridge  
iface bridge  
    bridge-ports vni-10 vni-2000  
    vni-30  
    bridge-vids 10 20 30  
    bridge-vlan-aware yes  
  
auto vni-10  
iface vni-10  
    bridge-access 10  
    mstpcl-bpduguard yes  
    mstpcl-portbpdufilter yes  
    vxlan-id 10  
    vxlan-local-tunnelip 10.2.1.1  
  
auto vni-2000  
iface vni-2000  
    bridge-access 20  
    mstpcl-bpduguard yes
```

```
cumulus@leaf2:~$ net add vxlan  
vni-10 vxlan local-tunnelip  
10.2.1.2  
cumulus@leaf2:~$ net add vxlan  
vni-10 bridge access 10  
cumulus@leaf2:~$ net add vxlan  
vni-2000 vxlan id 2000  
cumulus@leaf2:~$ net add vxlan  
vni-2000 vxlan local-tunnelip  
10.2.1.2  
cumulus@leaf2:~$ net add vxlan  
vni-2000 bridge access 20  
cumulus@leaf2:~$ net add vxlan  
vni-30 vxlan id 30  
cumulus@leaf2:~$ net add vxlan  
vni-30 vxlan local-tunnelip  
10.2.1.2  
cumulus@leaf2:~$ net add vxlan  
vni-30 bridge access 30  
cumulus@leaf2:~$ net pending  
cumulus@leaf2:~$ net commit
```

These commands create the following configuration in the /etc/network/interfaces file:

```
auto lo  
iface lo  
    address 10.2.1.2/32  
    vxrd-src-ip 10.2.1.2  
  
auto bridge  
iface bridge  
    bridge-ports vni-10 vni-2000  
    vni-30  
    bridge-vids 10 20 30  
    bridge-vlan-aware yes  
  
auto vni-10  
iface vni-10  
    bridge-access 10  
    mstpcl-bpduguard yes  
    mstpcl-portbpdufilter yes  
    vxlan-id 10  
    vxlan-local-tunnelip 10.2.1.2  
  
auto vni-2000  
iface vni-2000  
    bridge-access 20  
    mstpcl-bpduguard yes
```

```
mstpctl-portbpdufilter yes
vxlan-id 2000
vxlan-local-tunnelip 10.2.1.1

auto vni-30
iface vni-30
    bridge-access 30
    mstpctl-bpduguard yes
    mstpctl-portbpdufilter yes
    vxlan-id 30
    vxlan-local-tunnelip 10.2.1.1
```

```
mstpctl-portbpdufilter yes
vxlan-id 2000
vxlan-local-tunnelip 10.2.1.2

auto vni-30
iface vni-30
    bridge-access 30
    mstpctl-bpduguard yes
    mstpctl-portbpdufilter yes
    vxlan-id 30
    vxlan-local-tunnelip 10.2.1.2
```



Why is vni-2000 not vni-20? For example, why not tie VLAN 20 to VNI 20, or why was 2000 used? VXLANs and VLANs do not need to be the same number. This was done on purpose to highlight this fact. However if you are using fewer than 4096 VLANs, there is no reason not to make it easy and correlate VLANs to VXLANs. It is completely up to you.

Verifying the VLAN to VXLAN Mapping

Use the `brctl show` command to see the physical and logical interfaces associated with that bridge:

```
cumulus@leaf1:~$ brctl show
bridge name      bridge id      STP enabled      interfaces
bridge          8000.443839008404  no              swp32s0.10
                                         vni-10
bridge          8000.443839008404  no              swp32s0.20
                                         vni-2000
bridge          8000.443839008404  no              swp32s0.30
                                         vni-30
```

As with any logical interfaces on Linux, the name does not matter (other than a 15-character limit). To verify the associated VNI for the logical name, use the `ip -d link show` command:

```
cumulus@leaf1:~$ ip -d link show vni-10
43: vni-10: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue
    master br-10 state UNKNOWN mode DEFAULT
        link/ether 02:ec:ec:bd:7f:c6 brd ff:ff:ff:ff:ff:ff
        vxlan id 10 srcport 32768 61000 dstport 4789 ageing 300
        bridge_slave
```

The `vxlan id 10` indicates the VXLAN ID/VNI is indeed 10 as the logical name suggests.



Enabling and Managing Service Node and Registration Daemons

Every VTEP must run the registration daemon (`vxrd`). Typically, every leaf switch acts as a VTEP. A minimum of 1 switch (a switch not already acting as a VTEP) must run the service node daemon (`vxsnd`). The instructions for enabling these daemons follows.

Enabling the Service Node Daemon

The service node daemon (`vxsnd`) is included in the Cumulus Linux repository as `vxflid-vxsnd`. The service node daemon can run on any switch running Cumulus Linux as long as that switch is not also a VXLAN VTEP. In this example, enable the service node only on the spine1 switch, then restart the service.

```
cumulus@spine1:~$ sudo systemctl enable vxsnd.service
cumulus@spine1:~$ sudo systemctl restart vxsnd.service
```



Do not run `vxsnd` on a switch that is already acting as a VTEP.

Enabling the Registration Daemon

The registration daemon (`vxrd`) is included in the Cumulus Linux package as `vxflid-vxrd`. The registration daemon must run on each VTEP participating in LNV, so you must enable it on every TOR (leaf) switch acting as a VTEP, then restart the `vxrd` daemon. For example, on leaf1:

```
cumulus@leaf1:~$ sudo systemctl enable vxrd.service
cumulus@leaf1:~$ sudo systemctl restart vxrd.service
```

Then enable and restart the `vxrd` daemon on leaf2:

```
cumulus@leaf2:~$ sudo systemctl enable vxrd.service
cumulus@leaf2:~$ sudo systemctl restart vxrd.service
```

Checking the Daemon Status

To determine if the daemon is running, use the `systemctl status <daemon name>.service` command.

For the service node daemon:

```
cumulus@spine1:~$ sudo systemctl status vxsnd.service
vxsnd.service - Lightweight Network Virt Discovery Svc and Replicator
   Loaded: loaded (/lib/systemd/system/vxsnd.service; enabled)
```



```
Active: active (running) since Wed 2016-05-11 11:42:55 UTC; 10min
ago
Main PID: 774 (vxsnd)
CGroup: /system.slice/vxsnd.service
        774 /usr/bin/python /usr/bin/vxsnd

May 11 11:42:55 cumulus vxsnd[774]: INFO: Starting (pid 774) ...
```

For the registration daemon:

```
cumulus@leaf1:~$ sudo systemctl status vxrd.service
vxrd.service - Lightweight Network Virtualization Peer Discovery
Daemon
   Loaded: loaded (/lib/systemd/system/vxrd.service; enabled)
   Active: active (running) since Wed 2016-05-11 11:42:55 UTC; 10min
ago
 Main PID: 929 (vxrd)
    CGroup: /system.slice/vxrd.service
            929 /usr/bin/python /usr/bin/vxrd

May 11 11:42:55 cumulus vxrd[929]: INFO: Starting (pid 929) ...
```

Configuring the Registration Node

The registration node was configured earlier in `/etc/network/interfaces` in the [VXLAN mapping \(see page 388\)](#) section above; no additional configuration is typically needed. However, if you need to modify the registration node configuration, edit `/etc/vxrd.conf`.

Configuring the registration node in `/etc/vxrd.conf` ...

```
cumulus@leaf1:~$ sudo nano /etc/vxrd.conf
```

Then edit the `svcnod_ip` variable:

```
svcnod_ip = 10.2.1.3
```

Then perform the same on leaf2:

```
cumulus@leaf2:~$ sudo nano /etc/vxrd.conf
```

And again edit the `svcnod_ip` variable:

```
svcnod_ip = 10.2.1.3
```



Enable, then restart the registration node daemon for the change to take effect:

```
cumulus@leaf1:~$ sudo systemctl enable vxrd.service
cumulus@leaf1:~$ sudo systemctl restart vxrd.service
```

Restart the daemon on leaf2:

```
cumulus@leaf2:~$ sudo systemctl enable vxrd.service
cumulus@leaf2:~$ sudo systemctl restart vxrd.service
```

The complete list of options you can configure is listed below:

Registration node options ...

Name	Description	Default
loglevel	The log level, which can be DEBUG, INFO, WARNING, ERROR, CRITICAL.	INFO
logdest	The destination for log messages. It can be a file name, <code>stdout</code> or <code>syslog</code> .	syslog
logfilesize	Log file size in bytes. Used when <code>logdest</code> is a file name.	512000
logbackupcount	Maximum number of log files stored on the disk. Used when <code>logdest</code> is a file name.	14
pidfile	The PIF file location for the <code>vxrd</code> daemon.	/var/run/Vxrd.pid
udsfile	The file name for the Unix domain socket used for management.	/var/run/Vxrd.sock
vxld_port	The UDP port used for VXLAN control messages.	10001
svcnod_ip	The address to which registration daemons send control messages for registration and/or BUM packets for replication. This can also be configured under <code>/etc/network/interfaces</code> with the <code>vxrd-svcnode-ip</code> keyword.	
holdtime	Hold time (in seconds) for soft state, which is how long the service node waits before ageing out an IP address for a VNI. The <code>vxrd</code> includes this in the register messages it sends to a <code>vxsnd</code> .	90 seconds
src_ip	Local IP address to bind to for receiving control traffic from the service node daemon.	

Name	Description	Default
refresh_rate	Number of times to refresh within the hold time. The higher this number, the more lost UDP refresh messages can be tolerated.	3 seconds
config_check_rate	The number of seconds to poll the system for current VXLAN membership.	5 seconds
head_rep	Enables self replication. Instead of using the service node to replicate BUM packets, it will be done in hardware on the VTEP switch.	true



Use *1, yes, true* or *on* for True for each relevant option. Use *0, no, false* or *off* for False.

Configuring the Service Node

To configure the service node daemon, edit the `/etc/vxsnd.conf` configuration file.



For the example configuration, default values are used, except for the `svcnod_ip` field.

```
cumulus@spine1:~$ sudo nano /etc/vxsnd.conf
```

The address field is set to the loopback address of the switch running the `vxsnd` daemon.

```
svcnod_ip = 10.2.1.3
```

Enable, then restart the service node daemon for the change to take effect:

```
cumulus@spine1:~$ sudo systemctl enable vxsnd.service
cumulus@spine1:~$ sudo systemctl restart vxsnd.service
```

The complete list of options you can configure is listed below:

Name	Description	Default
loglevel	The log level, which can be DEBUG, INFO, WARNING, ERROR, CRITICAL.	INFO
logdest	Destination for log messages. It can be a file name, <code>stdout</code> or <code>syslog</code> .	syslog
logfilesize	The log file size in bytes. Used when <code>logdest</code> is a file name.	512000



Name	Description	Default
logbackupcount	Maximum number of log files stored on disk. Used when <code>logdest</code> is a file name.	14
pidfile	The PID file location for the <code>vxrd</code> daemon.	/var/run/ /vxrd. pid
udsfile	The file name for the Unix domain socket used for management.	/var/run/ /vxrd. sock
vxfld_port	The UDP port used for VXLAN control messages.	10001
svcnod_ip	This is the address to which registration daemons send control messages for registration and/or BUM packets for replication.	0.0.0.0
holdtime	Holddate (in seconds) for soft state. It is used when sending a register message to peers in response to learning a <vnid, addr> from a VXLAN data packet.	90
src_ip	Local IP address to bind to for receiving inter-vxsn control traffic.	0.0.0.0
svcnod_peers	Space-separated list of IP addresses with which the <code>vxsn</code> shares its state.	
enable_vxlan_listen	When set to true, the service node listens for VXLAN data traffic.	true
install_svcnode_ip	When set to true, the <code>sdn_peer_address</code> gets installed on the loopback interface. It gets withdrawn when the <code>vxsn</code> is not in service. If set to true, you must define the <code>sdn_peer_address</code> configuration variable.	false
age_check	Number of seconds to wait before checking the database to age out stale entries.	90 seconds



Use `1, yes, true` or `on` for True for each relevant option. Use `0, no, false` or `off` for False.

Verification and Troubleshooting

Verifying the Registration Node Daemon

Use the `vxrdctl vxlans` command to see the configured VNIs, the local address being used to source the VXLAN tunnel and the service node being used.



```
cumulus@leaf1:~$ vxrdctl vxlans
VNI      Local Addr      Svc
Node
===
=====
10       10.2.1.1
10.2.1.3
30       10.2.1.1
10.2.1.3
2000     10.2.1.1
10.2.1.3
```

```
cumulus@leaf2:~$ vxrdctl vxlans
VNI      Local Addr      Svc
Node
===
=====
10       10.2.1.2
10.2.1.3
30       10.2.1.2
10.2.1.3
2000     10.2.1.2
10.2.1.3
```

Use the `vxrdctl peers` command to see configured VNIs and all VTEPs (leaf switches) within the network that have them configured.

```
cumulus@leaf1:~$ 
vxrdctl peers
VNI      Peer Addrs
===
=====
10       10.2.1.1,
10.2.1.2
30       10.2.1.1,
10.2.1.2
2000     10.2.1.1,
10.2.1.2
```

```
cumulus@leaf2:~$ 
vxrdctl peers
VNI      Peer Addrs
===
=====
10       10.2.1.1,
10.2.1.2
30       10.2.1.1,
10.2.1.2
2000     10.2.1.1,
10.2.1.2
```



When head end replication mode is disabled, the command won't work.

Use the `vxrdctl peers` command to see the other VTEPs (leaf switches) and what VNIs are associated with them. This does not show anything unless you enabled head end replication mode by setting the `head_rep` option to `True`. Otherwise, replication is done by the service node.

```
cumulus@leaf2:~$ vxrdctl peers
Head-end replication is turned off on this device.
This command will not provide any output
```

Verifying the Service Node Daemon

Use the `vxsndctl fdb` command to verify which VNIs belong to which VTEP (leaf switches).

```
cumulus@spine1:~$ vxsndctl fdb
VNI      Address      Ageout
==      =====      =====
 10      10.2.1.1      82
 10      10.2.1.2      77
 30      10.2.1.1      82
 30      10.2.1.2      77
2000     10.2.1.1      82
2000     10.2.1.2      77
```

Verifying Traffic Flow and Checking Counters

VXLAN transit traffic information is stored in a flat file located at /cumulus/switchd/run/stats/vxlan/all.

```
cumulus@leaf1:~$ cat /cumulus/switchd/run/stats/vxlan/all
VNI                      : 10
Network In Octets          : 1090
Network In Packets         : 8
Network Out Octets          : 1798
Network Out Packets         : 13
Total In Octets             : 2818
Total In Packets            : 27
Total Out Octets            : 3144
Total Out Packets           : 39
VN Interface                : vni: 10, swp32s0.10
Total In Octets              : 1728
Total In Packets             : 19
Total Out Octets             : 552
Total Out Packets            : 18
VNI                      : 30
Network In Octets          : 828
Network In Packets         : 6
Network Out Octets          : 1224
Network Out Packets         : 9
Total In Octets              : 2374
Total In Packets             : 23
Total Out Octets             : 2300
Total Out Packets            : 32
VN Interface                : vni: 30, swp32s0.30
Total In Octets              : 1546
Total In Packets             : 17
Total Out Octets             : 552
Total Out Packets            : 17
VNI                      : 2000
Network In Octets          : 676
Network In Packets         : 5
Network Out Octets          : 1072
```

Network Out Packets	:	8
Total In Octets	:	2030
Total In Packets	:	20
Total Out Octets	:	2042
Total Out Packets	:	30
VN Interface	:	vni: 2000, swp32s0.20
Total In Octets	:	1354
Total In Packets	:	15
Total Out Octets	:	446

Pinging to Test Connectivity

To test the connectivity across the VXLAN tunnel with an ICMP echo request (ping), make sure to ping from the server rather than the switch itself.



As mentioned above, SVIs (switch VLAN interfaces) are not supported when using VXLAN. That is, there cannot be an IP address on the bridge that also contains a VXLAN.

Following is the IP address information used in this example configuration.

VNI	server1	server2
10	10.10.10.1	10.10.10.2
2000	10.10.20.1	10.10.20.2
30	10.10.30.1	10.10.30.2

To test connectivity between VNI 10 connected servers by pinging from server1:

```
cumulus@server1:~$ ping 10.10.10.2
PING 10.10.10.2 (10.10.10.2) 56(84) bytes of data.
64 bytes from 10.10.10.2: icmp_seq=1 ttl=64 time=3.90 ms
64 bytes from 10.10.10.2: icmp_seq=2 ttl=64 time=0.202 ms
64 bytes from 10.10.10.2: icmp_seq=3 ttl=64 time=0.195 ms
^C
--- 10.10.10.2 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2002ms
rtt min/avg/max/mdev = 0.195/1.432/3.900/1.745 ms
cumulus@server1:~$
```

The other VNIs were also tested and can be viewed in the expanded output below.

Test connectivity between VNI-2000 connected servers by pinging from server1:

```
cumulus@server1:~$ ping 10.10.20.2
PING 10.10.20.2 (10.10.20.2) 56(84) bytes of data.
```



```
64 bytes from 10.10.20.2: icmp_seq=1 ttl=64 time=1.81 ms
64 bytes from 10.10.20.2: icmp_seq=2 ttl=64 time=0.194 ms
64 bytes from 10.10.20.2: icmp_seq=3 ttl=64 time=0.206 ms
^C
--- 10.10.20.2 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2000ms
rtt min/avg/max/mdev = 0.194/0.739/1.819/0.763 ms
```

Test connectivity between VNI-30 connected servers by pinging from server1:

```
cumulus@server1:~$ ping 10.10.30.2
PING 10.10.30.2 (10.10.30.2) 56(84) bytes of data.
64 bytes from 10.10.30.2: icmp_seq=1 ttl=64 time=1.85 ms
64 bytes from 10.10.30.2: icmp_seq=2 ttl=64 time=0.239 ms
64 bytes from 10.10.30.2: icmp_seq=3 ttl=64 time=0.185 ms
64 bytes from 10.10.30.2: icmp_seq=4 ttl=64 time=0.212 ms
^C
--- 10.10.30.2 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3000ms
rtt min/avg/max/mdev = 0.185/0.622/1.853/0.711 ms
```

Troubleshooting with MAC Addresses

Since there is no SVI, there is no way to ping from the server to the directly attached leaf (top of rack) switch without cabling the switch to itself. The easiest way to see if the server can reach the leaf switch is to check the MAC address table of the leaf switch.

First, get the MAC address of the server:

```
cumulus@server1:~$ ip addr show eth3.10 | grep ether
    link/ether 90:e2:ba:55:f0:85 brd ff:ff:ff:ff:ff:ff
```

Next, check the MAC address table of the leaf switch:

```
cumulus@leaf1:~$ brctl showmacs br-10
port name mac addr      vlan  is local?   ageing timer
vni-10  46:c6:57:fc:1f:54  0     yes        0.00
swp32s0.10 90:e2:ba:55:f0:85  0     no         75.87
vni-10  90:e2:ba:7e:a9:c1  0     no         75.87
swp32s0.10 ec:f4:bb:fc:67:a1  0     yes        0.00
```

90:e2:ba:55:f0:85 appears in the MAC address table, which indicates that connectivity is occurring between leaf1 and server1.



Checking the Service Node Configuration

Use `ip -d link show` to verify the service node, VNI and administrative state of a particular logical VNI interface:

```
cumulus@leaf1:~$ ip -d link show vni-10
35: vni-10: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue
    master br-10 state UNKNOWN mode DEFAULT
        link/ether 46:c6:57:fc:1f:54 brd ff:ff:ff:ff:ff:ff
        vxlan id 10 remote 10.2.1.3 local 10.2.1.1 srcport 32768 61000
        dstport 4789 ageing 300 svcnode 10.2.1.3
            bridge_slave
```

Advanced LNV Usage

Scaling LNV by Load Balancing with Anycast

The above configuration assumes a single service node. A single service node can quickly be overwhelmed by BUM traffic. To load balance BUM traffic across multiple service nodes, use [Anycast](#). Anycast enables BUM traffic to reach the topologically nearest service node rather than overwhelming a single service node.

Enabling the Service Node Daemon on Additional Spine Switches

In this example, spine1 already has the service node daemon enabled. Enable it on the spine2 switch, then restart the `vxsnd` daemon:

```
cumulus@spine2:~$ sudo systemctl enable vxsnd.service
cumulus@spine2:~$ sudo systemctl restart vxsnd.service
```



Configuring the Anycast Address on All Participating Service Nodes

spine1:

Add the 10.10.10.10/32 address to the loopback address:

```
cumulus@spine1:~$ net add  
loopback lo ip address  
10.10.10.10/32  
cumulus@spine1:~$ net pending  
cumulus@spine1:~$ net commit
```

These commands create the following configuration in the /etc/network/interfaces file:

```
auto lo  
iface lo inet loopback  
    address 10.2.1.3/32  
    address 10.10.10.10/32
```

Verify the IP address is configured:

```
cumulus@spine1:~$ ip addr show  
lo  
1: lo: <LOOPBACK,UP,LOWER_UP>  
mtu 16436 qdisc noqueue state  
UNKNOWN  
    link/loopback 00:00:00:00:  
00:00 brd 00:00:00:00:00:00  
        inet 127.0.0.1/8 scope  
host lo  
    inet 10.2.1.3/32 scope  
global lo  
    inet 10.10.10.10/32 scope  
global lo  
    inet6 ::1/128 scope host  
        valid_lft forever  
preferred_lft forever
```

spine2:

Add the 10.10.10.10/32 address to the loopback address:

```
cumulus@spine2:~$ net add  
loopback lo ip address  
10.10.10.10/32  
cumulus@spine2:~$ net pending  
cumulus@spine2:~$ net commit
```

These commands create the following configuration in the /etc/network/interfaces file:

```
auto lo  
iface lo inet loopback  
    address 10.2.1.4/32  
    address 10.10.10.10/32
```

Verify the IP address is configured:

```
cumulus@spine2:~$ ip addr show  
lo  
1: lo: <LOOPBACK,UP,LOWER_UP>  
mtu 16436 qdisc noqueue state  
UNKNOWN  
    link/loopback 00:00:00:00:  
00:00 brd 00:00:00:00:00:00  
        inet 127.0.0.1/8 scope  
host lo  
    inet 10.2.1.4/32 scope  
global lo  
    inet 10.10.10.10/32 scope  
global lo  
    inet6 ::1/128 scope host  
        valid_lft forever  
preferred_lft forever
```

Configuring the Service Node vxsnd.conf File

spine1:

Use a text editor to edit the network configuration:

```
cumulus@spine1:~$ sudo nano  
/etc/vxsnd.conf
```

Change the following values:

```
svcnod_ip = 10.10.10.10  
svcnod_peers = 10.2.1.4  
src_ip = 10.2.1.3
```

This sets the address on which the service node listens to VXLAN messages to the configured Anycast address and sets it to sync with spine2.

Enable, then restart the `vxsnd` daemon:

```
cumulus@spine1:~$ sudo  
systemctl enable vxsnd.service  
cumulus@spine1:~$ sudo  
systemctl restart vxsnd.service
```

spine2:

Use a text editor to edit the network configuration:

```
cumulus@spine2:~$ sudo nano  
/etc/vxsnd.conf
```

Change the following values:

```
svcnod_ip = 10.10.10.10  
svcnod_peers = 10.2.1.3  
src_ip = 10.2.1.4
```

This sets the address on which the service node listens to VXLAN messages to the configured Anycast address and sets it to sync with spine1.

Enable, then restart the `vxsnd` daemon:

```
cumulus@spine1:~$ sudo  
systemctl enable vxsnd.service  
cumulus@spine1:~$ sudo  
systemctl restart vxsnd.service
```



Reconfiguring the VTEPs (Leafs) to Use the Anycast Address

leaf1:

Change the vxrd-svcnode-ip field to the anycast address:

```
cumulus@leaf1:~$ net add  
loopback lo vxrd-svcnode-ip  
10.10.10.10  
cumulus@leaf1:~$ net pending  
cumulus@leaf1:~$ net commit
```

These commands create the following configuration in the /etc/network/interfaces file:

```
auto lo  
iface lo inet loopback  
    address 10.2.1.1  
    vxrd-svcnode-ip 10.10.10.10
```

Verify the new service node is configured:

```
cumulus@leaf1:~$ ip -d link  
show vni-10  
35: vni-10: <BROADCAST,  
MULTICAST,UP,LOWER_UP> mtu  
1500 qdisc noqueue master br-  
10 state UNKNOWN mode DEFAULT  
    link/ether 46:c6:57:fc:1f:  
54 brd ff:ff:ff:ff:ff:ff  
        vxlan id 10 remote  
10.10.10.10 local 10.2.1.1  
srcport 32768 61000 dstport  
4789 ageing 300 svcnode  
10.10.10.10  
        bridge_slave
```

```
cumulus@leaf1:~$ ip -d link  
show vni-2000  
39: vni-2000: <BROADCAST,  
MULTICAST,UP,LOWER_UP> mtu  
1500 qdisc noqueue master br-  
20 state UNKNOWN mode DEFAULT
```

leaf2:

Change the vxrd-svcnode-ip field to the anycast address:

```
cumulus@leaf1:~$ net add  
loopback lo vxrd-svcnode-ip  
10.10.10.10  
cumulus@leaf1:~$ net pending  
cumulus@leaf1:~$ net commit
```

These commands create the following configuration in the /etc/network/interfaces file:

```
auto lo  
iface lo inet loopback  
    address 10.2.1.2  
    vxrd-svcnode-ip 10.10.10.10
```

Verify the new service node is configured:

```
cumulus@leaf2:~$ ip -d link  
show vni-10  
35: vni-10: <BROADCAST,  
MULTICAST,UP,LOWER_UP> mtu  
1500 qdisc noqueue master br-  
10 state UNKNOWN mode DEFAULT  
    link/ether 4e:03:a7:47:a7:  
9d brd ff:ff:ff:ff:ff:ff  
        vxlan id 10 remote  
10.10.10.10 local 10.2.1.2  
srcport 32768 61000 dstport  
4789 ageing 300 svcnode  
10.10.10.10  
        bridge_slave
```

```
cumulus@leaf2:~$ ip -d link  
show vni-2000  
39: vni-2000: <BROADCAST,  
MULTICAST,UP,LOWER_UP> mtu  
1500 qdisc noqueue master br-  
20 state UNKNOWN mode DEFAULT
```

```

link/ether 4a:fd:88:c3:fa:
df brd ff:ff:ff:ff:ff:ff
    vxlan id 2000 remote
10.10.10.10 local 10.2.1.1
srcport 32768 61000 dstport
4789 ageing 300 svcnode
10.10.10.10
    bridge_slave

cumulus@leaf1:~$ ip -d link
show vni-30
37: vni-30: <BROADCAST,
MULTICAST,UP,LOWER_UP> mtu
1500 qdisc noqueue master br-
30 state UNKNOWN mode DEFAULT
    link/ether 3e:b3:dc:f3:bd:
2b brd ff:ff:ff:ff:ff:ff
    vxlan id 30 remote
10.10.10.10 local 10.2.1.1
srcport 32768 61000 dstport
4789 ageing 300 svcnode
10.10.10.10
    bridge_slave

```

⚠ The `svcnode 10.10.10.10` means the interface has the correct service node configured.

Use the `vxrdctl vxlans` command to check the service node:

```

cumulus@leaf1:~$ vxrdctl vxlans
VNI      Local Addr      Svc
Node
=====
=====
10       10.2.1.1
10.2.1.3
30       10.2.1.1
10.2.1.3
2000     10.2.1.1
10.2.1.3

```

```

link/ether 72:3a:bd:06:00:
b7 brd ff:ff:ff:ff:ff:ff
    vxlan id 2000 remote
10.10.10.10 local 10.2.1.2
srcport 32768 61000 dstport
4789 ageing 300 svcnode
10.10.10.10
    bridge_slave

cumulus@leaf2:~$ ip -d link
show vni-30
37: vni-30: <BROADCAST,
MULTICAST,UP,LOWER_UP> mtu
1500 qdisc noqueue master br-
30 state UNKNOWN mode DEFAULT
    link/ether 22:65:3f:63:08:
bd brd ff:ff:ff:ff:ff:ff
    vxlan id 30 remote
10.10.10.10 local 10.2.1.2
srcport 32768 61000 dstport
4789 ageing 300 svcnode
10.10.10.10
    bridge_slave

```

⚠ The `svcnode 10.10.10.10` means the interface has the correct service node configured.

Use the `vxrdctl vxlans` command to check the service node:

```

cumulus@leaf2:~$ vxrdctl vxlans
VNI      Local Addr      Svc
Node
=====
=====
10       10.2.1.2
10.2.1.3
30       10.2.1.2
10.2.1.3
2000     10.2.1.2
10.2.1.3

```

Testing Connectivity

Repeat the ping tests from the previous section. Here is the table again for reference:

VNI	server1	server2
10	10.10.10.1	10.10.10.2
2000	10.10.20.1	10.10.20.2
30	10.10.30.1	10.10.30.2

```

cumulus@server1:~$ ping 10.10.10.2
PING 10.10.10.2 (10.10.10.2) 56(84) bytes of data.
64 bytes from 10.10.10.2: icmp_seq=1 ttl=64 time=5.32 ms
64 bytes from 10.10.10.2: icmp_seq=2 ttl=64 time=0.206 ms
^C
--- 10.10.10.2 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1001ms
rtt min/avg/max/mdev = 0.206/2.767/5.329/2.562 ms

PING 10.10.20.2 (10.10.20.2) 56(84) bytes of data.
64 bytes from 10.10.20.2: icmp_seq=1 ttl=64 time=1.64 ms
64 bytes from 10.10.20.2: icmp_seq=2 ttl=64 time=0.187 ms
^C
--- 10.10.20.2 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1001ms
rtt min/avg/max/mdev = 0.187/0.914/1.642/0.728 ms

cumulus@server1:~$ ping 10.10.30.2
PING 10.10.30.2 (10.10.30.2) 56(84) bytes of data.
64 bytes from 10.10.30.2: icmp_seq=1 ttl=64 time=1.63 ms
64 bytes from 10.10.30.2: icmp_seq=2 ttl=64 time=0.191 ms
^C
--- 10.10.30.2 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1001ms
rtt min/avg/max/mdev = 0.191/0.913/1.635/0.722 ms

```

Restarting Network Removes vxsnd Anycast IP Address from Loopback Interface

If you have not configured a loopback anycast IP address in `/etc/network/interfaces`, but you have enabled the `vxsnd` (service node daemon) log to automatically add anycast IP addresses, when you restart networking (with `systemctl restart networking`), the anycast IP address gets removed from the loopback interface.



To prevent this issue from occurring, you should specify an anycast IP address for the loopback interface in both `/etc/network/interfaces` and `vxsnd.conf`. This way, in case `vxsnd` fails, you can withdraw the IP address.

Related Information

- tools.ietf.org/html/rfc7348
- en.wikipedia.org/wiki/Anycast
- Network virtualization chapter, Cumulus Linux user guide (see page 346)

LNV VXLAN Active-Active Mode

LNV active-active mode allows a pair of `MLAG` switches to act as a single VTEP, providing active-active VXLAN termination for bare metal as well as virtualized workloads.

Contents

This chapter covers ...

- Terminology and Definitions (see page 407)
- Configuring LNV Active-Active Mode (see page 407)
 - active-active VTEP Anycast IP Behavior (see page 408)
 - Failure Scenario Behaviors (see page 408)
 - Checking VXLAN Interface Configuration Consistency (see page 409)
 - Configuring the Anycast IP Address (see page 409)
- Example VXLAN Active-Active Configuration (see page 411)
 - Quagga Configuration (see page 411)
 - Layer 3 IP Addressing (see page 411)
 - Quagga Configuration (see page 417)
 - Host Configuration (see page 420)
 - Enable the Registration Daemon (see page 421)
 - Configuring a VTEP (see page 422)
 - Enable the Service Node Daemon (see page 422)
 - Configuring the Service Node (see page 422)
- Considerations for Virtual Topologies Using Cumulus VX (see page 424)
 - Node ID (see page 424)
 - Bonds with Vagrant (see page 425)
- Troubleshooting with LNV Active-Active (see page 425)
- Caveats and Errata (see page 427)
- Related Information (see page 427)

Terminology and Definitions

Term	Definition
vxrd	VXLAN registration daemon. Runs on the switch that is mapping VLANs to VXLANs. The vxrd daemon needs to be configured to register to a service node. This turns the switch into a VTEP.
VTEP	Virtual tunnel endpoint. This is an encapsulation and decapsulation point for VXLANs.
active-active VTEP	A pair of switches acting as a single VTEP.
ToR	Top of rack switch. Also referred to as a leaf or access switch.
Spine	The aggregation switch for multiple leafs. Specifically used when a data center is using a Clos network architecture . Read more about spine-leaf architecture in this white paper .
vxsnd	VXLAN service node daemon, that can be run to register multiple VTEPs.
exit leaf	A switch dedicated to peering the Clos network to an outside network. Also referred to as border leafs, service leafs or edge leafs.
anycast	When an IP address is advertised from multiple locations. Allows multiple devices to share the same IP and effectively load balance traffic across them. With LNV, anycast is used in 2 places: <ol style="list-style-type: none"> 1. To share a VTEP IP address between a pair of MLAG switches. 2. To load balance traffic for service nodes (e.g. service nodes share an IP address).
ASIC	Application-specific integrated circuit. Also referred to as hardware, or hardware accelerated. Encapsulation and decapsulation are required for the best performance VXLAN-supported ASIC.
RIOT	Broadcom feature for Routing in and out of tunnels. Allows a VXLAN bridge to have a switch VLAN interface associated with it, and traffic to exit a VXLAN into the layer 3 fabric. Also called VXLAN Routing.
VXLAN Routing	Industry standard term for ability to route in and out of a VXLAN. Equivalent to Broadcom RIOT feature.

Configuring LNV Active-Active Mode

LNV requires the following underlying technologies to work correctly.

Technology	More Information
MLAG	Refer to the MLAG chapter (see page) for more detailed configuration information. Configurations for the demonstration are provided below.
OSPF or BGP	Refer to the OSPF chapter or the BGP Chapter for more detailed configuration information. Configurations for the demonstration are provided below.
LNV	Refer to the LNV chapter (see page 406) for more detailed configuration information. Configurations for the demonstration are provided below.
STP	BPDU filter and BPDU guard (see page) should be enabled in the VXLAN interfaces if STP (see page 406) is enabled in the bridge that is connected to the VXLAN. Configurations for the demonstration are provided below.

active-active VTEP Anycast IP Behavior

Each individual switch within an MLAG pair should be provisioned with a virtual IP address in the form of an anycast IP address for VXLAN data-path termination. The VXLAN termination address is an anycast IP address that you configure as a `c1agd` parameter (`c1agd-vxlan-anycast-ip`) under the loopback interface. `c1agd` dynamically adds and removes this address as the loopback interface address as follows:

1	When the switches boot up, <code>ifupdown2</code> places all VXLAN interfaces in a PROTO_DOWN state (see page) . The configured anycast addresses are not configured yet.
2	MLAG peering takes place, and a successful VXLAN interface consistency check between the switches occurs.
3	<code>c1agd</code> (the daemon responsible for MLAG) adds the anycast address to the loopback interface. It then changes the local IP address of the VXLAN interface from a unique address to the anycast virtual IP address and puts the interface in an UP state.

Failure Scenario Behaviors

Scenario	Behavior
The peer link goes down.	The primary MLAG switch continues to keep all VXLAN interfaces up with the anycast IP address while the secondary switch brings down all VXLAN interfaces and places them in a PROTO_DOWN state . The secondary MLAG switch removes the anycast IP address from the loopback interface and changes the local IP address of the VXLAN interface to the configured unique IP address.
One of the switches goes down.	The other operational switch continues to use the anycast IP address.



Scenario	Behavior
c1agd is stopped.	All VXLAN interfaces are put in a PROTO_DOWN state. The anycast IP address is removed from the loopback interface and the local IP addresses of the VXLAN interfaces are changed from the anycast IP address to unique non-virtual IP addresses.
MLAG peering could not be established between the switches.	c1agd brings up all the VXLAN interfaces after the reload timer expires with the configured unique IP address. This allows the VXLAN interface to be up and running on both switches even though peering is not established.
When the peer link goes down but the peer switch is up (i.e. the backup link is active).	All VXLAN interfaces are put into a PROTO_DOWN state on the secondary switch.
A configuration mismatch between the MLAG switches	The VXLAN interface is placed into a PROTO_DOWN state on the secondary switch.

Checking VXLAN Interface Configuration Consistency

The LNV active-active configuration for a given VXLAN interface has to be consistent between the MLAG switches for correct traffic behavior. MLAG ensures that the configuration consistency is met before bringing up the VXLAN interfaces.

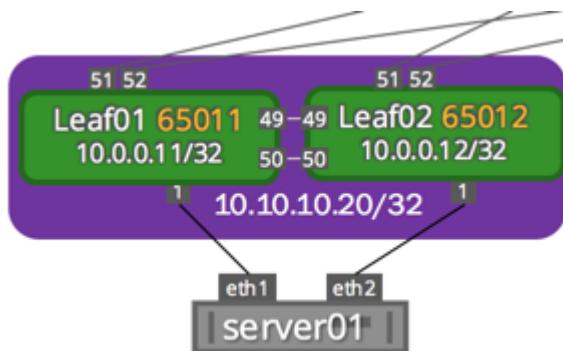
The consistency checks include:

- The anycast virtual IP address for VXLAN termination must be the same on each pair of switches.
- A VXLAN interface with the same VXLAN ID must be configured and administratively up on both switches.

You can use the `c1agctl` command to check if any VXLAN switches are in a PROTO_DOWN state.

Configuring the Anycast IP Address

With MLAG peering, both switches use an anycast IP address for VXLAN encapsulation and decapsulation. This allows remote VTEPs to learn the host MAC addresses attached to the MLAG switches against one logical VTEP, even though the switches independently encapsulate and decapsulate layer 2 traffic originating from the host. The anycast address under the loopback interface can be configured as shown below.



leaf01: /etc/network/interfaces snippet

```
auto lo
iface lo inet loopback
  address 10.0.0.11/32
  vxrd-src-ip 10.0.0.11
  vxrd-svcnode-ip 10.10.10.10
  clagd-vxlan-anycast-ip 10.10.10.20
```

leaf02: /etc/network/interfaces snippet

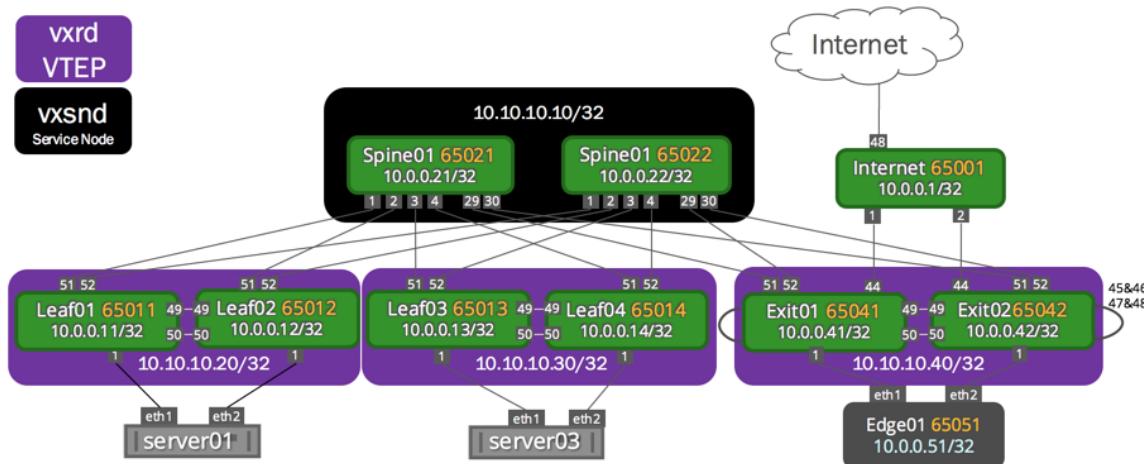
```
auto lo
iface lo inet loopback
  address 10.0.0.12/32
  vxrd-src-ip 10.0.0.12
  vxrd-svcnode-ip 10.10.10.10
  clagd-vxlan-anycast-ip 10.10.10.20
```

Explanation of Variables

Variable	Explanation
vxrd-src-ip	The unique IP address for the <code>vxrd</code> to bind to.
vxrd-svcnode-ip	The service node anycast IP address in the topology. In this demonstration, this is an anycast IP address being shared by both spine switches.

Variable	Description
clagd-vxlan-anycast-ip	The anycast address for the MLAG pair to share and bind to when MLAG is up and running.

Example VXLAN Active-Active Configuration



Note the configuration of the local IP address in the VXLAN interfaces below. They are configured with individual IP addresses, which `clagd` changes to anycast upon MLAG peering.

Quagga Configuration

The layer 3 fabric can be configured using [BGP](#) (see page 406) or [OSPF](#) (see page 406). The following example uses BGP Unnumbered. The MLAG switch configuration for the topology above is shown below.

Layer 3 IP Addressing

The IP address configuration for this example:

spine01: /etc/network/interfaces	spine02: /etc/network/interfaces
<pre>auto lo iface lo inet loopback address 10.0.0.21/32 address 10.10.10.10/32 auto eth0</pre>	<pre>auto lo iface lo inet loopback address 10.0.0.22/32 address 10.10.10.10/32 auto eth0</pre>

```

iface eth0 inet dhcp

# downlinks
auto swp1
iface swp1

auto swp2
iface swp2

auto swp3
iface swp3

auto swp4
iface swp4

auto swp29
iface swp29

auto swp30
iface swp30

```

```

iface eth0 inet dhcp

# downlinks
auto swp1
iface swp1

auto swp2
iface swp2

auto swp3
iface swp3

auto swp4
iface swp4

auto swp29
iface swp29

auto swp30
iface swp30

```

leaf01: /etc/network/interfaces

```

auto lo
iface lo inet loopback
    address 10.0.0.11/32
    vxrd-src-ip 10.0.0.11
    vxrd-svcnode-ip 10.10.10.10
    clagd-vxlan-anycast-ip
10.10.10.20

auto eth0
iface eth0 inet dhcp

# peerlinks
auto swp49
iface swp49

auto swp50
iface swp50

auto peerlink
iface peerlink
    bond-slaves swp49 swp50
    bond-mode 802.3ad
    bond-miimon 100
    bond-use-carrier 1
    bond-lacp-rate 1

```

leaf02: /etc/network/interfaces

```

auto lo
iface lo inet loopback
    address 10.0.0.12/32
    vxrd-src-ip 10.0.0.12
    vxrd-svcnode-ip 10.10.10.10
    clagd-vxlan-anycast-ip
10.10.10.20

auto eth0
iface eth0 inet dhcp

# peerlinks
auto swp49
iface swp49

auto swp50
iface swp50

auto peerlink
iface peerlink
    bond-slaves swp49 swp50
    bond-mode 802.3ad
    bond-miimon 100
    bond-use-carrier 1
    bond-lacp-rate 1

```



```
bond-min-links 1
bond-xmit-hash-policy
layer3+4

auto peerlink.4094
iface peerlink.4094
    address 169.254.1.1/30
    clagd-peer-ip 169.254.1.2
    clagd-backup-ip 10.0.0.12
    clagd-sys-mac 44:39:39:FF:40:
94

# Downlinks
auto swp1
iface swp1

auto bond0
iface bond0
    bond-slaves swp1
    clag-id 1
    bond-miimon 100
    bond-min-links 1
    bond-mode 802.3ad
    bond-xmit-hash-policy
layer3+4
    bond-lacp-rate 1

# bridges / vlan that contain
# peerlink and downlinks for L2
# connectivity

auto native
iface native
    bridge-ports peerlink bond0
vxlan1
    bridge-stp on
    mstpctl-portbpdufilter
vxlan1=yes
    mstpctl-bpduguard vxlan1=yes

auto vlan10
iface vlan10
    bridge-ports peerlink.10
bond0.10 vxlan10
    bridge-stp on
    mstpctl-portbpdufilter
vxlan10=yes
    mstpctl-bpduguard
vxlan10=yes
```

```
bond-min-links 1
bond-xmit-hash-policy
layer3+4

auto peerlink.4094
iface peerlink.4094
    address 169.254.1.2/30
    clagd-peer-ip 169.254.1.1
    clagd-backup-ip 10.0.0.11
    clagd-sys-mac 44:39:39:FF:40:
94

# Downlinks
auto swp1
iface swp1

auto bond0
iface bond0
    bond-slaves swp1
    clag-id 1
    bond-miimon 100
    bond-min-links 1
    bond-mode 802.3ad
    bond-xmit-hash-policy
layer3+4
    bond-lacp-rate 1

# bridges / vlan that contain
# peerlink and downlinks for L2
# connectivity

auto native
iface native
    bridge-ports peerlink bond0
vxlan1
    bridge-stp on
    mstpctl-portbpdufilter
vxlan1=yes
    mstpctl-bpduguard vxlan1=yes

auto vlan10
iface vlan10
    bridge-ports peerlink.10
bond0.10 vxlan10
    bridge-stp on
    mstpctl-portbpdufilter
vxlan10=yes
    mstpctl-bpduguard
vxlan10=yes
```

```

auto vlan20
iface vlan20
    bridge-ports peerlink.20
bond0.20 vxlan20
    bridge-stp on
    mstpcctl-portbpdufilter
vxlan20=yes
    mstpcctl-bpduguard vxlan20=yes

#vxlan config
auto vxlan1
iface vxlan1
    vxlan-id 1
    vxlan-local-tunnelip
10.0.0.11

auto vxlan10
iface vxlan10
    vxlan-id 10
    vxlan-local-tunnelip
10.0.0.11

auto vxlan20
iface vxlan20
    vxlan-id 20
    vxlan-local-tunnelip
10.0.0.11

# uplinks
auto swp51
iface swp51

auto swp52
iface swp52

```

```

auto vlan20
iface vlan20
    bridge-ports peerlink.20
bond0.20 vxlan20
    bridge-stp on
    mstpcctl-portbpdufilter
vxlan20=yes
    mstpcctl-bpduguard vxlan20=yes

#vxlan config
auto vxlan1
iface vxlan1
    vxlan-id 1
    vxlan-local-tunnelip
10.0.0.12

auto vxlan10
iface vxlan10
    vxlan-id 10
    vxlan-local-tunnelip
10.0.0.12

auto vxlan20
iface vxlan20
    vxlan-id 20
    vxlan-local-tunnelip
10.0.0.12

# uplinks
auto swp51
iface swp51

auto swp52
iface swp52

```

leaf3: /etc/network/interfaces

```

auto lo
iface lo inet loopback
    address 10.0.0.13/32
    vxrd-src-ip 10.0.0.13
    vxrd-svcnode-ip 10.10.10.10
    clagd-vxlan-anycast-ip
10.10.10.30

auto eth0
iface eth0 inet dhcp

```

leaf4: /etc/network/interfaces

```

auto lo
iface lo inet loopback
    address 10.0.0.14/32
    vxrd-src-ip 10.0.0.14
    vxrd-svcnode-ip 10.10.10.10
    clagd-vxlan-anycast-ip
10.10.10.30

auto eth0
iface eth0 inet dhcp

```



```
# peerlinks
auto swp49
iface swp49

auto swp50
iface swp50p

auto peerlink
iface peerlink
    bond-slaves swp49 swp50
    bond-mode 802.3ad
    bond-miimon 100
    bond-use-carrier 1
    bond-lacp-rate 1
    bond-min-links 1
    bond-xmit-hash-policy
layer3+4

auto peerlink.4094
iface peerlink.4094
    address 169.254.1.1/30
    clagd-peer-ip 169.254.1.2
    clagd-backup-ip 10.0.0.14
    clagd-sys-mac 44:39:39:FF:40:
95

# Downlinks
auto swp1
iface swp1

auto bond0
iface bond0
    bond-slaves swp1
    clag-id 1
    bond-miimon 100
    bond-min-links 1
    bond-mode 802.3ad
    bond-xmit-hash-policy
layer3+4
    bond-lacp-rate 1

# bridges / vlan that contain
# peerlink and downlinks for L2
# connectivity

auto native
iface native
    bridge-ports peerlink bond0
vxlan1
    bridge-stp on
```

```
# peerlinks
auto swp49
iface swp49

auto swp50
iface swp50

auto peerlink
iface peerlink
    bond-slaves swp49 swp50
    bond-mode 802.3ad
    bond-miimon 100
    bond-use-carrier 1
    bond-lacp-rate 1
    bond-min-links 1
    bond-xmit-hash-policy
layer3+4

auto peerlink.4094
iface peerlink.4094
    address 169.254.1.2/30
    clagd-peer-ip 169.254.1.1
    clagd-backup-ip 10.0.0.13
    clagd-sys-mac 44:39:39:FF:40:
95

# Downlinks
auto swp1
iface swp1

auto bond0
iface bond0
    bond-slaves swp1
    clag-id 1
    bond-miimon 100
    bond-min-links 1
    bond-mode 802.3ad
    bond-xmit-hash-policy
layer3+4
    bond-lacp-rate 1

# bridges / vlan that contain
# peerlink and downlinks for L2
# connectivity

auto native
iface native
    bridge-ports peerlink bond0
vxlan1
    bridge-stp on
```

```

mstpctl-portbpdufilter
vxlan1=yes
  mstpctl-bpduguard
vxlan1=yes

auto vlan10
iface vlan10
  bridge-ports peerlink.10
bond0.10 vxlan10
  bridge-stp on
  mstpctl-portbpdufilter
vxlan10=yes
  mstpctl-bpduguard
vxlan10=yes

auto vlan20
iface vlan20
  bridge-ports peerlink.20
bond0.20 vxlan20
  bridge-stp on
  mstpctl-portbpdufilter
vxlan20=yes
  mstpctl-bpduguard vxlan20=yes

#vxlan config
auto vxlan1
iface vxlan1
  vxlan-id 1
  vxlan-local-tunnelip
10.0.0.13

auto vxlan10
iface vxlan10
  vxlan-id 10
  vxlan-local-tunnelip
10.0.0.13

auto vxlan20
iface vxlan20
  vxlan-id 20
  vxlan-local-tunnelip
10.0.0.13

# uplinks
auto swp51
iface swp51

auto swp52
iface swp52
  
```

```

mstpctl-portbpdufilter
vxlan1=yes
  mstpctl-bpduguard
vxlan1=yes

auto vlan10
iface vlan10
  bridge-ports peerlink.10
bond0.10 vxlan10
  bridge-stp on
  mstpctl-portbpdufilter
vxlan10=yes
  mstpctl-bpduguard
vxlan10=yes

auto vlan20
iface vlan20
  bridge-ports peerlink.20
bond0.20 vxlan20
  bridge-stp on
  mstpctl-portbpdufilter
vxlan20=yes
  mstpctl-bpduguard vxlan20=yes

#vxlan config
auto vxlan1
iface vxlan1
  vxlan-id 1
  vxlan-local-tunnelip
10.0.0.14

auto vxlan10
iface vxlan10
  vxlan-id 10
  vxlan-local-tunnelip
10.0.0.14

auto vxlan20
iface vxlan20
  vxlan-id 20
  vxlan-local-tunnelip
10.0.0.14

# uplinks
auto swp51
iface swp51

auto swp52
iface swp52
  
```



Quagga Configuration

The service nodes and registration nodes must all be routable between each other. The L3 fabric on Cumulus Linux can either be [BGP \(see page 406\)](#) or [OSPF \(see page 406\)](#). In this example, OSPF is used to demonstrate full reachability.

The Quagga configuration using OSPF:

spine01: /etc/quagga/Quagga.conf

```
!
interface swp1
no ipv6 nd suppress-ra
ipv6 nd ra-interval 3
!
interface swp2
no ipv6 nd suppress-ra
ipv6 nd ra-interval 3
!
interface swp3
no ipv6 nd suppress-ra
ipv6 nd ra-interval 3
!
interface swp4
no ipv6 nd suppress-ra
ipv6 nd ra-interval 3
!
interface swp29
no ipv6 nd suppress-ra
ipv6 nd ra-interval 3
!
interface swp30
no ipv6 nd suppress-ra
ipv6 nd ra-interval 3
!
router bgp 65020
bgp router-id 10.0.0.21
network 10.0.0.21/32
network 10.10.10.10/32
bgp bestpath as-path
multipath-relax
bgp bestpath compare-routerid
bgp default show-hostname
neighbor FABRIC peer-group
neighbor FABRIC remote-as
external
neighbor FABRIC description
Internal Fabric Network
neighbor FABRIC
advertisement-interval 0
neighbor FABRIC timers 1 3
```

spine02: /etc/quagga/Quagga.conf

```
!
interface swp1
no ipv6 nd suppress-ra
ipv6 nd ra-interval 3
!
interface swp2
no ipv6 nd suppress-ra
ipv6 nd ra-interval 3
!
interface swp3
no ipv6 nd suppress-ra
ipv6 nd ra-interval 3
!
interface swp4
no ipv6 nd suppress-ra
ipv6 nd ra-interval 3
!
interface swp29
no ipv6 nd suppress-ra
ipv6 nd ra-interval 3
!
interface swp30
no ipv6 nd suppress-ra
ipv6 nd ra-interval 3
!
router bgp 65020
bgp router-id 10.0.0.22
network 10.0.0.22/32
network 10.10.10.10/32
bgp bestpath as-path
multipath-relax
bgp bestpath compare-routerid
bgp default show-hostname
neighbor FABRIC peer-group
neighbor FABRIC remote-as
external
neighbor FABRIC description
Internal Fabric Network
neighbor FABRIC
advertisement-interval 0
neighbor FABRIC timers 1 3
```

```

neighbor FABRIC timers
connect 3
  neighbor FABRIC capability
  extended-nexthop
  neighbor FABRIC prefix-list
  dc-spine in
  neighbor FABRIC prefix-list
  dc-spine out
  neighbor swp1 interface
  neighbor swp1 peer-group
FABRIC
  neighbor swp2 interface
  neighbor swp2 peer-group
FABRIC
  neighbor swp3 interface
  neighbor swp3 peer-group
FABRIC
  neighbor swp4 interface
  neighbor swp4 peer-group
FABRIC
  neighbor swp29 interface
  neighbor swp29 peer-group
FABRIC
  neighbor swp30 interface
  neighbor swp30 peer-group
FABRIC
!
ip prefix-list dc-spine seq 10
permit 0.0.0.0/0
ip prefix-list dc-spine seq 15
permit 10.0.0.0/24 le 32
ip prefix-list dc-spine seq 20
permit 10.10.10.0/24 le 32
ip prefix-list dc-spine seq 30
permit 172.16.1.0/24
ip prefix-list dc-spine seq 40
permit 172.16.2.0/24
ip prefix-list dc-spine seq 50
permit 172.16.3.0/24
ip prefix-list dc-spine seq 60
permit 172.16.4.0/24
ip prefix-list dc-spine seq
500 deny any
!
```

```

neighbor FABRIC timers
connect 3
  neighbor FABRIC capability
  extended-nexthop
  neighbor FABRIC prefix-list
  dc-spine in
  neighbor FABRIC prefix-list
  dc-spine out
  neighbor swp1 interface
  neighbor swp1 peer-group
FABRIC
  neighbor swp2 interface
  neighbor swp2 peer-group
FABRIC
  neighbor swp3 interface
  neighbor swp3 peer-group
FABRIC
  neighbor swp4 interface
  neighbor swp4 peer-group
FABRIC
  neighbor swp29 interface
  neighbor swp29 peer-group
FABRIC
  neighbor swp30 interface
  neighbor swp30 peer-group
FABRIC
!
ip prefix-list dc-spine seq 10
permit 0.0.0.0/0
ip prefix-list dc-spine seq 15
permit 10.0.0.0/24 le 32
ip prefix-list dc-spine seq 20
permit 10.10.10.0/24 le 32
ip prefix-list dc-spine seq 30
permit 172.16.1.0/24
ip prefix-list dc-spine seq 40
permit 172.16.2.0/24
ip prefix-list dc-spine seq 50
permit 172.16.3.0/24
ip prefix-list dc-spine seq 60
permit 172.16.4.0/24
ip prefix-list dc-spine seq
500 deny any
!
```

leaf01: /etc/quagga/Quagga.conf

!

leaf02: /etc/quagga/Quagga.conf

!

```

interface swp51
    no ipv6 nd suppress-ra
    ipv6 nd ra-interval 3
!
interface swp52
    no ipv6 nd suppress-ra
    ipv6 nd ra-interval 3
!
router bgp 65011
    bgp router-id 10.0.0.11
    network 10.0.0.11/32
    network 172.16.1.0/24
    network 10.10.10.20/32
    bgp bestpath as-path
    multipath-relax
        bgp bestpath compare-routerid
        bgp default show-hostname
        neighbor FABRIC peer-group
        neighbor FABRIC remote-as
    external
        neighbor FABRIC description
Internal Fabric Network
    neighbor FABRIC
    advertisement-interval 0
        neighbor FABRIC timers 1 3
        neighbor FABRIC timers
connect 3
    neighbor FABRIC capability
extended-nexthop
    neighbor FABRIC filter-list
dc-leaf-out out
    neighbor swp51 interface
    neighbor swp51 peer-group
FABRIC
    neighbor swp52 interface
    neighbor swp52 peer-group
FABRIC
!
ip as-path access-list dc-leaf-
out permit ^$*
!
```

leaf03: /etc/quagga/Quagga.conf

```

!
interface swp51
    no ipv6 nd suppress-ra
    ipv6 nd ra-interval 3
```

```

interface swp51
    no ipv6 nd suppress-ra
    ipv6 nd ra-interval 3
!
interface swp52
    no ipv6 nd suppress-ra
    ipv6 nd ra-interval 3
!
router bgp 65012
    bgp router-id 10.0.0.12
    network 10.0.0.12/32
    network 172.16.1.0/24
    network 10.10.10.20/32
    bgp bestpath as-path
    multipath-relax
        bgp bestpath compare-routerid
        bgp default show-hostname
        neighbor FABRIC peer-group
        neighbor FABRIC remote-as
    external
        neighbor FABRIC description
Internal Fabric Network
    neighbor FABRIC
    advertisement-interval 0
        neighbor FABRIC timers 1 3
        neighbor FABRIC timers
connect 3
    neighbor FABRIC capability
extended-nexthop
    neighbor FABRIC filter-list
dc-leaf-out out
    neighbor swp51 interface
    neighbor swp51 peer-group
FABRIC
    neighbor swp52 interface
    neighbor swp52 peer-group
FABRIC
!
ip as-path access-list dc-leaf-
out permit ^$*
```

leaf04: /etc/quagga/Quagga.conf

```

!
interface swp51
    no ipv6 nd suppress-ra
    ipv6 nd ra-interval 3
```

```
!
interface swp52
  no ipv6 nd suppress-ra
  ipv6 nd ra-interval 3
!
router bgp 65013
  bgp router-id 10.0.0.13
  network 10.0.0.13/32
  network 172.16.3.0/24
  network 10.10.10.30/32
  bgp bestpath as-path
multipath-relax
  bgp bestpath compare-routerid
  bgp default show-hostname
  neighbor FABRIC peer-group
  neighbor FABRIC remote-as
external
  neighbor FABRIC description
Internal Fabric Network
  neighbor FABRIC
advertisement-interval 0
  neighbor FABRIC timers 1 3
  neighbor FABRIC timers
connect 3
  neighbor FABRIC capability
extended-nexthop
  neighbor FABRIC filter-list
dc-leaf-out out
  neighbor swp51 interface
  neighbor swp51 peer-group
FABRIC
  neighbor swp52 interface
  neighbor swp52 peer-group
FABRIC
!
ip as-path access-list dc-leaf-
out permit ^$
```

```
!
interface swp52
  no ipv6 nd suppress-ra
  ipv6 nd ra-interval 3
!
router bgp 65014
  bgp router-id 10.0.0.14
  network 10.0.0.14/32
  network 172.16.3.0/24
  network 10.10.10.30/32
  bgp bestpath as-path
multipath-relax
  bgp bestpath compare-routerid
  bgp default show-hostname
  neighbor FABRIC peer-group
  neighbor FABRIC remote-as
external
  neighbor FABRIC description
Internal Fabric Network
  neighbor FABRIC
advertisement-interval 0
  neighbor FABRIC timers 1 3
  neighbor FABRIC timers
connect 3
  neighbor FABRIC capability
extended-nexthop
  neighbor FABRIC filter-list
dc-leaf-out out
  neighbor swp51 interface
  neighbor swp51 peer-group
FABRIC
  neighbor swp52 interface
  neighbor swp52 peer-group
FABRIC
!
ip as-path access-list dc-leaf-
out permit ^$
```

Host Configuration

In this example, the servers are running Ubuntu 14.04. A layer2 bond must be mapped from server01 and server03 to the respective switch. In Ubuntu this is done with subinterfaces.

server01

```
auto lo
iface lo inet loopback
```

server03

```
auto lo
iface lo inet loopback
```



```
auto lo
iface lo inet static
    address 10.0.0.31/32

auto eth0
iface eth0 inet dhcp

auto eth1
iface eth1 inet manual
    bond-master bond0

auto eth2
iface eth2 inet manual
    bond-master bond0

auto bond0
iface bond0 inet static
    bond-slaves none
    bond-miimon 100
    bond-min-links 1
    bond-mode 802.3ad
    bond-xmit-hash-policy
layer3+4
    bond-lacp-rate 1
    address 172.16.1.101/24

auto bond0.10
iface bond0.10 inet static
    address 172.16.10.101/24

auto bond0.20
iface bond0.20 inet static
    address 172.16.20.101/24
```

```
auto lo
iface lo inet static
    address 10.0.0.33/32

auto eth0
iface eth0 inet dhcp

auto eth1
iface eth1 inet manual
    bond-master bond0

auto eth2
iface eth2 inet manual
    bond-master bond0

auto bond0
iface bond0 inet static
    bond-slaves none
    bond-miimon 100
    bond-min-links 1
    bond-mode 802.3ad
    bond-xmit-hash-policy
layer3+4
    bond-lacp-rate 1
    address 172.16.1.103/24

auto bond0.10
iface bond0.10 inet static
    address 172.16.10.103/24

auto bond0.20
iface bond0.20 inet static
    address 172.16.20.103/24
```

Enable the Registration Daemon

The registration daemon (`vxrd`) must be enabled on each ToR switch acting as a VTEP, that is participating in LNV. The daemon is installed by default.

1. Open the `/etc/default/vxrd` configuration file in a text editor.
2. Enable the daemon, then save the file.

```
START=yes
```

3. Restart the `vxrd` daemon.



```
cumulus@leaf:~$ sudo systemctl restart vxrd.service
```

Configuring a VTEP

The registration node was configured earlier in `/etc/network/interfaces`; no additional configuration is typically needed. Alternatively, the configuration can be done in `/etc/vxrd.conf`, which has additional configuration knobs available.

Enable the Service Node Daemon

1. Open the `/etc/default/vxsnd` configuration file in a text editor.
2. Enable the daemon, then save the file:

```
START=yes
```

3. Restart the daemon.

```
cumulus@spine:~$ sudo systemctl restart vxsnd.service
```

Configuring the Service Node

To configure the service node daemon, edit the `/etc/vxsnd.conf` configuration file:

spine01: /etc/vxsnd.conf

```
svcnodes_ip = 10.10.10.10  
src_ip = 10.0.0.21  
svcnodes_peers = 10.0.0.21  
10.0.0.22
```

Full configuration of vxsnd.conf

```
[common]  
# Log level is one of DEBUG,  
INFO, WARNING, ERROR, CRITICAL  
#loglevel = INFO  
# Destination for log  
message. Can be a file name,  
'stdout', or 'syslog'  
#logdest = syslog
```

spine02: /etc/vxsnd.conf

```
svcnodes_ip = 10.10.10.10  
src_ip = 10.0.0.22  
svcnodes_peers = 10.0.0.21  
10.0.0.22
```

Full configuration of vxsnd.conf

```
[common]  
# Log level is one of DEBUG,  
INFO, WARNING, ERROR, CRITICAL  
#loglevel = INFO  
# Destination for log  
message. Can be a file name,  
'stdout', or 'syslog'  
#logdest = syslog
```



```
# log file size in bytes. Used
when logdest is a file
#logfilesize = 512000
# maximum number of log files
stored on disk. Used when
logdest is a file
#logbackupcount = 14
# The file to write the pid.
If using monit, this must
match the one
# in the vxsnd.rc
#pidfile = /var/run/vxsnd.pid
# The file name for the unix
domain socket used for mgmt.
#udsfile = /var/run/vxsnd.sock
# UDP port for vxflld control
messages
#vxflld_port = 10001
# This is the address to which
registration daemons send
control messages for
# registration and/or BUM
packets for replication
svcnode_ip = 10.10.10.10
# Holdtime (in seconds) for
soft state. It is used when
sending a
# register msg to peers in
response to learning a <vni,
addr> from a
# VXLAN data pkt
#holdtime = 90
# Local IP address to bind to
for receiving inter-vxsnd
control traffic
src_ip = 10.0.0.21
[vxsnd]
# Space separated list of IP
addresses of vxsnd to share
state with
svcnode_peers = 10.0.0.21
10.0.0.22
# When set to true, the
service node will listen for
vxlan data traffic
# Note: Use 1, yes, true, or
on, for True and 0, no, false,
or off,
# for False
#enable_vxlan_listen = true
```

```
# log file size in bytes. Used
when logdest is a file
#logfilesize = 512000
# maximum number of log files
stored on disk. Used when
logdest is a file
#logbackupcount = 14
# The file to write the pid.
If using monit, this must
match the one
# in the vxsnd.rc
#pidfile = /var/run/vxsnd.pid
# The file name for the unix
domain socket used for mgmt.
#udsfile = /var/run/vxsnd.sock
# UDP port for vxflld control
messages
#vxflld_port = 10001
# This is the address to which
registration daemons send
control messages for
# registration and/or BUM
packets for replication
svcnode_ip = 10.10.10.10
# Holdtime (in seconds) for
soft state. It is used when
sending a
# register msg to peers in
response to learning a <vni,
addr> from a
# VXLAN data pkt
#holdtime = 90
# Local IP address to bind to
for receiving inter-vxsnd
control traffic
src_ip = 10.0.0.22
[vxsnd]
# Space separated list of IP
addresses of vxsnd to share
state with
svcnode_peers = 10.0.0.21
10.0.0.22
# When set to true, the
service node will listen for
vxlan data traffic
# Note: Use 1, yes, true, or
on, for True and 0, no, false,
or off,
# for False
#enable_vxlan_listen = true
```

```
# When set to true, the
svcnode_ip will be installed
on the loopback
# interface, and it will be
withdrawn when the vxsnd is no
longer in
# service. If set to true,
the svcnode_ip configuration
# variable must be defined.
# Note: Use 1, yes, true, or
on, for True and 0, no, false,
or off,
# for False
#install_svcnode_ip = false
# Seconds to wait before
checking the database to age
out stale entries
#age_check = 90
```

```
# When set to true, the
svcnode_ip will be installed
on the loopback
# interface, and it will be
withdrawn when the vxsnd is no
longer in
# service. If set to true,
the svcnode_ip configuration
# variable must be defined.
# Note: Use 1, yes, true, or
on, for True and 0, no, false,
or off,
# for False
#install_svcnode_ip = false
# Seconds to wait before
checking the database to age
out stale entries
#age_check = 90
```

Considerations for Virtual Topologies Using Cumulus VX

Node ID

vxrd requires a unique `node_id` for each individual switch. This `node_id` is based off of the first interface's MAC address; when using certain virtual topologies like Vagrant, both leaf switches within an MLAG pair can generate the same exact unique `node_id`. One of the `node_ids` must then be configured manually (or make sure the first interface always has a unique MAC address), as they are not unique.

To verify the `node_id` that gets configured by your switch, use the `vxrdctl get config` command:

```
cumulus@leaf01$ vxrdctl get config
{
    "concurrency": 1000,
    "config_check_rate": 60,
    "debug": false,
    "eventlet_backdoor_port": 9000,
    "head_rep": true,
    "holdtime": 90,
    "logbackupcount": 14,
    "logdest": "syslog",
    "logfilesize": 512000,
    "loglevel": "INFO",
    "max_packet_size": 1500,
    "node_id": 13,
    "pidfile": "/var/run/vxrd.pid",
    "refresh_rate": 3,
    "src_ip": "10.2.1.50",
    "svcnode_ip": "10.10.10.10",
```

```

"udsfile": "/var/run/vxrd.sock",
"vxflid_port": 10001
}

```

To set the `node_id` manually:

1. Open `/etc/vxrd.conf` in a text editor.
2. Set the `node_id` value within the `common` section, then save the file:

```

[common]
node_id = 13

```



Ensure that each leaf has a separate `node_id` so that LNV can function correctly.

Bonds with Vagrant

Bonds (or LACP Etherchannels) fail to work in a Vagrant setup unless the link is set to promiscuous mode. This is a limitation on virtual topologies only, and is not needed on real hardware.

```

auto swp49
iface swp49
    #for vagrant so bonds work correctly
    post-up ip link set $IFACE promisc on

auto swp50
iface swp50
    #for vagrant so bonds work correctly
    post-up ip link set $IFACE promisc on

```

For more information on using Cumulus VX and Vagrant, refer to the [Cumulus VX documentation](#).

Troubleshooting with LNV Active-Active

In addition to the [troubleshooting for single-attached LNV](#), there is now the MLAG daemon (`clagd`) to consider. The `clagctl` command gives the output of MLAG behavior and any inconsistencies that may arise between a MLAG pair.

```

cumulus@leaf01$ clagctl
The peer is alive
    Our Priority, ID, and Role: 32768 44:38:39:00:00:35 primary
    Peer Priority, ID, and Role: 32768 44:38:39:00:00:36 secondary
        Peer Interface and IP: peerlink.4094 169.254.1.2
        VxLAN Anycast IP: 10.10.10.30
        Backup IP: 10.0.0.14 (inactive)

```

System MAC: 44:39:39:ff:40:95			
CLAG Interfaces			
Our Interface	Peer Interface	CLAG Id	
Conflicts	Proto-Down Reason		
-----	-----	-----	-----
bond0	bond0	1	
-	-	-	
vxlan20	vxlan20	-	
-	-	-	
vxlan1	vxlan1	-	
-	-	-	
vxlan10	vxlan10	-	
-	-	-	

The additions to normal MLAG behavior are the following:

Output	Explanation
VXLAN Anycast IP: 10.10.10.30	The anycast IP address being shared by the MLAG pair for VTEP termination is in use and is 10.10.10.30.
Conflicts: -	There are no conflicts for this MLAG Interface.
Proto-Down Reason: -	The VXLAN is up and running (there is no Proto-Down).

In the next example the `vxlan-id` on VXLAN10 was switched to the wrong `vxlan-id`. When the `clagctl` command is run, you will see that VXLAN10 went down because this switch was the secondary switch and the peer switch took control of VXLAN. The reason code is `vxlan-single` indicating that there is a `vxlan-id` mis-match on VXLAN10

```
cumulus@leaf02$ clagctl
The peer is alive
    Peer Priority, ID, and Role: 32768 44:38:39:00:00:11 primary
    Our Priority, ID, and Role: 32768 44:38:39:00:00:12 secondary
        Peer Interface and IP: peerlink.4094 169.254.1.1
            VXLAN Anycast IP: 10.10.10.20
                Backup IP: 10.0.0.11 (inactive)
                    System MAC: 44:39:39:ff:40:94
CLAG Interfaces
```

Our Interface Conflicts	Peer Interface Proto-Down Reason	CLAG Id
-	-	-
bond0	bond0	1
-	-	-
vxlan20	vxlan20	-
-	-	-
vxlan1	vxlan1	-
-	-	-
vxlan10	-	-
-	vxlan-single	-

Caveats and Errata

- The VLAN used for the peer link layer 3 subinterface should not be reused for any other interface in the system. A high VLAN ID value is recommended. For more information on VLAN ID ranges, refer to [the section above \(see page 346\)](#).
- Active-active mode only works with LNV in this release. Integration with controller-based VXLANs such as VMware NSX and Midokura MidoNet will be supported in the future.

Related Information

- Network virtualization chapter, Cumulus Linux user guide ([see page 346](#))

LNV Full Example

Lightweight Network Virtualization (LNV) is a technique for deploying [VXLANs \(see page 346\)](#) without a central controller on bare metal switches. This a full example complete with diagram. Please reference the [Lightweight Network Virtualization chapter \(see page 378\)](#) for more detailed information. This full example uses the **recommended way** of deploying LNV, which is to use Anycast to load balance the service nodes.



LNV is a lightweight controller option. Please [contact Cumulus Networks](#) with your scale requirements and we can make sure this is the right fit for you. There are also other controller options that can work on Cumulus Linux.

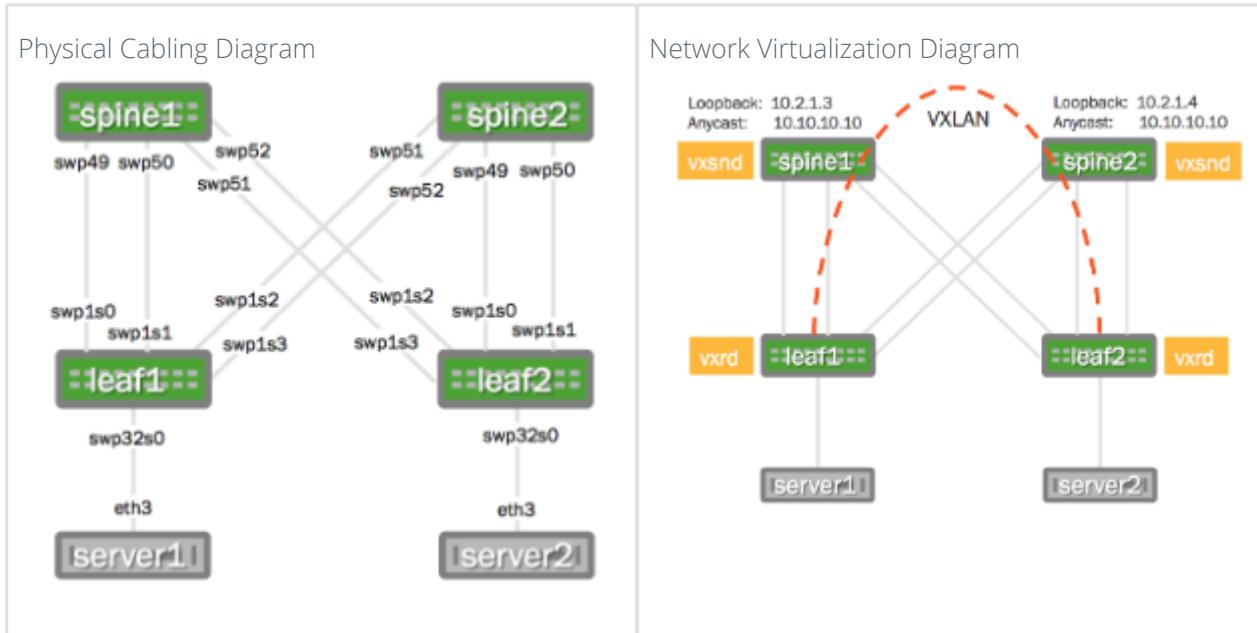
Contents

This chapter covers ...

- Example LNV Configuration ([see page 427](#))
 - Layer 3 IP Addressing ([see page 428](#))
 - Quagga Configuration ([see page 430](#))
 - Host Configuration ([see page 432](#))
 - Service Node Configuration ([see page 433](#))
- Related Information ([see page 434](#))

Example LNV Configuration

The following images illustrate the configuration:



 Want to try out configuring LNV and don't have a Cumulus Linux switch? Check out [Cumulus VX](#).



Feeling Overwhelmed? Come join a [Cumulus Boot Camp](#) and get instructor-led training!

Layer 3 IP Addressing

Here is the configuration for the IP addressing information used in this example:

spine1: /etc/network/interfaces

```

auto lo
iface lo inet loopback
  address 10.2.1.3/32
  address 10.10.10.10/32

auto eth0
iface eth0 inet dhcp

auto swp49
iface swp49
  address 10.1.1.2/30
  
```

spine2: /etc/network/interfaces

```

auto lo
iface lo inet loopback
  address 10.2.1.4/32
  address 10.10.10.10/32

auto eth0
iface eth0 inet dhcp

auto swp49
iface swp49
  address 10.1.1.18/30
  
```



```
auto swp50
iface swp50
    address 10.1.1.6/30

auto swp51
iface swp51
    address 10.1.1.50/30

auto swp52
iface swp52
    address 10.1.1.54/30
```

```
auto swp50
iface swp50
    address 10.1.1.22/30

auto swp51
iface swp51
    address 10.1.1.34/30

auto swp52
iface swp52
    address 10.1.1.38/30
```

leaf1: /etc/network/interfaces

```
auto lo
iface lo inet loopback
    address 10.2.1.1/32
    vxrd-src-ip 10.2.1.1
    vxrd-svcnode-ip 10.10.10.10

auto eth0
iface eth0 inet dhcp

auto swp1s0
iface swp1s0
    address 10.1.1.1/30

auto swp1s1
iface swp1s1
    address 10.1.1.5/30

auto swp1s2
iface swp1s2
    address 10.1.1.33/30

auto swp1s3
iface swp1s3
    address 10.1.1.37/30

auto vni-10
iface vni-10
    vxlan-id 10
    vxlan-local-tunnelip 10.2.1.1
    mstptctl-bpduguard yes
    mstptctl-portbpdufilter yes

auto vni-2000
iface vni-2000
```

leaf2: /etc/network/interfaces

```
auto lo
iface lo inet loopback
    address 10.2.1.2/32
    vxrd-src-ip 10.2.1.2
    vxrd-svcnode-ip 10.10.10.10

auto eth0
iface eth0 inet dhcp

auto swp1s0
iface swp1s0 inet static
    address 10.1.1.17/30

auto swp1s1
iface swp1s1 inet static
    address 10.1.1.21/30

auto swp1s2
iface swp1s2 inet static
    address 10.1.1.49/30

auto swp1s3
iface swp1s3 inet static
    address 10.1.1.53/30

auto vni-10
iface vni-10
    vxlan-id 10
    vxlan-local-tunnelip 10.2.1.2
    mstptctl-bpduguard yes
    mstptctl-portbpdufilter yes

auto vni-2000
iface vni-2000
```

```

vxlan-id 2000
vxlan-local-tunnelip 10.2.1.1
mstpctl-bpduguard yes
mstpctl-portbpdufilter yes

auto vni-30
iface vni-30
  vxlan-id 30
  vxlan-local-tunnelip 10.2.1.1
  mstpctl-bpduguard yes
  mstpctl-portbpdufilter yes

auto br-10
iface br-10
  bridge-ports swp32s0.10 vni-10

auto br-20
iface br-20
  bridge-ports swp32s0.20 vni-2000

auto br-30
iface br-30
  bridge-ports swp32s0.30 vni-30

```

```

vxlan-id 2000
vxlan-local-tunnelip 10.2.1.2
mstpctl-bpduguard yes
mstpctl-portbpdufilter yes

auto vni-30
iface vni-30
  vxlan-id 30
  vxlan-local-tunnelip 10.2.1.2
  mstpctl-bpduguard yes
  mstpctl-portbpdufilter yes

auto br-10
iface br-10
  bridge-ports swp32s0.10 vni-10

auto br-20
iface br-20
  bridge-ports swp32s0.20 vni-2000

auto br-30
iface br-30
  bridge-ports swp32s0.30 vni-30

```

Quagga Configuration

The service nodes and registration nodes must all be routable between each other. The L3 fabric on Cumulus Linux can either be [BGP](#) (see page 516) or [OSPF](#) (see page 500). In this example, OSPF is used to demonstrate full reachability.

Here is the Quagga configuration using OSPF:

spine1: /etc/quagga/Quagga.conf

```

interface lo
  ip ospf area 0.0.0.0
interface swp49
  ip ospf network point-to-point
  ip ospf area 0.0.0.0
!
interface swp50
  ip ospf network point-to-point
  ip ospf area 0.0.0.0
!
```

spine2: /etc/quagga/Quagga.conf

```

interface lo
  ip ospf area 0.0.0.0
interface swp49
  ip ospf network point-to-point
  ip ospf area 0.0.0.0
!
interface swp50
  ip ospf network point-to-point
  ip ospf area 0.0.0.0
!
```



```
interface swp51
  ip ospf network point-to-
  point
  ip ospf area 0.0.0.0
!
interface swp52
  ip ospf network point-to-
  point
  ip ospf area 0.0.0.0
!
!
!
!
!
router-id 10.2.1.3
router ospf
  ospf router-id 10.2.1.3
```

```
interface swp51
  ip ospf network point-to-
  point
  ip ospf area 0.0.0.0
!
interface swp52
  ip ospf network point-to-
  point
  ip ospf area 0.0.0.0
!
!
!
!
!
router-id 10.2.1.4
router ospf
  ospf router-id 10.2.1.4
```

leaf1: /etc/quagga/Quagga.conf

```
interface lo
  ip ospf area 0.0.0.0
interface swp1s0
  ip ospf network point-to-
  point
  ip ospf area 0.0.0.0
!
interface swp1s1
  ip ospf network point-to-
  point
  ip ospf area 0.0.0.0
!
interface swp1s2
  ip ospf network point-to-
  point
  ip ospf area 0.0.0.0
!
interface swp1s3
  ip ospf network point-to-
  point
  ip ospf area 0.0.0.0
!
!
!
!
!
router-id 10.2.1.1
router ospf
```

leaf2: /etc/quagga/Quagga.conf

```
interface lo
  ip ospf area 0.0.0.0
interface swp1s0
  ip ospf network point-to-
  point
  ip ospf area 0.0.0.0
!
interface swp1s1
  ip ospf network point-to-
  point
  ip ospf area 0.0.0.0
!
interface swp1s2
  ip ospf network point-to-
  point
  ip ospf area 0.0.0.0
!
interface swp1s3
  ip ospf network point-to-
  point
  ip ospf area 0.0.0.0
!
!
!
!
!
router-id 10.2.1.2
router ospf
```



```
ospf router-id 10.2.1.1
```

```
ospf router-id 10.2.1.2
```

Host Configuration

In this example, the servers are running Ubuntu 14.04. A trunk must be mapped from server1 and server2 to the respective switch. In Ubuntu this is done with subinterfaces.

server1

```
auto eth3.10
iface eth3.10 inet
static
    address 10.10.10.1/24

auto eth3.20
iface eth3.20 inet
static
    address 10.10.20.1/24

auto eth3.30
iface eth3.30 inet
static
    address 10.10.30.1/24
```

server2

```
auto eth3.10
iface eth3.10 inet
static
    address 10.10.10.2/24

auto eth3.20
iface eth3.20 inet
static
    address 10.10.20.2/24

auto eth3.30
iface eth3.30 inet
static
    address 10.10.30.2/24
```



Service Node Configuration

spine1:/etc/vxsnd.conf

```
[common]
# Log level is one of DEBUG,
INFO, WARNING, ERROR, CRITICAL
#loglevel = INFO
# Destination for log
message. Can be a file name, '
stdout', or 'syslog'
#logdest = syslog
# log file size in bytes. Used
when logdest is a file
#logfilesize = 512000
# maximum number of log files
stored on disk. Used when
logdest is a file
#logbackupcount = 14
# The file to write the pid.
If using monit, this must
match the one
# in the vxsnd.rc
#pidfile = /var/run/vxsnd.pid
# The file name for the unix
domain socket used for mgmt.
#udsfile = /var/run/vxsnd.sock
# UDP port for vxfld control
messages
#vxfld_port = 10001
# This is the address to which
registration daemons send
control messages for
# registration and/or BUM
packets for replication
svcnode_ip = 10.10.10.10
# Holdtime (in seconds) for
soft state. It is used when
sending a
# register msg to peers in
response to learning a <vni,
addr> from a
# VXLAN data pkt
#holdtime = 90
# Local IP address to bind to f
or receiving inter-vxsnd
control traffic
src_ip = 10.2.1.3
```

spine2:/etc/vxsnd.conf

```
[common]
# Log level is one of DEBUG,
INFO, WARNING, ERROR, CRITICAL
#loglevel = INFO
# Destination for log
message. Can be a file name, '
stdout', or 'syslog'
#logdest = syslog
# log file size in bytes. Used
when logdest is a file
#logfilesize = 512000
# maximum number of log files
stored on disk. Used when
logdest is a file
#logbackupcount = 14
# The file to write the pid.
If using monit, this must
match the one
# in the vxsnd.rc
#pidfile = /var/run/vxsnd.pid
# The file name for the unix
domain socket used for mgmt.
#udsfile = /var/run/vxsnd.sock
# UDP port for vxfld control
messages
#vxfld_port = 10001
# This is the address to which
registration daemons send
control messages for
# registration and/or BUM
packets for replication
svcnode_ip = 10.10.10.10
# Holdtime (in seconds) for
soft state. It is used when
sending a
# register msg to peers in
response to learning a <vni,
addr> from a
# VXLAN data pkt
#holdtime = 90
# Local IP address to bind to f
or receiving inter-vxsnd
control traffic
src_ip = 10.2.1.4
```

```
[vxsnd]
# Space separated list of IP
addresses of vxsnd to share
state with
svcnode_peers = 10.2.1.4
# When set to true, the
service node will listen for
vxlan data traffic
# Note: Use 1, yes, true, or
on, for True and 0, no, false,
or off,
# for False
#enable_vxlan_listen = true
# When set to true, the
svcnode_ip will be installed
on the loopback
# interface, and it will be
withdrawn when the vxsnd is no
longer in
# service. If set to true,
the svcnode_ip configuration
# variable must be defined.
# Note: Use 1, yes, true, or
on, for True and 0, no, false,
or off,
# for False
#install_svcnode_ip = false
# Seconds to wait before
checking the database to age
out stale entries
#age_check = 90
```

```
[vxsnd]
# Space separated list of IP
addresses of vxsnd to share
state with
svcnode_peers = 10.2.1.3
# When set to true, the
service node will listen for
vxlan data traffic
# Note: Use 1, yes, true, or
on, for True and 0, no, false,
or off,
# for False
#enable_vxlan_listen = true
# When set to true, the
svcnode_ip will be installed
on the loopback
# interface, and it will be
withdrawn when the vxsnd is no
longer in
# service. If set to true,
the svcnode_ip configuration
# variable must be defined.
# Note: Use 1, yes, true, or
on, for True and 0, no, false,
or off,
# for False
#install_svcnode_ip = false
# Seconds to wait before
checking the database to age
out stale entries
#age_check = 90
```

Related Information

- tools.ietf.org/html/rfc7348
- en.wikipedia.org/wiki/Anycast
- Detailed LNV Configuration Guide (see page 378)
- Cumulus Networks Training
- Network virtualization chapter, Cumulus Linux user guide (see page 346)

Static MAC Bindings with VXLAN

Cumulus Linux includes native Linux VXLAN kernel support.

Contents

This chapter covers ...

- Requirements (see page 435)
- Example VXLAN Configuration (see page 435)
- Configuring the Static MAC Bindings VXLAN (see page 436)
- Troubleshooting VXLANs in Cumulus Linux (see page 437)

Requirements

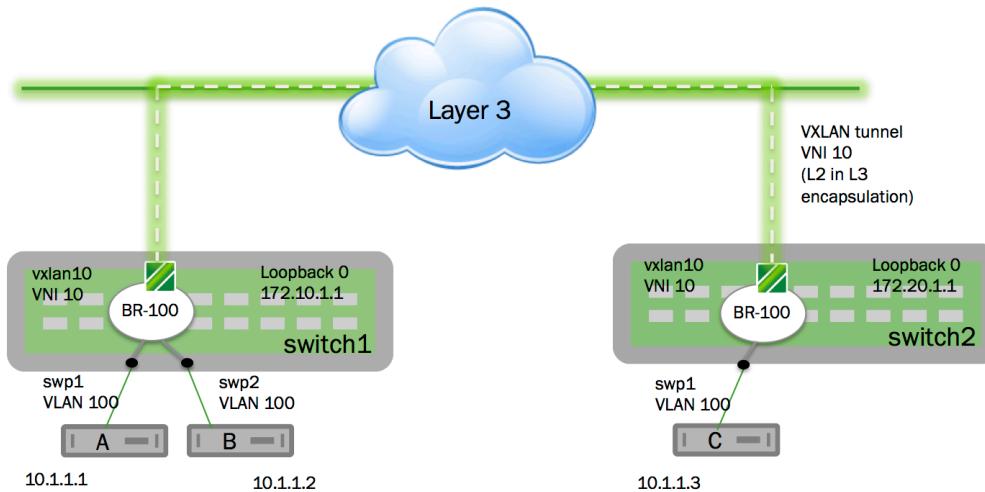
A VXLAN configuration requires a switch with a Broadcom Tomahawk, Trident II+ or Trident II chipset running Cumulus Linux 2.0 or later, or a Mellanox Spectrum chipset running Cumulus Linux 3.2.0 or later.

For a basic VXLAN configuration, you should ensure that:

- The VXLAN has a network identifier (VNI); do not use 0 or 16777215 as the VNI ID, as they are reserved values under Cumulus Linux.
- The VXLAN link and local interfaces are added to bridge to create the association between port, VLAN and VXLAN instance.
- Each bridge on the switch has only one VXLAN interface. Cumulus Linux does not support more than one VXLAN link in a bridge; however a switch can have multiple bridges only if you are using [traditional mode bridges](#) (see page 288).

Example VXLAN Configuration

Consider the following example:



Preconfiguring remote MAC addresses does not scale. A better solution is to use the Cumulus Networks [Lightweight Network Virtualization](#) feature, or a controller-based option like [Midokura MidoNet](#) and [OpenStack](#) or [VMware NSX](#).



Configuring the Static MAC Bindings VXLAN

To configure the example illustrated above, first create the following configuration on switch1:

```
cumulus@switch1:~$ net add loopback lo ip address 172.10.1.1
cumulus@switch1:~$ net add loopback lo vxrd-src-ip 172.10.1.1
cumulus@switch1:~$ net add bridge bridge ports swp1-2
cumulus@switch1:~$ net add bridge post-up bridge fdb add 0:00:10:00:
00:0C dev vtep1000 dst 172.20.1.1 vni 1000
cumulus@switch1:~$ net add vxlan vtep1000 vxlan id 1000
cumulus@switch1:~$ net add vxlan vtep1000 vxlan local-tunnelip
172.10.1.1
cumulus@switch1:~$ net add vxlan vtep1000 bridge access 10
cumulus@switch1:~$ net pending
cumulus@switch1:~$ net commit
```

These commands create the following configuration in the /etc/network/interfaces file:

```
auto vtep1000
iface vtep1000
    vxlan-id 1000
    vxlan-local-tunnelip 172.10.1.1

auto bridge
iface bridge
    bridge-ports swp1 swp2 vtep1000
    bridge-vids 10
    bridge-vlan-aware yes
    post-up bridge fdb add 0:00:10:00:00:0C dev vtep1000 dst 172.20.1.
1 vni 1000
```

Then create the following configuration on switch2:

```
cumulus@switch2:~$ net add loopback lo ip address 172.20.1.1
cumulus@switch2:~$ net add loopback lo vxrd-src-ip 172.20.1.1
cumulus@switch1:~$ net add bridge bridge ports swp1-2
cumulus@switch2:~$ net add bridge post-up bridge fdb add 00:00:10:00:
00:0A dev vtep1000 dst 172.10.1.1 vni 1000
cumulus@switch2:~$ net add bridge post-up bridge fdb add 00:00:10:00:
00:0B dev vtep1000 dst 172.10.1.1 vni 1000
cumulus@switch2:~$ net add vxlan vtep1000 vxlan id 1000
cumulus@switch2:~$ net add vxlan vtep1000 vxlan local-tunnelip
172.10.1.1
cumulus@switch2:~$ net add vxlan vtep1000 bridge access 10
cumulus@switch2:~$ net pending
cumulus@switch2:~$ net commit
```



These commands create the following configuration in the /etc/network/interfaces file:

```
auto vtep1000
iface vtep1000
    vxlan-id 1000
    vxlan-local-tunnelip 172.20.1.1

auto bridge
iface bridge
    bridge-ports swp1 swp2 vtep1000
    bridge-vlan-aware yes
    post-up bridge fdb add 00:00:10:00:00:0A dev vtep1000 dst 172.10.1
.1 vni 1000
    post-up bridge fdb add 00:00:10:00:00:0B dev vtep1000 dst 172.10.1
.1 vni 1000
```

Troubleshooting VXLANS in Cumulus Linux

Use the following commands to troubleshoot issues on the switch:

- `brctl show`: Verifies the VXLAN configuration in a bridge:

```
cumulus@switch:~$ brctl show
bridge name      bridge id          STP enabled     interfaces
bridge           8000.2a179a8cc471   yes            swp1
                                         swp2
                                         vni-10
                                         vni-2000
```

- `bridge fdb show`: Displays the list of MAC addresses in an FDB:

```
cumulus@switch1:~$ bridge fdb show
44:38:39:00:00:18 dev swp1 master bridge permanent
44:38:39:00:00:1c dev swp2 master bridge permanent
2a:17:9a:8c:c4:71 dev vni-2000 master bridge permanent
9a:e8:ef:a1:9d:6f dev vni-10 master bridge permanent
00:00:10:00:00:0c dev vni-10 dst 172.20.1.1 self permanent
```

- `ip -d link show`: Displays information about the VXLAN link:

```
cumulus@switch1:~$ ip -d link show vni-10
15: vni-10: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue
        master bridge state UNKNOWN mode DEFAULT group default
        link/ether 9a:e8:ef:a1:9d:6f brd ff:ff:ff:ff:ff:ff
        promiscuity 1
```

```

vxlan id 10 remote 10.2.1.3 local 10.2.1.1 srcport 0 0
dstport 4789 ageing 300
    bridge_slave state forwarding priority 8 cost 100 hairpin
    off guard off root_block off fastleave off learning on flood on
    port_id 0x8004 port_no 0x4 designated_port 32772 designated_cost
    0 designated_bridge 8000.2a:17:9a:8c:c4:71 designated_root 8000.2
    a:17:9a:8c:c4:71 hold_timer      0.00 message_age_timer      0.00
    forward_delay_timer      0.00 topology_change_ack 0 config_pending
    0 proxy_arp off proxy_arp_wifi off mcast_router 1
    mcast_fast_leave off mcast_flood on addrgenmode eui64

```

Ethernet Virtual Private Network - EVPN



EVPN requires Cumulus Linux version 3.2.1 or newer.



Only VNI values less than 65535 are supported.

Ethernet Virtual Private Network (EVPN) provides a control plane for [VXLANs](#) (see page 346) in Cumulus Linux, with the following functionality:

- VNI membership exchange between VTEPs using the EVPN type-3 (Inclusive Multicast Ethernet Tag) route.
- Exchange of MACs learned on local bridge ports using the EVPN type-2 (MAC/IP Advertisement) route.
- Support for MAC mobility through exchange of the MAC Mobility Extended community.
- Support for dual-attached hosts via VXLAN active-active mode; note that MAC synchronization between the peer switches is done using [MLAG](#) (see page 300).

Contents

This chapter covers ...

- [Installing the EVPN Package](#) (see page 439)
- [Enabling Quagga](#) (see page 440)
- [Configuring EVPN](#) (see page 441)
 - [Enabling EVPN between BGP Neighbors](#) (see page 441)
 - [Enabling EVPN in an iBGP Environment with an OSPF Underlay](#) (see page 441)
 - [Advertising All VNIs](#) (see page 442)
 - [Enabling EVPN with Route Distinguishers \(RDs\) and Route Targets \(RTs\)](#) (see page 442)
 - [Disabling Data Plane MAC Learning over VXLAN Tunnels](#) (see page 443)
- [Handling BUM Traffic](#) (see page 444)
- [EVPN and VXLAN Active-Active Mode](#) (see page 444)
- [Example Configuration](#) (see page 444)

- leaf01 and leaf02 Configurations (see page 445)
- spine01 and spine02 Configurations (see page 448)
- server01 and server02 Configurations (see page 450)
- Testing Connectivity between Servers (see page 450)
- Cumulus Linux Output Commands (see page 451)
- BGP Output Commands (see page 451)
- EVPN Output Commands (see page 452)
 - Displaying EVPN address-family Peers (see page 452)
 - Displaying VNIs (see page 453)
 - Displaying EVPN VXLANs (see page 454)
 - Examining Local and Remote MAC Addresses for a VNI in Quagga (see page 455)
 - Displaying the Global BGP EVPN Routing Table (see page 456)
 - Displaying EVPN Type-2 (MAC/IP) Routes (see page 457)
 - Displaying a Specific EVPN Route (see page 458)
 - Displaying the per-VNI EVPN Routing Table (see page 460)
 - Displaying a Specific MAC or Remote VTEP (see page 460)
- Troubleshooting EVPN (see page 462)
- Caveats (see page 462)

Installing the EVPN Package



Any existing EVPN configuration on the system should be deleted prior to installing this version of EVPN.



Before proceeding, ensure the version of Cumulus Linux is 3.2.1 or higher.

To install EVPN on a switch:

1. Open the `/etc/apt/sources.list` file in a text editor.
2. Uncomment the early access repo lines and save the file:

```
#deb      http://repo3.cumulusnetworks.com/repo CumulusLinux-3-
early-access cumulus
#deb-src  http://repo3.cumulusnetworks.com/repo CumulusLinuz-3-
early-access cumulus
```



EVPN has been engineered and tested to be production-ready. However, as EVPN was made Generally Available (GA) post Cumulus Linux 3.2.1 release, the packages have been left in the Early Access (EA) repository to create minimal end-user disruption, and to ease deployment and upgrade processes. These packages will be moved to the main repository in an upcoming release, and will ship as part of the base image.

3. Run the following commands to install the EVPN package in Cumulus Linux:

```
cumulus@switch:~$ sudo apt-get update
cumulus@switch:~$ sudo apt-get install cumulus-evpn
cumulus@switch:~$ sudo apt-get upgrade
```

4. Check your Quagga version using the `dpkg -l quagga` command. Please make sure you are running version 1.0.0+cl3eau8 or newer.

```
cumulus@leaf01:~$ dpkg -l quagga
Desired=Unknown/Install/Remove/Purge/Hold
| Status=Not/Inst/Conf-files/Unpacked/half-conf/Half-inst/trig-
aWait/Trig-pend
| / Err?=(none)/Reinst-required (Status,Err: uppercase=bad)
|| / Name          Version      Architecture
Description
=====
ii  quagga        1.0.0+cl3eau8    amd64
BGP/OSPF/RIP routing daemon
```

Enabling Quagga

Quagga (see page 489) needs to be enabled prior to using EVPN.

1. Open `/etc/quagga/daemons` file in a text editor:

```
cumulus@switch:~$ sudo nano /etc/quagga/daemons
```

2. Change the `no` to `yes` for `zebra` and `bgpd`, then save the file:

```
zebra=yes
bgpd=yes
```

3. Enable and start Quagga using the `systemctl` commands:

```
cumulus@switch:~$ sudo systemctl enable quagga.service
cumulus@switch:~$ sudo systemctl start quagga.service
```



Configuring EVPN

Enabling EVPN between BGP Neighbors

You enable EVPN between [BGP](#) (see page 516) neighbors by adding the address family `evpn` to the existing neighbor address-family activation command.

For a non-VTEP device, such as a spine switch, that is merely participating in EVPN route exchange (as in, the network deployment uses hop-by-hop eBGP or the switch is acting as an iBGP route reflector), activating the interface for the EVPN address family is the only configuration needed.

This command does not result in BGP knowing about the local VNIs defined on the system and advertising them to peers. This requires additional configuration, as described below.

EVPN supports the configuration of only these two BGP neighbor address-family-specific configurations: `allowas-in` and `route-reflector-client`.

To configure an EVPN route exchange with a BGP peer, the peer or peer-group must be activated within the EVPN address-family in Quagga:

```
router bgp 65000
  address-family evpn
    neighbor swp1 activate
```

Enabling EVPN in an iBGP Environment with an OSPF Underlay

EVPN can be deployed with an [OSPF](#) (see page 500) or static route underlay if needed. This is a more complex configuration than using eBGP. In this case, iBGP advertises EVPN routes directly between VTEPs, and the spines are unaware of EVPN or BGP.

The leaf switches peer with each other in a full mesh within the EVPN address family without using route reflectors. The leafs generally peer to their loopback addresses, which are advertised in OSPF. The receiving VTEP imports routes into a specific VNI with a matching route target community.

```
interface lo
  ip ospf area 0.0.0.0
!
interface swp50
  ip ospf area 0.0.0.0
  ip ospf network point-to-point

interface swp51
  ip ospf area 0.0.0.0
  ip ospf network point-to-point
!

router bgp 65020
  neighbor 10.1.1.2 remote-as internal
  neighbor 10.1.1.3 remote-as internal
  neighbor 10.1.1.4 remote-as internal
!
```

```

address-family evpn
neighbor 10.1.1.2 activate
neighbor 10.1.1.3 activate
neighbor 10.1.1.4 activate
advertise-all-vni
exit-address-family
!
Router ospf
  Ospf router-id 10.1.1.1
  Passive-interface lo

```

Advertising All VNIs

A single configuration variable enables the BGP control plane for all VNIs configured on the switch. Set the variable `advertise-all-vni` to provision all locally configured VNIs to be advertised by the BGP control plane. Quagga is not aware of any local VNIs and MACs associated with that VNI until `advertise-all-vni` is configured.



This configuration is only needed on leaf switches that are VTEPs.

```

router bgp 65000
address-family evpn
  neighbor swp1 activate
  advertise-all-vni

```



EVPN routes received from a BGP peer do get accepted, even without `advertise-all-vni` configured. These routes are maintained in the global EVPN routing table. However, they only become effective (that is, imported into the per-VNI routing table and appropriate entries installed in the kernel) when the VNI corresponding to the received route is locally known.

Enabling EVPN with Route Distinguishers (RDs) and Route Targets (RTs)

EVPN can be provisioned with RDs and RTs either by automatically configuring them, or by manually defining them. In each case, the `advertise-vni` option is used on each VTEP (as shown in the example configuration section described later in this chapter).

VNIs can be configured in Quagga either prior to or after enabling EVPN to advertise all VNIs (through `advertise-all-vni`, as described above). When a local VNI is learned and there is no explicit configuration for that VNI, the route distinguisher (RD) and import and export route targets (RTs) for this VNI are automatically derived — the RD uses "RouterId:VNI" and both RTs use "AS:VNI".

For eBGP EVPN peering, since the peers are in a different AS, using an automatic RT of "AS:VNI" does not work for route import. Thus, the import RT is actually treated as "*:VNI" for determining which received routes are applicable to a particular VNI. This only applies when the import RT is auto-derived and not configured.



Enabling EVPN with Automatic RDs and RTs

The `advertise-all-vni` option is sufficient for provisioning EVPN with RDs and RTs:

```
router bgp 65000
  address-family evpn
    neighbor swp1 activate
    advertise-all-vni
```

Enabling EVPN with User-defined RDs and RTs for Some VNIs

To manually define RDs and RTs, use the `vni` option to configure the switch:

```
router bgp 65100
  address-family evpn
    neighbor SPINE activate
    advertise-vni
    vni 10200
    rd 172.16.10.1:20
    route-target import 65100:20
```



These commands are per VNI and must be specified under `address-family evpn` in BGP.



If you delete the RD or RT later, it reverts back to its corresponding default value.

Disabling Data Plane MAC Learning over VXLAN Tunnels

When EVPN is provisioned, data plane MAC learning should be disabled for VXLAN interfaces to avoid race conditions between control plane learning and data plane learning. In the `/etc/network/interfaces` file, configure the `bridge-learning` value to `off`:

```
auto vxlan200
iface vxlan200
  vxlan-id 10200
  vxlan-local-tunnelip 10.0.0.1
  bridge-learning off
  bridge-access 200
```

Handling BUM Traffic

With EVPN, the only method of handling BUM traffic is [Head End Replication \(HER\)](#) (see page 380). HER is enabled by default, as it is when [Lightweight Network Virtualization \(LNv\)](#) is used.

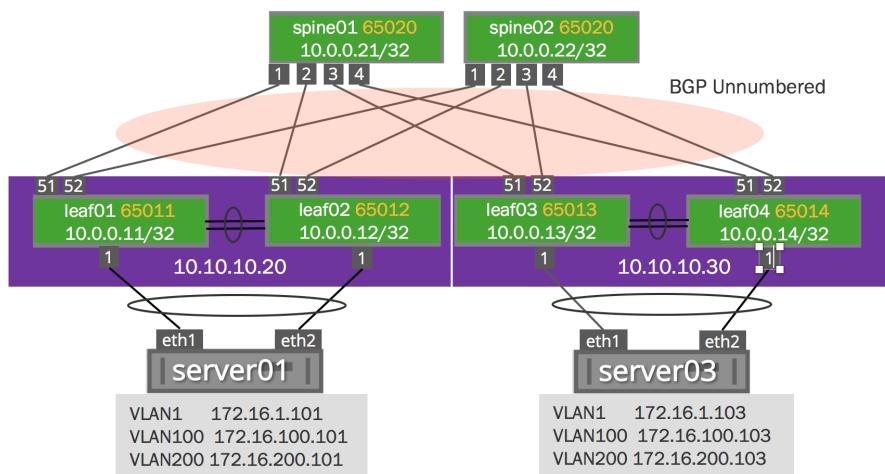
EVPN and VXLAN Active-Active Mode

There is no specific configuration to enable EVPN to work with VXLAN active-active mode. Both switches in the MLAG pair establish EVPN peering with other EVPN speakers (for example, with spine switches, if using hop-by-hop eBGP) and inform about their locally known VNIs and MACs. When MLAG is active, both switches announce this information with the shared anycast IP address.

MLAG syncs information between the two switches in the MLAG pair, EVPN does not synchronize. Therefore, Cumulus Networks recommends you do not configure EVPN peering between the MLAG switches over the peerlink.

Example Configuration

The following configurations are used throughout this chapter. You can find the flat-file configurations for the network devices in the Cumulus Networks [GitHub repository](#). Only a subset is shown here for brevity (not shown are configurations for leaf03, leaf04, server03, server04). Here is the topology diagram:





leaf01 and leaf02 Configurations

leaf01 /etc/network/interfaces

```
auto lo
iface lo inet loopback
    alias loopback
    address 10.0.0.11/32
    clagd-vxlan-anycast-ip
10.10.10.20
auto eth0
iface eth0 inet dhcp
# uplinks
auto swp51
iface swp51
auto swp52
iface swp52
#host connections
auto swp1
iface swp1
auto server01
iface server01
    alias server01 MLAG bond
    bond-slaves swp1
    clag-id 1
auto peerlink
iface peerlink
    alias MLAG peerlink bond
    bond-slaves swp49 swp50
auto peerlink.4094
iface peerlink.4094
    address 169.254.1.1/30
    clagd-peer-ip 169.254.1.2
    clagd-backup-ip
192.168.200.12
    clagd-sys-mac 44:38:39:FF:40:
94
auto bridge
iface bridge
    bridge-ports server01
peerlink vxlan10001 vxlan10100
vxlan10200
    bridge-vlan-aware yes
    bridge-vids 1 100 200
    bridge-pvid 1
auto vxlan10001
iface vxlan10001
    vxlan-id 10001
```

leaf02 /etc/network/interfaces

```
auto lo
iface lo inet loopback
    alias loopback
    address 10.0.0.12/32
    clagd-vxlan-anycast-ip
10.10.10.20
auto eth0
iface eth0 inet dhcp
# uplinks
auto swp51
iface swp51
auto swp52
iface swp52
#host connections
auto swp1
iface swp1
auto server01
iface server01
    alias server01 MLAG bond
    bond-slaves swp1
    clag-id 1
auto peerlink
iface peerlink
    alias MLAG peerlink bond
    bond-slaves swp49 swp50
auto peerlink.4094
iface peerlink.4094
    address 169.254.1.2/30
    clagd-peer-ip 169.254.1.1
    clagd-backup-ip
192.168.200.11
    clagd-sys-mac 44:38:39:FF:40:
94
auto bridge
iface bridge
    bridge-ports server01
peerlink vxlan10001 vxlan10100
vxlan10200
    bridge-vlan-aware yes
    bridge-vids 1 100 200
    bridge-pvid 1
auto vxlan10001
iface vxlan10001
    vxlan-id 10001
```

```

vxlan-local-tunnelip
10.0.0.11
  bridge-access 1
  bridge-learning off
auto vxlan10100
iface vxlan10100
  vxlan-id 10100
  vxlan-local-tunnelip
10.0.0.11
  bridge-access 100
  bridge-learning off
auto vxlan10200
iface vxlan10200
  vxlan-id 10200
  vxlan-local-tunnelip
10.0.0.11
  bridge-access 200
  bridge-learning off

```

```

vxlan-local-tunnelip
10.0.0.12
  bridge-access 1
  bridge-learning off
auto vxlan10100
iface vxlan10100
  vxlan-id 10100
  vxlan-local-tunnelip
10.0.0.12
  bridge-access 100
  bridge-learning off
auto vxlan10200
iface vxlan10200
  vxlan-id 10200
  vxlan-local-tunnelip
10.0.0.12
  bridge-access 200
  bridge-learning off

```

leaf01 /etc/quagga/Quagga.conf

```

!
interface swp51
  ipv6 nd ra-interval 10
  no ipv6 nd suppress-ra
!
interface swp52
  ipv6 nd ra-interval 10
  no ipv6 nd suppress-ra
!
router bgp 65011
  bgp router-id 10.0.0.11
  bgp bestpath as-path
multipath-relax
  neighbor FABRIC peer-group
  neighbor FABRIC remote-as
external
  neighbor FABRIC description
Internal FABRIC Network
  neighbor FABRIC capability
extended-nexthop
  neighbor swp51 interface peer-
group FABRIC
  neighbor swp52 interface peer-
group FABRIC
!
address-family ipv4 unicast
  network 10.0.0.11/32

```

leaf02 /etc/quagga/Quagga.conf

```

!
interface swp51
  ipv6 nd ra-interval 10
  no ipv6 nd suppress-ra
!
interface swp52
  ipv6 nd ra-interval 10
  no ipv6 nd suppress-ra
!
router bgp 65012
  bgp router-id 10.0.0.12
  bgp bestpath as-path
multipath-relax
  neighbor FABRIC peer-group
  neighbor FABRIC remote-as
external
  neighbor FABRIC description
Internal FABRIC Network
  neighbor FABRIC capability
extended-nexthop
  neighbor swp51 interface peer-
group FABRIC
  neighbor swp52 interface peer-
group FABRIC
!
address-family ipv4 unicast
  network 10.0.0.12/32

```

```
network 10.10.10.20/32
exit-address-family
!
!
address-family ipv6 unicast
  neighbor FABRIC activate
exit-address-family
!
address-family evpn
  neighbor FABRIC activate
  advertise-all-vni
exit-address-family
exit
!
line vty
!
end
```

```
network 10.10.10.20/32
exit-address-family
!
!
address-family ipv6 unicast
  neighbor FABRIC activate
exit-address-family
!
address-family evpn
  neighbor FABRIC activate
  advertise-all-vni
exit-address-family
exit
!
line vty
!
end
```

spine01 and spine02 Configurations

spine01 /etc/network/interfaces

```
auto lo
iface lo inet loopback
    address 10.0.0.21/32
auto eth0
iface eth0 inet dhcp
# downlinks
auto swp1
iface swp1
auto swp2
iface swp2
auto swp3
iface swp3
auto swp4
iface swp4
```

spine02 /etc/network/interfaces

```
auto lo
iface lo inet loopback
    address 10.0.0.22/32
auto eth0
iface eth0 inet dhcp
# downlinks
auto swp1
iface swp1
auto swp2
iface swp2
auto swp3
iface swp3
auto swp4
iface swp4
```

spine01 /etc/quagga/Quagga.conf

```
!
log syslog
!
interface swp1
    ipv6 nd ra-interval 10
    no ipv6 nd suppress-ra
!
interface swp2
    ipv6 nd ra-interval 10
    no ipv6 nd suppress-ra
!
interface swp3
    ipv6 nd ra-interval 10
    no ipv6 nd suppress-ra
!
interface swp4
    ipv6 nd ra-interval 10
    no ipv6 nd suppress-ra
!
router bgp 65020
    bgp router-id 10.0.0.21
    bgp bestpath as-path
    multipath-relax
    neighbor fabric peer-group
```

spine02 /etc/quagga/Quagga.conf

```
!
log syslog
!
interface swp1
    ipv6 nd ra-interval 10
    no ipv6 nd suppress-ra
!
interface swp2
    ipv6 nd ra-interval 10
    no ipv6 nd suppress-ra
!
interface swp3
    ipv6 nd ra-interval 10
    no ipv6 nd suppress-ra
!
interface swp4
    ipv6 nd ra-interval 10
    no ipv6 nd suppress-ra
!
router bgp 65020
    bgp router-id 10.0.0.22
    bgp bestpath as-path
    multipath-relax
    neighbor fabric peer-group
```

```

neighbor fabric remote-as
external
    neighbor fabric description
Internal Fabric Network
    neighbor fabric capability
extended-nexthop
    neighbor swp1 interface peer-
group fabric
    neighbor swp2 interface peer-
group fabric
    neighbor swp3 interface peer-
group fabric
    neighbor swp4 interface peer-
group fabric
!
address-family ipv4 unicast
    network 10.0.0.21/32
exit-address-family
!
!
address-family ipv6 unicast
    neighbor fabric activate
exit-address-family
!
address-family evpn
    neighbor fabric activate
exit-address-family
exit
!
line vty
!
end
    
```

```

neighbor fabric remote-as
external
    neighbor fabric description
Internal Fabric Network
    neighbor fabric capability
extended-nexthop
    neighbor swp1 interface peer-
group fabric
    neighbor swp2 interface peer-
group fabric
    neighbor swp3 interface peer-
group fabric
    neighbor swp4 interface peer-
group fabric
!
address-family ipv4 unicast
    network 10.0.0.22/32
exit-address-family
!
!
address-family ipv6 unicast
    neighbor fabric activate
exit-address-family
!
address-family evpn
    neighbor fabric activate
exit-address-family
exit
!
line vty
!
end
    
```



server01 and server02 Configurations

server01 /etc/network/interfaces

```
auto eth0
iface eth0 inet dhcp
auto bond0
iface bond0
    bond-slaves eth1 eth2
    address 172.16.1.101/24
auto bond0.100
iface bond0.100
    address 172.16.100.101
/24
auto bond0.200
iface bond0.200
    address 172.16.200.101
/24
```

server02 /etc/network/interfaces

```
auto eth0
iface eth0 inet dhcp
auto eth2
iface eth2
    address 172.16.1.102/24
auto eth2.100
iface eth2.100
    address 172.16.100.102
/24
auto eth2.200
iface eth2.200
    address 172.16.200.102
/24
```

Testing Connectivity between Servers

SSH to server01 and ping the VLAN1 IP address on server02:

```
user@server01:~$ ping 172.16.1.102
PING 172.16.1.102 (172.16.1.102) 56(84) bytes of data.
64 bytes from 172.16.1.102: icmp_seq=1 ttl=64 time=2.52 ms
64 bytes from 172.16.1.102: icmp_seq=2 ttl=64 time=2.74 ms
^C
--- 172.16.1.102 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1001ms
rtt min/avg/max/mdev = 2.528/2.638/2.749/0.121 ms
```

The following table lists all the servers IP addresses to test connectivity across the L3 fabric:

	server01	server02	server03	server04
VLAN1	172.16.1.101	172.16.1.102	172.16.1.103	172.16.1.104
VLAN100	172.16.100.101	172.16.100.102	172.16.100.103	172.16.100.104
VLAN200	172.16.200.101	172.16.200.102	172.16.200.103	172.16.200.104



Cumulus Linux Output Commands

You can use various `iproute2` commands to examine links, VLAN mappings and displaying bridge FDB data:

- `ip [-d] link show`
- `bridge link show`
- `bridge vlan show`
- `bridge [-s] fdb show`

For example, the output from the following `bridge fdb show` command reveals information relevant only for a VTEP.

- 3 remote VTEPs (10.0.0.5, 10.0.0.6 and 80.80.80.2) for each of the 2 VNIs.
- MAC address 00:02:00:00:00:03 is a local MAC learned over a bond interface while MAC address 00:02:00:00:00:08 is a remote MAC from VTEP 80.80.80.2 for VNI 10100 (vtep100).
- The entries with MAC "00:00:00:00:00:00" are for BUM traffic replication.

```
cumulus@switch:~$ bridge fdb show
00:02:00:00:00:0f dev swp3 master bridge permanent
00:02:00:00:00:01 dev swp3 vlan 1 master bridge
00:02:00:00:00:04 dev peerlink vlan 1 master bridge
00:02:00:00:00:12 dev peerlink master bridge permanent
92:b0:8f:b6:82:7b dev vtep100 master bridge permanent
00:02:00:00:00:08 dev vtep100 vlan 100 master bridge
00:00:00:00:00:00 dev vtep100 dst 10.0.0.5 self permanent
00:00:00:00:00:00 dev vtep100 dst 10.0.0.6 self permanent
00:00:00:00:00:00 dev vtep100 dst 80.80.80.2 self permanent
00:02:00:00:00:08 dev vtep100 dst 80.80.80.2 self
5e:75:42:b8:47:e6 dev vtep200 master bridge permanent
00:00:00:00:00:00 dev vtep200 dst 10.0.0.5 self permanent
00:00:00:00:00:00 dev vtep200 dst 10.0.0.6 self permanent
00:00:00:00:00:00 dev vtep200 dst 80.80.80.2 self permanent
00:02:00:00:00:10 dev bond0 master bridge permanent
02:02:00:00:00:03 dev bond0 vlan 1 master bridge
02:02:00:00:00:02 dev bond0 vlan 1 master bridge
00:02:00:00:00:03 dev bond0 vlan 100 master bridge
```

BGP Output Commands

The following commands are not unique to EVPN but help troubleshoot connectivity and route propagation. You can display the L3 fabric by running the Quagga `show ip bgp summary` command on one of the spines:

```
cumulus@spine01:~$ sudo vtysh
spine01# show ip bgp sum
BGP router identifier 10.0.0.21, local AS number 65020 vrf-id 0
BGP table version 7
```

```
RIB entries 9, using 1152 bytes of memory
Peers 4, using 83 KiB of memory
Peer groups 1, using 72 bytes of memory
Neighbor          V           AS MsgRcvd MsgSent     TblVer  InQ OutQ Up
/Down   State/PfxRcd
leaf01(swp1)    4 65011      1038    1012        0     0     0 00:49:
11             1
leaf02(swp2)    4 65012      1042    1018        0     0     0 00:49:
30             1
leaf03(swp3)    4 65013      1013    995         0     0     0 00:41:
20             1
leaf04(swp4)    4 65014      1026    1012        0     0     0 00:49:
10             1
Total number of neighbors 4
```

You can see the loopback addresses for all the network devices participating in BGP by running the `show ip bgp` command:

```
cumulus@spine01:~$ sudo vtysh
spine01# show ip bgp
BGP table version is 7, local router ID is 10.0.0.21
Status codes: s suppressed, d damped, h history, * valid, > best, =
multipath,
                  i internal, r RIB-failure, S Stale, R Removed
Origin codes: i - IGP, e - EGP, ? - incomplete
Network          Next Hop            Metric LocPrf Weight Path
*> 10.0.0.11/32  swp1              0          0 65011 i
*> 10.0.0.12/32  swp2              0          0 65012 i
*> 10.0.0.13/32  swp3              0          0 65013 i
*> 10.0.0.14/32  swp4              0          0 65014 i
*> 10.0.0.21/32  0.0.0.0           0          0 32768 i
Displayed 5 out of 5 total prefixes
```

EVPN Output Commands

The following commands are unique to EVPN address-families and VXLAN. Note that just because two network nodes are BGP peers does not mean they are EVPN address-family peers or are exchanging VXLAN information.

Displaying EVPN address-family Peers

The network device participating in BGP EVPN address-family can be shown using the `show bgp evpn summary` command

```
cumulus@leaf01:~$ sudo vtysh
leaf01# show bgp evpn summary
BGP router identifier 10.0.0.1, local AS number 65001 vrf-id 0
BGP table version 0
```

```
RIB entries 15, using 1920 bytes of memory
Peers 2, using 42 KiB of memory
Peer groups 1, using 72 bytes of memory
```

Neighbor /Down	V State/PfxRcd	AS	MsgRcvd	MsgSent	TblVer	InQ	OutQ	Up
s1(swp1) 29	4 65100 6	16	22		0	0	0	00:01:
s2(swp2) 28	4 65100 6	16	22		0	0	0	00:01:

```
Total number of neighbors 2
```



You cannot configure EVPN address families within a VRF (see page 574).

Displaying VNIs

You can display the configured VNIs on a network device participating in BGP EVPN by running the `show bgp evpn vni` command. This command works only when run on a VTEP.

The following example examines leaf01, where 3 VNIs are configured; VNIs 10100 and 10200 are purely system-defined in `/etc/network/interfaces`. The RD and RT values for these VNIs have been automatically derived as described earlier. The * indicates that both VNIs exist in the kernel, whereas VNI 20100 is user-configured in BGP with a non-default RD and export RT, but does not yet exist in the kernel:

```
cumulus@leaf01:~$ sudo vtysh
leaf01# show bgp evpn vni
Advertise All VNI flag: Enabled
Number of VNIs: 3
Flags: * - Kernel
      VNI      Orig IP      RD          Import
      RT      Export RT
* 10200    80.80.80.1   10.0.0.1:10200   65001:
10200      65001:10200
  20100    10.0.0.1      65001:20100   65001:
20100      1:20100
* 10100    80.80.80.1   10.0.0.1:10100   65001:
10100      65001:10100
```

```
leaf01# show bgp evpn vni 10100
VNI: 10100 (known to the kernel)
RD: 10.0.0.1:10100
Originator IP: 80.80.80.1
Import Route Target:
  65001:10100
```

**Export Route Target:**

65001:10100

```
leaf01# show bgp evpn vni 20100
VNI: 20100
RD: 65001:20100
Originator IP: 10.0.0.1
Import Route Target:
 65001:20100
Export Route Target:
 1:20100
```

The corresponding BGP configuration for VNI 20100 is follows (only the EVPN section is shown):

```
address-family evpn
  neighbor SPINE activate
  advertise-all-vni
  vni 20100
  rd 65001:20100
  route-target export 1:20100
  exit-vni
exit-address-family
```

Displaying EVPN VXLANs

Run the `show evpn vni [<vni>]` command to list all local configured VXLANs and remote VTEPs. This command only works when run on a VTEP.

The following output indicates that VNI 10100 is present on 3 remote VTEPs (10.0.0.5, 10.0.0.6 and 80.80.80.2) while VNI 10200 is present on 2 remote VTEPs (10.0.0.6 and 80.80.80.2):

```
cumulus@leaf01:~$ sudo vtysh
leaf01# show evpn vni
Number of VNIs: 2
VNI          VxLAN IF           VTEP IP      # MACs   Remote
VTEPs
10200        vtep200          80.80.80.1    2
80.80.80.2

10.0.0.6
10100        vtep100          80.80.80.1    2
80.80.80.2

10.0.0.6
10.0.0.5
```



To see the EVPN configuration for a single VXLAN:

```
cumulus@leaf01:~$ sudo vtysh
leaf01# show evpn vni 10100
VNI: 10100
VxLAN interface: vtep100 ifIndex: 11 VTEP IP: 80.80.80.1
Remote VTEPs for this VNI:
  80.80.80.2
  10.0.0.6
  10.0.0.5
Number of MACs (local and remote) known for this VNI: 2
```

Examining Local and Remote MAC Addresses for a VNI in Quagga

You can examine all local and remote MAC addresses for a VNI by running `show evpn mac vni <vnid>`.

```
cumulus@leaf01:~$ sudo vtysh
leaf01# show evpn mac vni 10100
Number of MACs (local and remote) known for this VNI: 2

MAC          Type    Intf/Remote VTEP      VLAN
00:02:00:00:00:03 local   bond0
00:02:00:00:00:08 remote  80.80.80.2
```

You can examine MAC addresses across VNIs using `show evpn mac vni all`:

```
cumulus@leaf01:~$ sudo vtysh
leaf01# show evpn mac vni all
VNI 10200 #MACs (local and remote) 2

MAC          Type    Intf/Remote VTEP      VLAN
00:02:00:00:00:01 local   swp3
00:02:00:00:00:0b remote  10.0.0.6

VNI 10100 #MACs (local and remote) 2

MAC          Type    Intf/Remote VTEP      VLAN
00:02:00:00:00:03 local   bond0
00:02:00:00:00:08 remote  80.80.80.2
```

You can examine MAC addresses for a remote VTEP and/or query a specific MAC address. This command only works when run on a VTEP:



```
cumulus@leaf01:~$ sudo vtysh
leaf01# show evpn mac vni 10100 mac 00:02:00:00:00:08
MAC: 00:02:00:00:00:08
  Remote VTEP: 80.80.80.2

leadf01# show evpn mac vni 10100 vtep 80.80.80.2
MAC          Type    Intf/Remote VTEP      VLAN
00:02:00:00:00:08 remote 80.80.80.2
```

Displaying the Global BGP EVPN Routing Table

Run the `show bgp evpn route [type <multicast | macip>]` command to display all EVPN routes at the same time:

```
cumulus@leaf01:~$ sudo vtysh
leaf01# show bgp evpn route
BGP table version is 0, local router ID is 10.0.0.1
Status codes: s suppressed, d damped, h history, * valid, > best, i -
internal
Origin codes: i - IGP, e - EGP, ? - incomplete
EVPN type-2 prefix: [2]:[ESI]:[EthTag]:[MAClen]:[MAC]
EVPN type-3 prefix: [3]:[EthTag]:[IPlen]:[OrigIP]
      Network          Next Hop          Metric LocPrf Weight Path
Route Distinguisher: 10.0.0.1:10100
*> [2]:[0]:[0]:[48]:[00:02:00:00:00:03]
              80.80.80.1                  32768 i
*> [3]:[0]:[32]:[80.80.80.1]
              80.80.80.1                  32768 i
Route Distinguisher: 10.0.0.1:10200
*> [2]:[0]:[0]:[48]:[00:02:00:00:00:01]
              80.80.80.1                  32768 i
*> [3]:[0]:[32]:[80.80.80.1]
              80.80.80.1                  32768 i
Route Distinguisher: 10.0.0.3:10100
*> [2]:[0]:[0]:[48]:[00:02:00:00:00:08]
              80.80.80.2                  0 65100
65003 i
* [2]:[0]:[0]:[48]:[00:02:00:00:00:08]
              80.80.80.2                  0 65100
65003 i
* [3]:[0]:[32]:[80.80.80.2]
              80.80.80.2                  0 65100
65003 i
*> [3]:[0]:[32]:[80.80.80.2]
              80.80.80.2                  0 65100
65003 i
Route Distinguisher: 10.0.0.3:10200
* [3]:[0]:[32]:[80.80.80.2]
```



```
          80.80.80.2          0 65100
65003 i
*> [3]:[0]:[32]:[80.80.80.2]
          80.80.80.2          0 65100
65003 i
Route Distinguisher: 10.0.0.4:10100
*> [2]:[0]:[0]:[48]:[00:02:00:00:00:08]
          80.80.80.2          0 65100
65004 i
* [2]:[0]:[0]:[48]:[00:02:00:00:00:08]
          80.80.80.2          0 65100
65004 i
* [3]:[0]:[32]:[80.80.80.2]
          80.80.80.2          0 65100
65004 i
*> [3]:[0]:[32]:[80.80.80.2]
          80.80.80.2          0 65100
65004 i
(truncated)

Displayed 15 prefixes (26 paths)
```

Output Explained

- The output `*> [3]:[0]:[32]:[10.0.0.14]` is explained as follows:

Output	Explanation
[3]	Type 3 EVPN route
[0]	Ethernet tag
[32]	IP address length of 32 bits
10.0.0.14	IPv4 address originating this route

Displaying EVPN Type-2 (MAC/IP) Routes

To display only EVPN type-2 (MAC/IP) routes, run `show bgp evpn route type macip`. The output displays the EVPN route-type fields followed by type-specific fields:

- Type 2 route: [type]:[ESI]:[ET]:[MAC length]:[MAC]
- Type 2 route with ARP suppression: [type]:[ESI]:[ET]:[MAC length]:[MAC]:[IP length]:[IP]
 - The Ethernet Segment Id (ESI) and Ethernet Tag (ET) are always 0.
- Type 3 route: [type]:[ET]:[Originating Router IP]
 - The Ethernet Tag (ET) is always 0.
 - The "Originating Router IP" is the VTEP local IP for the corresponding VNI.

```

cumulus@leaf01:~$ sudo vtysh
leaf01# show bgp evpn route type macip
BGP table version is 0, local router ID is 10.0.0.1
Status codes: s suppressed, d damped, h history, * valid, > best, i -
internal
Origin codes: i - IGP, e - EGP, ? - incomplete
EVPN type-2 prefix: [2]:[ESI]:[EthTag]:[MAClen]:[MAC]
EVPN type-3 prefix: [3]:[EthTag]:[IPlen]:[OrigIP]
      Network          Next Hop          Metric LocPrf Weight Path
Route Distinguisher: 10.0.0.1:10100
*> [2]:[0]:[0]:[48]:[00:02:00:00:00:03]
              80.80.80.1          32768 i
Route Distinguisher: 10.0.0.1:10200
*> [2]:[0]:[0]:[48]:[00:02:00:00:00:01]
              80.80.80.1          32768 i
Route Distinguisher: 10.0.0.3:10100
*> [2]:[0]:[0]:[48]:[00:02:00:00:00:08]
              80.80.80.2          0 65100
65003 i
* [2]:[0]:[0]:[48]:[00:02:00:00:00:08]
              80.80.80.2          0 65100
65003 i
Route Distinguisher: 10.0.0.4:10100
*> [2]:[0]:[0]:[48]:[00:02:00:00:00:08]
              80.80.80.2          0 65100
65004 i
* [2]:[0]:[0]:[48]:[00:02:00:00:00:08]
              80.80.80.2          0 65100
65004 i
Route Distinguisher: 10.0.0.6:10200
*> [2]:[0]:[0]:[48]:[00:02:00:00:00:0b]
              10.0.0.6          0 65100
65006 i
* [2]:[0]:[0]:[48]:[00:02:00:00:00:0b]
              10.0.0.6          0 65100
65006 i
Displayed 5 prefixes (8 paths) (of requested type)

```

Displaying a Specific EVPN Route

To drill down on a specific route for more information, run the `show bgp evpn route rd <VTEP> VXLAN>` command. The following example shows leaf01 receiving a type-2 route and a type-3 route from two spine switches (s1 and s2). The actual remote VTEP is 80.80.80.2, specified in the next hop of the route. Both routes contain the BGP Encapsulation extended community (ET) with value 8 (VxLAN); the type-2 route also carries the VNI (10100).

```

cumulus@leaf01:~$ sudo vtysh
leaf01# show bgp evpn route rd 10.0.0.4:10100

```



```
EVPN type-2 prefix: [2]:[ESI]:[EthTag]:[MAClen]:[MAC]
EVPN type-3 prefix: [3]:[EthTag]:[IPlen]:[OrigIP]
```

```
BGP routing table entry for 10.0.0.4:10100:[2]:[0]:[0]:[48]:[00:02:00:
00:00:08]
Paths: (2 available, best #1)
    Advertised to non peer-group peers:
        s1(swp1) s2(swp2)
    Route [2]:[0]:[0]:[48]:[00:02:00:00:00:08] VNI 10100
        65100 65004
            80.80.80.2 from s1(swp1) (20.0.0.1)
                Origin IGP, localpref 100, valid, external, bestpath-from-AS
                65100, best
                    Extended Community: RT:65004:10100 ET:8
                    AddPath ID: RX 0, TX 45
                    Last update: Thu Jan 12 19:42:58 2017
    Route [2]:[0]:[0]:[48]:[00:02:00:00:00:08] VNI 10100
        65100 65004
            80.80.80.2 from s2(swp2) (20.0.0.2)
                Origin IGP, localpref 100, valid, external
                Extended Community: RT:65004:10100 ET:8
                AddPath ID: RX 0, TX 44
                Last update: Thu Jan 12 19:42:58 2017
```

```
BGP routing table entry for 10.0.0.4:10100:[3]:[0]:[32]:[80.80.80.2]
Paths: (2 available, best #2)
    Advertised to non peer-group peers:
        s1(swp1) s2(swp2)
    Route [3]:[0]:[32]:[80.80.80.2]
        65100 65004
            80.80.80.2 from s2(swp2) (20.0.0.2)
                Origin IGP, localpref 100, valid, external
                Extended Community: RT:65004:10100 ET:8
                AddPath ID: RX 0, TX 31
                Last update: Thu Jan 12 19:04:36 2017
    Route [3]:[0]:[32]:[80.80.80.2]
        65100 65004
            80.80.80.2 from s1(swp1) (20.0.0.1)
                Origin IGP, localpref 100, valid, external, bestpath-from-AS
                65100, best
                    Extended Community: RT:65004:10100 ET:8
                    AddPath ID: RX 0, TX 28
                    Last update: Thu Jan 12 19:04:35 2017
```

```
Displayed 2 prefixes (4 paths) with this RD
```



- Though the local VNI is included in the type-2 route, the receiver does not use it. It uses the received RT to match the route to an appropriate local VNI and then assumes the remote VTEP uses the same VNI value — that is, global VNIs are in use.
- If MAC mobility extended community is exchanged, it gets shown in the above output.
- If the remote MAC is dual attached, the next hop for the EVPN route is the anycast IP address of the remote [MLAG \(see page 300\)](#) pair (when MLAG is active). You can see this in the above example, where 80.80.80.2 is actually the anycast IP of the MLAG pair.

Displaying the per-VNI EVPN Routing Table

Received EVPN routes are maintained in the global EVPN routing table (described above), even if there are no appropriate local VNIs to **import** them into. For example, a spine switch maintains the global EVPN routing table even though there are no VNIs present on it. When local VNIs are present, received EVPN routes are imported into the per-VNI routing tables based on the route target attributes. The per-VNI routing table can be examined using `show bgp evpn route vni <vni> [type <multicast | macip>]`:

```
cumulus@leaf01:~$ sudo vtysh
leaf01# show bgp evpn route vni 10200
BGP table version is 0, local router ID is 10.0.0.1
Status codes: s suppressed, d damped, h history, * valid, > best, i -
internal
Origin codes: i - IGP, e - EGP, ? - incomplete
EVPN type-2 prefix: [2]:[ESI]:[EthTag]:[MAClen]:[MAC]
EVPN type-3 prefix: [3]:[ESI]:[EthTag]:[IPlen]:[OrigIP]
      Network          Next Hop           Metric LocPrf Weight Path
*> [2]:[0]:[48]:[00:02:00:00:00:02]          10.0.0.1            32768 i
*   [2]:[0]:[48]:[00:02:00:00:00:08]          10.0.0.4            0 121000
120004 i
*> [2]:[0]:[48]:[00:02:00:00:00:08]          10.0.0.4            0 121000
120004 i
*> [3]:[0]:[32]:[10.0.0.1]          10.0.0.1            32768 i
*   [3]:[0]:[32]:[10.0.0.2]          10.0.0.2            0 121000
120002 i
...
Displayed 6 prefixes (10 paths)
```

Displaying a Specific MAC or Remote VTEP

You can examine a specific MAC or IP (remote VTEP):



```
cumulus@leaf01:~$ sudo vtysh
leaf01# show bgp evpn route vni 10100 mac 00:02:00:00:00:05
BGP routing table entry for [2]:[0]:[0]:[48]:[00:02:00:00:00:05]
Paths: (2 available, best #2)
Not advertised to any peer
Route [2]:[0]:[0]:[48]:[00:02:00:00:00:05] VNI 10100
Imported from 10.0.0.3:10100:[2]:[0]:[0]:[48]:[00:02:00:00:00:05]
121000 120003
  10.0.0.3 from s2(swp2) (20.0.0.2)
    Origin IGP, localpref 100, valid, external
    Extended Community: RT:120003:10100 ET:8
    AddPath ID: RX 0, TX 44
    Last update: Thu Feb  9 08:16:57 2017

Route [2]:[0]:[0]:[48]:[00:02:00:00:00:05] VNI 10100
Imported from 10.0.0.3:10100:[2]:[0]:[0]:[48]:[00:02:00:00:00:05]
121000 120003
  10.0.0.3 from s1(swp1) (20.0.0.1)
    Origin IGP, localpref 100, valid, external, bestpath-from-AS
121000, best
    Extended Community: RT:120003:10100 ET:8
    AddPath ID: RX 0, TX 42
    Last update: Thu Feb  9 08:16:57 2017

Displayed 2 paths for requested prefix
```

To display the VNI routing table for all VNIs, run `show bgp evpn route vni all`:

```
cumulus@leaf01:~$ sudo vtysh
leaf01# show bgp evpn route vni all
VNI: 10200
BGP table version is 0, local router ID is 10.0.0.1
Status codes: s suppressed, d damped, h history, * valid, > best, i -
internal
Origin codes: i - IGP, e - EGP, ? - incomplete
EVPN type-2 prefix: [2]:[ESI]:[EthTag]:[MAClen]:[MAC]
EVPN type-3 prefix: [3]:[EthTag]:[IPlen]:[OrigIP]
      Network          Next Hop          Metric LocPrf Weight Path
*>  [2]:[0]:[0]:[48]:[00:02:00:00:00:02]          10.0.0.1          32768  i
*   [2]:[0]:[0]:[48]:[00:02:00:00:00:08]          10.0.0.4          0  65100
65004 i
*>  [2]:[0]:[0]:[48]:[00:02:00:00:00:08]          10.0.0.4          0  65100
65004 i
*>  [3]:[0]:[32]:[10.0.0.1]          10.0.0.1          32768  i
*   [3]:[0]:[32]:[10.0.0.2]
```



```
10.0.0.2          0 65100
65002 i
...
Displayed 6 prefixes (10 paths)

VNI: 10100
BGP table version is 0, local router ID is 10.0.0.1
Status codes: s suppressed, d damped, h history, * valid, > best, i -
internal
Origin codes: i - IGP, e - EGP, ? - incomplete
EVPN type-2 prefix: [2]:[ESI]:[EthTag]:[MAClen]:[MAC]
EVPN type-3 prefix: [3]:[EthTag]:[IPlen]:[OrigIP]
      Network           Next Hop           Metric LocPrf Weight Path
*> [2]:[0]:[0]:[48]:[00:02:00:00:00:01]          10.0.0.1          32768 i
*  [2]:[0]:[0]:[48]:[00:02:00:00:00:07]          10.0.0.4          0 65100
65004 i
...
*> [3]:[0]:[32]:[10.0.0.4]          10.0.0.4          0 65100
65004 i

Displayed 6 prefixes (10 paths)
```

Troubleshooting EVPN

The primary way to troubleshoot EVPN is by enabling Quagga debug logs. The relevant debug options are:

- `debug zebra vxlan` — which traces VNI addition and deletion (local and remote) as well as MAC and neighbor addition and deletion (local and remote).
- `debug zebra kernel` — which traces actual netlink messages exchanged with the kernel, which includes everything, not just EVPN.
- `debug bgp updates` — which traces BGP update exchanges, including all updates. Output is extended to show EVPN specific information.
- `debug bgp zebra` - which traces interactions between BGP and zebra for EVPN (and other) routes.

Caveats

The following caveat applies to EVPN in Cumulus Linux 3.2.x:

- Only VNI values less than 65535 are supported.
- EVPN in Cumulus Linux does not support the ability to only announce certain VNIs. It supports either all locally defined VNIs (using the configuration specified here) or none of them. Provisioning options to advertise only specific VNIs will be introduced in a future release.



- Only one import or export RT is allowed for a VNI. Support for additional RTs (for import) will be introduced in a future release.
- Support for one shot configuration of import and export RTs, using `route-target both <rt>` will be added in a future release.

VXLAN Hyperloop

This chapter covers configuring VXLAN gateways using a loopback cable (which we call a *hyperloop*) on non-RIOT (VXLAN routing) capable ASICs running Cumulus Linux.

The Broadcom Trident II and Tomahawk ASICs have a limitation where a L2 bridge that contains a VXLAN interface can not also have an IP address assigned to it. This is an expected limitation with this ASIC, because of the ordering of the decapsulation. A packet that is decapsulated will already have passed the portion of the ASIC capable of reading the IP address lookup (for example, VXLAN lookup happens before IP address lookup). As of Cumulus Linux 3.2.1 even ASICs that are capable of VXLAN routing (such as the Broadcom Trident II+ or Mellanox Spectrum) also require a hyperloop. Please contact your [sales team](#) if there is any confusion. Refer to the [Cumulus Networks Hardware Compatibility List](#) to determine which ASIC is running on the switch.

This limitation will not exist in future ASICs. For example, the Trident II+ has the [RIOT \(Routing In/Out of Tunnels\)](#) feature.

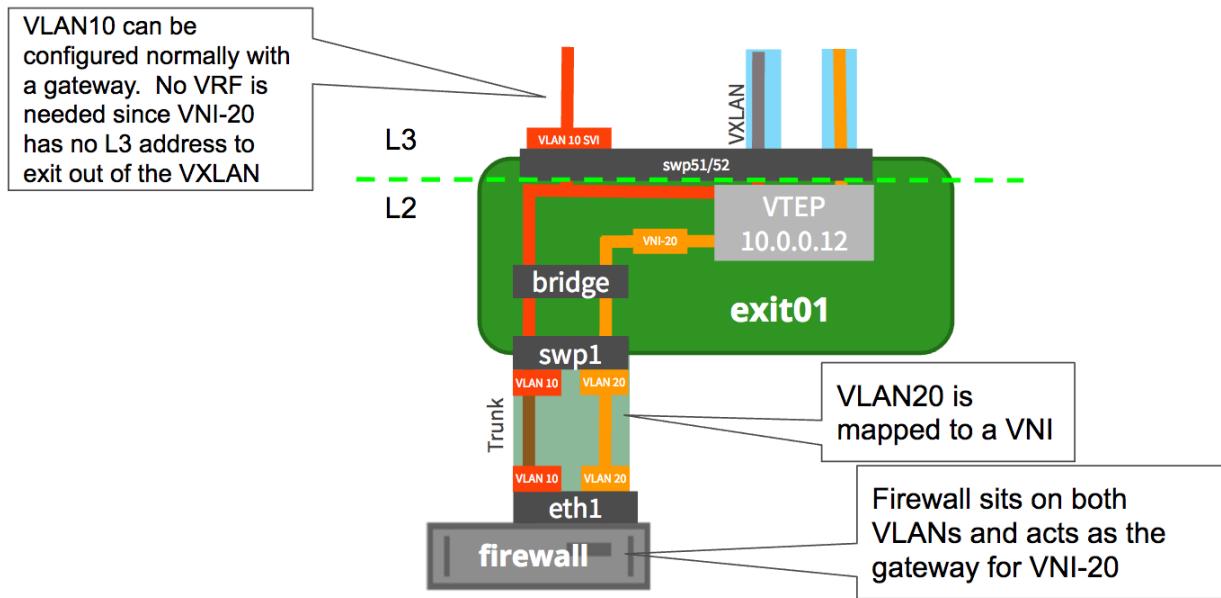


RIOT and VXLAN Routing are not supported in Cumulus Linux 3.2.1, and requires a hyperloop.

Why would you need a hyperloop?

Without native VXLAN routing support, external gateways, firewalls or routers are attached to a VTEP do the routing. (see diagram below)

It is very common in Network Virtualization environments for firewalls to sit on internal VXLAN tied VLANs as well as external VLANs that are routed out to the internet. Look at the following illustration as an example of this method. No special configuration is needed on the switch because the firewall acts as the gateway between internal (VLAN20 / VNI-20 in the example) and the routable external VLAN 10. VLAN10 could have a SVI (Switch Virtual Interface) configured to route out of the VLAN. This also has the benefit of where a VXLAN represents a tenant (or purposely separated application) to keep the firewall between VXLANs so that traffic can be filtered and sanitized to the network operators specification.



With integrated VXLAN routing and bridging using a hyperloop:

- One can avoid having to use external gateways/routers. A hyperloop gives you the ability to do integrated VXLAN routing on a non-RIOT (VXLAN routing) ASIC.
- If applications are hosted on the switch and need L3 connectivity, then a hyperloop can be used as well to provide reachability for this application..

When not to use a hyperloop

- If external firewall is used for routing and security (as shown above), then there is no need for external loopback as firewall takes care of routing across subnets.
- If bandwidth for traffic to be routed is huge that we cannot provision such a high bandwidth using a hyperloop, then one should have dedicated gateways connected to exit leafs.
- If north-south routing involves edge router functionality, then that functionality cannot be provided by leaf switches and so it requires dedicated edge gateways to achieve the same, like NAT.

Contents

This chapter covers ...

- Exiting a VXLAN with a Hyperloop (see page 464)
 - VXLAN Hyperloop (see page 464)
 - (see page 465)
 - Packet Flow Diagram (see page 466)
 - Trident II and Tomahawk switchd Flag (see page 466)
- VXLAN Routing and Hyperloop Troubleshooting Matrix (see page 467)

Exiting a VXLAN with a Hyperloop

VXLAN Hyperloop



This limitation means a physical cable must be attached from one port on leaf1 to another port on leaf1. One port is an L3 port while the other is a member of the bridge. For example, following the configuration above, in order for a layer 3 address to be used as the gateway for vni-10, you could configure the following on exit01:

```
auto lo
iface lo inet loopback
    address 10.0.0.11/32

***some output removed for brevity (e.g. peerlink and host facing
bonds)***

auto bridge
iface bridge
    bridge-vlan-aware yes
    bridge-ports inside server01 server02 vni-10 vni-20 peerlink
    bridge-vids 100 200
    bridge-pvid 1
    mstptctl-treepriority 8192

auto outside
iface outside
    bond-slaves swp45 swp47
    alias hyperloop outside
    mstptctl-bpduguard yes
    mstptctl-portbpdufilter yes

auto inside
iface inside
    bond-slaves swp46 swp48
    alias hyperloop inside
    mstptctl-bpduguard yes
    mstptctl-portbpdufilter yes

auto VLAN100GW
iface VLAN100GW
    bridge_ports outside.100
    address 172.16.100.2/24
    alias VXLAN GW 100 Linux Bridge
    address-virtual 44:39:39:FF:01:90 172.16.100.1/24

auto VLAN200GW
iface VLAN200GW
    bridge_ports outside.200
    address 172.16.200.2/24
    alias VXLAN GW 200 Linux Bridge
    address-virtual 44:39:39:FF:02:90 172.16.200.1/24

auto vni-10
iface vni-10
```

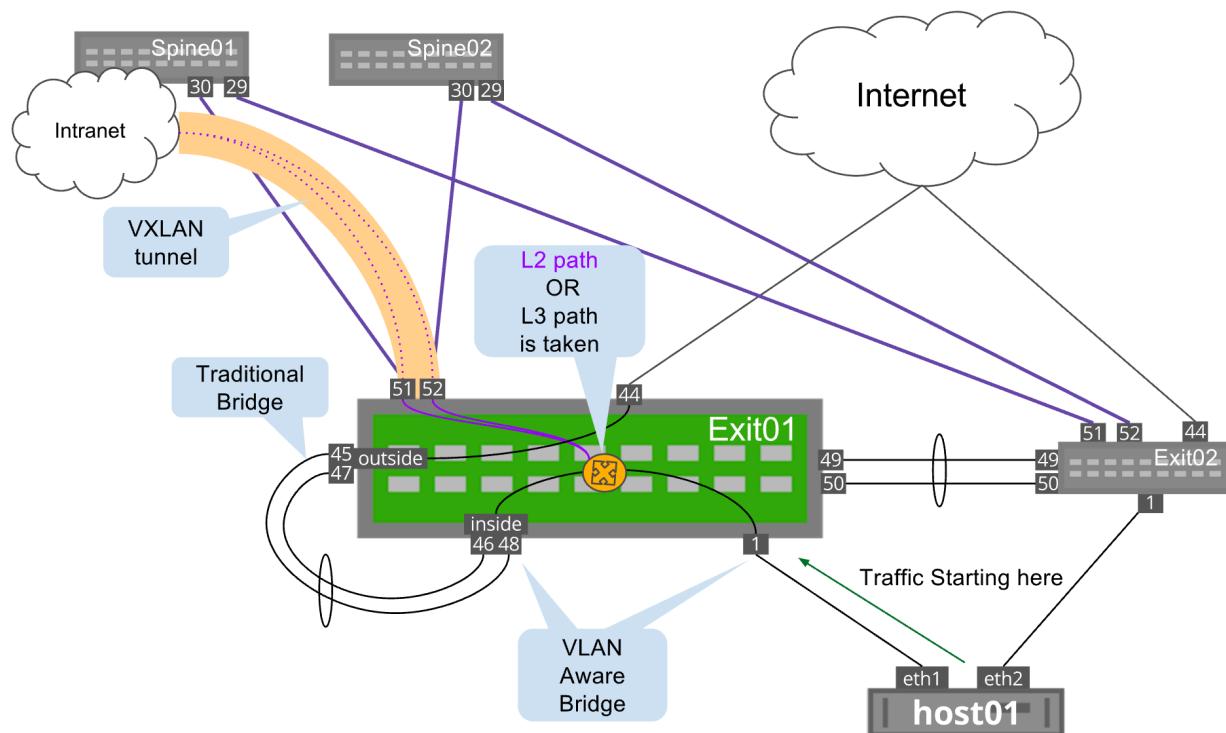
```

vxlan-id 10
vxlan-local-tunnelip 10.0.0.11
bridge-access 100

auto vni-20
iface vni-20
    vxlan-id 20
    vxlan-local-tunnelip 10.0.0.11
    bridge-access 200

```

Packet Flow Diagram



Trident II and Tomahawk switchd Flag

For the Broadcom Trident II and Tomahawk ASICs to be able to have a hyperloop work correctly, you must configure a `switchd` flag. This change is not needed on any other hardware ASIC.

```
cumulus@exit01:mgmt-vrf:/root$ sudo nano /etc/cumulus/switchd.conf
halbcm.per_vlan_router_mac_lookup = TRUE
```

Restart `switchd` for the change to take place:

```
cumulus@exit01:mgmt-vrf:/root$ sudo systemctl restart switchd.service
```



Restarting `switchd` is a disruptive change and affects data plane network traffic.



Turning on the `hal.bcm.per_vlan_router_mac_lookup = TRUE` will limit the Trident 2 switch to a configurable 512 local IP addresses (SVIs, etc) so this should only be used as a last resort. This is only a limitation on this specific ASIC type.

VXLAN Routing and Hyperloop Troubleshooting Matrix

- **Are you running an ASIC capable of VXLAN?**

Must be running one of the following:

- Broadcom Trident 2
- Broadcom Trident 2+
- Broadcom Tomahawk
- Mellanox Spectrum

If you are unsure use a "net show system" as displayed in the following step. If you don't have an ASIC capable of VXLAN you can't encapsulate or decapsulate VXLAN traffic.

- **Are you running the newest code?** This guide and testing was done on Cumulus Linux 3.2.1. Please upgrade the code on the Cumulus Linux system. (see page 43)

```
cumulus@leaf01:mgmt-vrf:~$ cat /etc/lsb-release
DISTRIB_ID="Cumulus Linux"
DISTRIB_RELEASE=3.2.1
DISTRIB_DESCRIPTION="Cumulus Linux 3.2.1"
```

or use [NCLU](#) (see page 498):

```
cumulus@leaf01:mgmt-vrf:~$ net show system
Dell S4048-ON
Cumulus Version 3.2.1
Build: Cumulus Linux 3.2.1
Chipset: Broadcom Trident2 BCM56854
Port Config: 48 x 10G-SFP+ & 6 x 40G-QSFP+
CPU: (x86_64) Intel Atom C2338 1.74GHz
Uptime: 7 days, 21:00:57
```

- **If you are using EVPN are you using the correct Quagga version?**

Use a `dpkg -l` to check the Quagga version as shown below:

```
cumulus@leaf01:mgmt-vrf:~$ dpkg -l quagga
Desired=Unknown/Install/Remove/Purge/Hold
```

```
| Status=Not/Inst/Conf-files/Unpacked/half-conf/Half-inst/trig-
aWait/Trig-pend
| / Err?=(none)/Reinst-required (Status,Err: uppercase=bad)
| | / Name          Version      Architecture
Description
=====
ii  quagga        1.0.0+cl3eau8    amd64
BGP/OSPF/RIP routing daemon
```

You must be running `eau8` as shown above under the Version column. If you are not running the right version of Quagga for EVPN [follow the directions here \(see page 438\)](#).

- **Are HER (Head End Replication) entries being programmed into the bridge fdb table?**
Check for 00:00:00:00:00:00 entries per VXLAN:

```
cumulus@leaf03:mgmt-vrf:~$ bridge fdb show | grep 00:00:00:00:00:
00
00:00:00:00:00:00 dev vni-40 dst 10.10.10.30 self permanent
00:00:00:00:00:00 dev vni-40 dst 10.10.10.40 self permanent
00:00:00:00:00:00 dev vni-1 dst 10.10.10.30 self permanent
00:00:00:00:00:00 dev vni-1 dst 10.10.10.40 self permanent
00:00:00:00:00:00 dev vni-30 dst 10.10.10.30 self permanent
00:00:00:00:00:00 dev vni-30 dst 10.10.10.40 self permanent
00:00:00:00:00:00 dev vni-20 dst 10.10.10.30 self permanent
00:00:00:00:00:00 dev vni-20 dst 10.10.10.40 self permanent
00:00:00:00:00:00 dev vni-50 dst 10.10.10.30 self permanent
00:00:00:00:00:00 dev vni-50 dst 10.10.10.40 self permanent
00:00:00:00:00:00 dev vni-10 dst 10.10.10.30 self permanent
00:00:00:00:00:00 dev vni-10 dst 10.10.10.40 self permanent
```

or use [NCLU \(see page 498\)](#):

```
cumulus@leaf03:mgmt-vrf:~$ net show bridge macs
VLAN      Master      Interface      MAC
TunnelDest   State      Flags      LastSeen
-----  -----  -----
-----  -----  -----
1         bridge      server03     90:e2:ba:7e:96:
                                         00:01:07
d9
untagged      vni-1      00:00:00:00:00:00  10.10.10.30
permanent    self       4 days, 22:25:42
untagged      vni-1      00:00:00:00:00:00  10.10.10.40
permanent    self       4 days, 22:25:42
untagged      vni-10     00:00:00:00:00:00  10.10.10.30
permanent    self       4 days, 22:25:42
untagged      vni-10     00:00:00:00:00:00  10.10.10.40
permanent    self       4 days, 22:25:42
untagged      vni-20     00:00:00:00:00:00  10.10.10.30
permanent    self       4 days, 22:25:42
```

untagged	vni-20	00:00:00:00:00:00	10.10.10.40
permanent self	4 days, 22:25:42		
untagged	vni-30	00:00:00:00:00:00	10.10.10.30
permanent self	4 days, 22:25:42		
untagged	vni-30	00:00:00:00:00:00	10.10.10.40
permanent self	4 days, 22:25:42		
untagged	vni-40	00:00:00:00:00:00	10.10.10.30
permanent self	4 days, 22:25:42		
untagged	vni-40	00:00:00:00:00:00	10.10.10.40
permanent self	4 days, 22:25:42		
untagged	vni-50	00:00:00:00:00:00	10.10.10.30
permanent self	4 days, 22:25:42		
untagged	vni-50	00:00:00:00:00:00	10.10.10.40
permanent self	4 days, 22:25:42		
untagged bridge	peerlink	2c:60:0c:72:eb:	
a0	permanent	5 days, 00:53:13	
untagged bridge	server03	2c:60:0c:72:eb:70	
permanent	5 days, 00:53:13		
untagged bridge	vni-1	86:c9:5c:cc:88:54	
permanent	4 days, 22:28:23		
untagged bridge	vni-10	32:d5:3a:99:36:	
f7	permanent	4 days, 22:28:23	
untagged bridge	vni-20	9a:91:5a:7e:0f:	
e8	permanent	4 days, 22:28:23	
untagged bridge	vni-30	6a:33:ff:fd:ca:34	
permanent	4 days, 22:28:23		
untagged bridge	vni-40	e2:1f:a4:7c:75:	
2b	permanent	4 days, 22:28:23	
untagged bridge	vni-50	d6:df:b4:85:4d:55	
permanent	4 days, 22:28:23		

If you are not getting HER entries, there is several things you can try:

- Make sure you are using LNV **OR** EVPN. You can only use one or the other, not both at the same time.
- Make sure you are not trying to use any VNI/VXLAN values over 65535. For example, VXLAN 70000 is not supported in Cumulus Linux 3.2.1.
- Make sure you are not using the reserved VLAN range, by default this is 3000-3999. The value can be seen in /etc/cumulus/switchd.conf for the `resv_vlan_range` variable.
- **If you are using an MLAG VTEP (dual attached), is it set up correctly?**

Check the outputs. Often, when VXLAN is considered to be non-working, it's actually due to an incorrect setup on the server OS, whether it's Ubuntu, Microsoft Windows or RHEL.

```
cumulus@leaf03:mgmt-vrf:~$ clagctl
The peer is alive
Our Priority, ID, and Role: 32768 2c:60:0c:72:eb:a0 primary
Peer Priority, ID, and Role: 32768 cc:37:ab:2c:95:cb secondary
Peer Interface and IP: peerlink.4094 169.254.255.2
VxLAN Anycast IP: 10.10.10.20
Backup IP: leaf04 vrf mgmt (active)
System MAC: 44:39:39:ff:41:94
```

VXLAN Anycast IP should match on both MLAG Peers

CLAG Interfaces Our Interface	Peer Interface	CLAG Id	Conflicts	Proto-Down Reason
vni-40	vni-40	-	-	-
vni-1	vni-1	-	-	-
vni-30	vni-30	-	-	-
vni-20	vni-20	-	-	-
vni-50	vni-50	-	-	-
server03	server03	1	-	-
vni-10	vni-10	-	-	-

Our Interface and Peer Interface should match or the bond is down on the server side

The output should read "The peer is alive"

Backup IP should be active and System MAC should match between peers

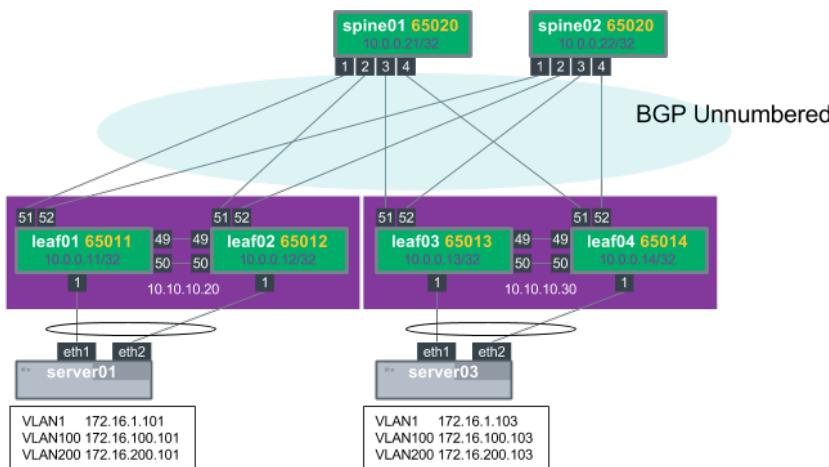
There should be nothing in the Proto-Down State

Every physical server should have a unique CLAG Id

There should be no Conflicts

- Can you ping from host to host on the same VXLAN?**

For example, in the following network diagram, can server01 ping to server03 on any of the VLANs (VLAN1, VLAN100, VLAN200)?



If you can't even ping from server to server this is not a VXLAN gateway problem, but a problem with the network itself. This must be fixed prior to making a VXLAN gateway with or without a hyperloop.



Only proceed past this point if you can get server to server connectivity on the same VXLAN



- **Are you on a Broadcom Trident II?**

If and only if you are using a switch with a Broadcom Trident II ASIC, you must set the /etc/cumulus/switchd.conf flag for halbcm.per_vlan_router_mac_lookup = TRUE and restart switchd.

- **Is the SVI on a physical interface or on a traditional bridge?**



The SVI (Switch VLAN interface) IP address for a hyperloop MUST be on a traditional bridge. Please follow the configuration guidelines on the above.

- **Is the port plugged in where it is supposed to be plugged in?**

Use lldpctl or net show lldp to see what ports are hooked up.

```
cumulus@leaf03:mgmt-vrf:~$ net show lldp
  Local Port      Speed     Mode          Remote Port
  Remote Host    Summary
-----  -----  -----  -----
-----  -----
eth0        1G      Mgmt      ===  swp42
oob-mgmt-switch  IP: 10.50.100.53/24 (DHCP)
swp1        10G     BondMember ===  90:e2:ba:7e:96:d8
server03           Master: server03(UP)
swp49        40G     BondMember ===  swp49
leaf04           Master: peerlink(UP)
swp50        40G     BondMember ===  swp50
leaf04           Master: peerlink(UP)
swp51        40G     NotConfigured ===  swp3
spine01           ...
swp52        40G     NotConfigured ===  swp3
spine02           ...
swp53        40G     NotConfigured ===  swp54
leaf03           ...
swp54        40G     NotConfigured ===  swp53
leaf03           ...
```

Notice above that swp53 and swp54 are a loopback cable (hyperloop) where it is connected to itself.

- **Is the VRR MAC address unique per subnet?**

Make sure that [VRR is configured correctly](#) and that each MAC address is unique per VLAN.

Static VXLAN Tunnels

In VXLAN-based networks, there are a range of complexities and challenges in determining the destination *virtual tunnel endpoints* (VTEPs) for any given VXLAN. At scale, various solutions, including [Lightweight Network Virtualization \(see page 378\)](#) (LNV), controller-based options like [Midokura MidoNet \(see page 361\)](#) or [VMware NSX \(see page 348\)](#) and even new standards like [EVPN \(see page 438\)](#) are attempts to address these complexities, however do retain their own complexities.

Enter *static VXLAN tunnels*, which simply serve to connect two VTEPs in a given environment. Static VXLAN tunnels are the simplest deployment mechanism for small scale environments, and they should be interoperable with other vendors that adhere to VXLAN standards. It's an easy configuration, since you are simply mapping which VTEPs are in a particular VNI, so you can avoid the tedious process of defining connections to every VLAN on every other VTEP on every other rack.

Contents

This chapter covers ...

- Requirements (see page 472)
- Example Configuration (see page 472)
- Configuring Static VXLAN Tunnels (see page 473)
- Verifying the Configuration (see page 477)

Requirements

While they should be interoperable with other vendors, Cumulus Networks supports static VXLAN tunnels only on switches in the [Cumulus Linux HCL](#) using the Broadcom Tomahawk, Trident II+ and Trident II chipsets as well as the Mellanox Spectrum chipset.

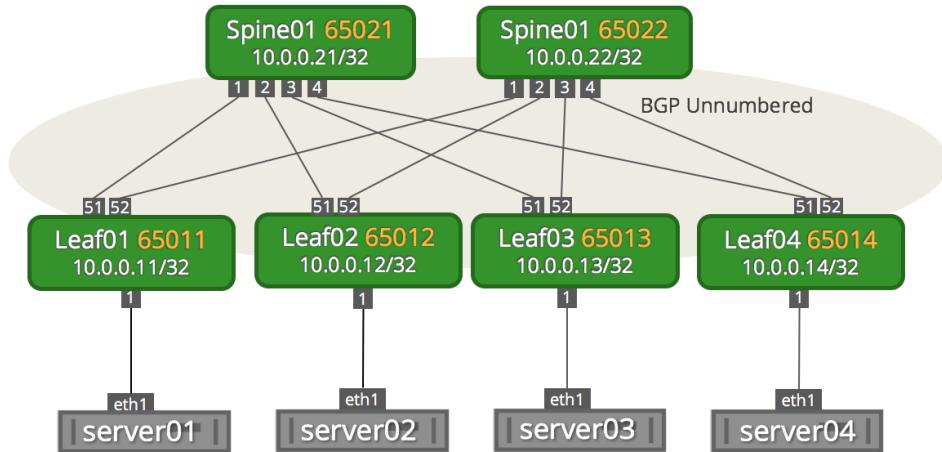
For a basic VXLAN configuration, you should ensure that:

- The VXLAN has a network identifier (VNI); do not use 0 or 16777215 as the VNI ID, as they are reserved values under Cumulus Linux.
- The VXLAN link and local interfaces are added to bridge to create the association between port, VLAN and VXLAN instance.
- Each bridge on the switch has only one VXLAN interface. Cumulus Linux does not support more than one VXLAN link in a bridge.
- The VXLAN registration daemon (`vxrd`) is not running. Static VXLAN tunnels do not interoperate with LNV or EVPN. If vxrd is running, stop it with:

```
cumulus@switch:~ sudo systemctl stop vxrd.service
```

Example Configuration

The following topology is used in this chapter. Each IP address corresponds to the switch's loopback address:



Configuring Static VXLAN Tunnels

To configure static VXLAN tunnels, for each leaf you need to do the following:

- Specify an IP address for the loopback
- Create a VXLAN interface, using the loopback address for the local tunnel IP address
- Create the tunnels by configuring the remote IP address to each other leaf switch's loopback address

To configure leaf01, run the following commands:

```
cumulus@leaf01:~$ net add loopback lo ip address 10.0.0.11/32
cumulus@leaf01:~$ net add vxlan vni-10 vxlan id 10
cumulus@leaf01:~$ net add vxlan vni-10 vxlan local-tunnelip 10.0.0.11
cumulus@leaf01:~$ net add vxlan vni-10 vxlan remoteip 10.0.0.12
cumulus@leaf01:~$ net add vxlan vni-10 vxlan remoteip 10.0.0.13
cumulus@leaf01:~$ net add vxlan vni-10 vxlan remoteip 10.0.0.14
cumulus@leaf01:~$ net add vxlan vni-10 bridge access 10
cumulus@leaf01:~$ net pending
cumulus@leaf01:~$ net commit
```

These commands create the following configuration in the /etc/network/interfaces file:

```
# The loopback network interface
auto lo
iface lo inet loopback
    address 10.0.0.11/32

# The primary network interface
auto eth0
iface eth0 inet dhcp

auto swp1
iface swp1
```

```

auto swp2
iface swp2

auto bridge
iface bridge
    bridge-ports vni-10
    bridge-vids 10
    bridge-vlan-aware yes

auto vni-10
iface vni-10
    bridge-access 10
    mstpcctl-bpduguard yes
    mstpcctl-portbpdufilter yes
    vxlan-id 10
    vxlan-local-tunnelip 10.0.0.11
    vxlan-remoteip 10.0.0.12
    vxlan-remoteip 10.0.0.13
    vxlan-remoteip 10.0.0.14

```

Repeat these steps for leaf02, leaf03 and leaf04:

Node	NCLU Commands	/etc/network/interfaces Configuration
leaf02	<pre> cumulus@leaf02:~\$ net add loopback lo ip address 10.0.0.12/32 cumulus@leaf02:~\$ net add vxlan vni-10 vxlan id 10 cumulus@leaf02:~\$ net add vxlan vni-10 vxlan local-tunnelip 10.0.0.12 cumulus@leaf02:~\$ net add vxlan vni-10 vxlan remoteip 10.0.0.11 cumulus@leaf02:~\$ net add vxlan vni-10 vxlan remoteip 10.0.0.13 cumulus@leaf02:~\$ net add vxlan vni-10 vxlan remoteip 10.0.0.14 cumulus@leaf02:~\$ net add vxlan vni-10 bridge access 10 cumulus@leaf02:~\$ net pending cumulus@leaf02:~\$ net commit </pre>	<pre> # The loopback network interface auto lo iface lo inet loopback address 10.0.0.12/32 # The primary network interface auto eth0 iface eth0 inet dhcp auto swp1 iface swp1 auto swp2 iface swp2 auto bridge iface bridge </pre>



Node	NCLU Commands	/etc/network/interfaces Configuration
		<pre>bridge-ports vni-10 bridge-vids 10 bridge-vlan- aware yes auto vni-10 iface vni-10 bridge-access 10 mstpcl- bpduguard yes mstpcl- portbpdufilter yes vxlan-id 10 vxlan-local- tunnelip 10.0.0.12 vxlan- remoteip 10.0.0.11 vxlan- remoteip 10.0.0.13 vxlan- remoteip 10.0.0.14</pre>
leaf03	<pre>cumulus@leaf03:~\$ net add loopback lo ip address 10.0.0.13/32 cumulus@leaf03:~\$ net add vxlan vni-10 vxlan id 10 cumulus@leaf03:~\$ net add vxlan vni-10 vxlan local-tunnelip 10.0.0.13 cumulus@leaf03:~\$ net add vxlan vni-10 vxlan remoteip 10.0.0.11 cumulus@leaf03:~\$ net add vxlan vni-10 vxlan remoteip 10.0.0.12 cumulus@leaf03:~\$ net add vxlan vni-10 vxlan remoteip 10.0.0.14 cumulus@leaf03:~\$ net add vxlan vni-10 bridge access 10 cumulus@leaf03:~\$ net pending cumulus@leaf03:~\$ net commit</pre>	<pre># The loopback network interface auto lo iface lo inet loopback address 10.0.0.13/32 # The primary network interface auto eth0 iface eth0 inet dhcp auto swp1 iface swp1 auto swp2 iface swp2 auto bridge</pre>



Node	NCLU Commands	/etc/network/interfaces Configuration
		<pre>iface bridge bridge-ports vni-10 bridge-vids 10 bridge-vlan- aware yes auto vni-10 iface vni-10 bridge-access 10 mstpctl- bpduguard yes mstpctl- portbpdufilter yes vxlan-id 10 vxlan-local- tunnelip 10.0.0.13 vxlan- remoteip 10.0.0.11 vxlan- remoteip 10.0.0.12 vxlan- remoteip 10.0.0.14</pre>
leaf04	<pre>cumulus@leaf04:~\$ net add loopback lo ip address 10.0.0.14/32 cumulus@leaf04:~\$ net add vxlan vni-10 vxlan id 10 cumulus@leaf04:~\$ net add vxlan vni-10 vxlan local-tunnelip 10.0.0.14 cumulus@leaf04:~\$ net add vxlan vni-10 vxlan remoteip 10.0.0.11 cumulus@leaf04:~\$ net add vxlan vni-10 vxlan remoteip 10.0.0.12 cumulus@leaf04:~\$ net add vxlan vni-10 vxlan remoteip 10.0.0.13 cumulus@leaf04:~\$ net add vxlan vni-10 bridge access 10 cumulus@leaf04:~\$ net pending cumulus@leaf04:~\$ net commit</pre>	<pre># The loopback network interface auto lo iface lo inet loopback address 10.0.0.14/32 # The primary network interface auto eth0 iface eth0 inet dhcp auto swp1 iface swp1 auto swp2 iface swp2</pre>

Node	NCLU Commands	/etc/network/interfaces Configuration
		<pre> auto bridge iface bridge bridge-ports vni-10 bridge-vids 10 bridge-vlan- aware yes auto vni-10 iface vni-10 bridge-access 10 mstpctl- bpduguard yes mstpctl- portbpdufilter yes vxlan-id 10 vxlan-local- tunnelip 10.0.0.14 vxlan- remoteip 10.0.0.11 vxlan- remoteip 10.0.0.12 vxlan- remoteip 10.0.0.13 </pre>

Verifying the Configuration

Once you configure all the leaf switches, check for replication entries:

```

cumulus@leaf01:~$ sudo bridge fdb show | grep 00:00:00:00:00:00
00:00:00:00:00:00 dev vni-10 dst 10.0.0.14 self permanent
00:00:00:00:00:00 dev vni-10 dst 10.0.0.12 self permanent
00:00:00:00:00:00 dev vni-10 dst 10.0.0.13 self permanent

```



Layer Three

Routing

This chapter discusses routing on switches running Cumulus Linux.

Contents

This chapter covers ...

- Managing Static Routes (see page 478)
 - Static Routing via ip route (see page 479)
 - Applying a Route Map for Route Updates (see page 480)
- Supported Route Table Entries (see page 480)
 - Forwarding Table Profiles (see page 480)
 - TCAM Resource Profiles for Mellanox Switches (see page 481)
 - Number of Supported Route Entries, by Platform (see page 482)
- Caveats and Errata (see page 483)
 - Don't Delete Routes via Linux Shell (see page 483)
 - Adding IPv6 Default Route with src Address on eth0 Fails without Adding Delay (see page 483)
- Related Information (see page 484)

Managing Static Routes

You manage static routes using [NCLU](#) (see page 80) or the Cumulus Linux `ip route` command. The routes are added to the [Quagga](#) routing table, and are then updated into the kernel routing table as well.

To add a static route, run:

```
cumulus@switch:~$ net add routing route 203.0.113.0/24 198.51.100.2
cumulus@switch:~$ net pending
cumulus@switch:~$ net commit
```

These commands create the following configuration in the `/etc/quagga/Quagga.conf` file:

```
!
ip route 203.0.113.0/24 198.51.100.2
!
```

To delete a static route, run:

```
cumulus@switch:~$ net del routing route 203.0.113.0/24 198.51.100.2
```



```
cumulus@switch:~$ net pending  
cumulus@switch:~$ net commit
```

To view static routes, run:

```
cumulus@switch:~$ net show route static  
RIB entry for static  
=====  
Codes: K - kernel route, C - connected, S - static, R - RIP,  
      O - OSPF, I - IS-IS, B - BGP, P - PIM, T - Table,  
      > - selected route, * - FIB route  
S>* 203.0.113.0/24 [1/0] via 198.51.100.2, swp3
```

Static Routing via ip route

A static route can also be created by adding `post-up ip route add` command to a switch port configuration. For example:

```
cumulus@switch:~$ net add interface swp3 ip address 198.51.100.1/24  
cumulus@switch:~$ net add interface swp3 post-up routing route add  
203.0.113.0/24 via 198.51.100.2  
cumulus@switch:~$ net pending  
cumulus@switch:~$ net commit
```

These commands produce the following configuration in the `/etc/network/interfaces` file:

```
auto swp3  
iface swp3  
    address 198.51.100.1/24  
    post-up ip route add 203.0.113.0/24 via 198.51.100.2
```



If an IPv6 address is assigned to a DOWN interface, the associated route is still installed into the routing table. The type of IPv6 address doesn't matter: link local, site local and global all exhibit the same problem.

If the interface is bounced up and down, then the routes are no longer in the route table.

The `ip route` command allows manipulating the kernel routing table directly from the Linux shell. See `man ip(8)` for details. `quagga` monitors the kernel routing table changes and updates its own routing table accordingly.

To display the routing table:

```
cumulus@switch:~$ ip route show  
default via 10.0.1.2 dev eth0
```

```

10.0.1.0/24 dev eth0 proto kernel scope link src 10.0.1.52
192.0.2.0/24 dev swp1 proto kernel scope link src 192.0.2.12
192.0.2.10/24 via 192.0.2.1 dev swp1 proto zebra metric 20
192.0.2.20/24 proto zebra metric 20
    nexthop via 192.0.2.1 dev swp1 weight 1
    nexthop via 192.0.2.2 dev swp2 weight 1
192.0.2.30/24 via 192.0.2.1 dev swp1 proto zebra metric 20
192.0.2.40/24 dev swp2 proto kernel scope link src 192.0.2.42
192.0.2.50/24 via 192.0.2.2 dev swp2 proto zebra metric 20
192.0.2.60/24 via 192.0.2.2 dev swp2 proto zebra metric 20
192.0.2.70/24 proto zebra metric 30
    nexthop via 192.0.2.1 dev swp1 weight 1
    nexthop via 192.0.2.2 dev swp2 weight 1
198.51.100.0/24 dev swp3 proto kernel scope link src 198.51.100.1
198.51.100.10/24 dev swp4 proto kernel scope link src 198.51.100.11
198.51.100.20/24 dev br0 proto kernel scope link src 198.51.100.21

```

Applying a Route Map for Route Updates

To apply a [route map](#) to filter route updates from Zebra into the Linux kernel:

```
cumulus@switch:$ net add ip protocol static route-map <route-map-name>
```

Supported Route Table Entries

Cumulus Linux — via `switchd` — advertises the maximum number of route table entries that are supported on a given switch architecture, including:

- L3 IPv4 LPM (longest prefix match) entries, which have a mask that is less than /32
- L3 IPv6 LPM entries, which have a mask that is /64 or less
- L3 IPv6 LPM entries, which have a mask that is greater than /64
- L3 IPv4 neighbor (or host) entries, which are the next hops seen in `ip neighbor`
- L3 IPv6 neighbor entries, which are the next hops seen in `ip -6 neighbor`
- ECMP next hops, which are IP address entries in a router's routing table that specify the next closest /most optimal router in its routing path
- MAC addresses

In addition, switches on the Tomahawk, Trident II+ and Trident II platforms are configured to manage route table entries using Algorithm Longest Prefix Match (ALPM). In ALPM mode, the hardware can store significantly more route entries.

You can use `c1-resource-query` (see page 633) to determine the current table sizes on a given switch.

Forwarding Table Profiles

Mellanox Spectrum and some Broadcom ASICs provide the ability to configure the allocation of forwarding table resources and mechanisms. Cumulus Linux provides a number of generalized profiles for the platforms described below. These profiles work only with layer 2 and layer 3 unicast forwarding.



Cumulus Linux defines these profiles as *default*, *l2-heavy*, *v4-lpm-heavy* and *v6-lpm-heavy*. Choose the profile that best suits your network architecture and specify the profile name for the `forwarding_table.profile` variable in the `/etc/cumulus/datapath/traffic.conf` file.

```
cumulus@switch:~$ cat /etc/cumulus/datapath/traffic.conf | grep forwarding_table -B 4
# Manage shared forwarding table allocations
# Valid profiles -
# default, l2-heavy, v4-lpm-heavy, v6-lpm-heavy
#
forwarding_table.profile = default
```

After you specify a different profile, [restart switchd](#) (see page 173) for the change to take effect. You can see the forwarding table profile when you run `cl-resource-query`.



Broadcom ASICs other than Tomahawk and Trident II/Trident II+ support only the *default* profile.



For Broadcom ASICs, the maximum number of IP multicast entries is 8k.

TCAM Resource Profiles for Mellanox Switches

The Mellanox Spectrum ASIC provides the ability to configure the TCAM resource allocation, which is shared between IP multicast forwarding entries and ACL tables. Cumulus Linux provides a number of general profiles for this platform: *default*, *ipmc-heavy* and *acl-heavy*. Choose the profile that best suits your network architecture and specify that profile name in the `tcam_resource.profile` variable in the `/usr/lib/python2.7/dist-packages/cumulus/__chip_config/mlx/datapath.conf` file.

```
cumulus@switch:~$ cat /usr/lib/python2.7/dist-packages/cumulus/__chip_config/mlx/datapath.conf | grep -B3 "tcam_resource"
#TCAM resource forwarding profile
# Valid profiles -
# default, ipmc-heavy, acl-heavy
tcam_resource.profile = default
```

After you specify a different profile, [restart switchd](#) for the change to take effect.

When [nonatomic updates](#) (see page 131) are enabled (that is, the `acl.non_atomic_update_mode` is set to *TRUE* in `/etc/cumulus/switchd.conf` file), the maximum number of mroute and ACL entries for each profile are as follows:

Profile	Mroute Entries	ACL Entries
default	450	1500 (IPv6) or 3500 (IPv4)

Profile	Mroute Entries	ACL Entries
ipmc-heavy	6000	160 (IPv6) or 400 (IPv4)
acl-heavy	450	2500 (IPv6) or 6000 (IPv4)

When [nonatomic updates](#) (see page 131) are disabled (that is, the `acl.non_atomic_update_mode` is set to *FALSE* in `/etc/cumulus/switchd.conf` file), the maximum number of mroute and ACL entries for each profile are as follows:

Profile	Mroute Entries	ACL Entries
default	450	750 (IPv6) or 1750 (IPv4)
ipmc-heavy	6000	80 (IPv6) or 200 (IPv4)
acl-heavy	450	2000 (IPv6) or 3000 (IPv4)

Number of Supported Route Entries, by Platform

The following tables list the number of MAC addresses, layer 3 neighbors and LPM routes validated for each forwarding table profile for the various supported platforms. If you are not specifying any profiles as described above, the *default* values are the ones that the switch will use.



The values in the following tables reflect results from our testing on the different platforms we support, and may differ from published manufacturers' specifications provided about these chipsets.

Mellanox Spectrum Switches

Profile	MAC Addresses	L3 Neighbors	Longest Prefix Match (LPM)
default	40k	32k (IPv4) and 16k (IPv6)	64k (IPv4) or 28k (IPv6-long)
l2-heavy	88k	48k (IPv4) and 40k (IPv6)	8k (IPv4) and 8k (IPv6-long)
l2-heavy-1	180K	8k (IPv4) and 8k (IPv6)	8k (IPv4) and 8k (IPv6-long)
v4-lpm-heavy	8k	8k (IPv4) and 16k (IPv6)	80k (IPv4) and 16k (IPv6-long)
v4-lpm-heavy-1	8k	8k (IPv4) and 2k (IPv6)	176k (IPv4) and 2k (IPv6-long)
v6-lpm-heavy	40k	8k (IPv4) and 40k (IPv6)	8k (IPv4) and 64k (IPv6-long)



Broadcom Tomahawk Switches

Profile	MAC Addresses	L3 Neighbors	Longest Prefix Match (LPM)
default	40k	40k	64k (IPv4) or 8k (IPv6-long)
I2-heavy	72k	72k	8k (IPv4) or 2k (IPv6-long)
v4-lpm-heavy, v6-lpm-heavy	8k	8k	128k (IPv4) or 20k (IPv6-long)

Broadcom Trident II/Trident II+ Switches

Profile	MAC Addresses	L3 Neighbors	Longest Prefix Match (LPM)
default	32k	16k	128k (IPv4) or 20k (IPv6-long)
I2-heavy	160k	96k	8k (IPv4) or 2k (IPv6-long)
v4-lpm-heavy, v6-lpm-heavy	32k	16k	128k (IPv4) or 20k (IPv6-long)

Caveats and Errata

Don't Delete Routes via Linux Shell

Static routes added via Quagga can be deleted via Linux shell. This operation, while possible, should be avoided. Routes added by Quagga should only be deleted by Quagga, otherwise Quagga might not be able to clean up all its internal state completely and incorrect routing can occur as a result.

Adding IPv6 Default Route with src Address on eth0 Fails without Adding Delay

Attempting to install an IPv6 default route on eth0 with a source address fails at reboot or when running `ifup` on eth0.

The first execution of `ifup -dv` returns this warning and does not install the route:

```
cumulus@switch:~$ sudo ifup -dv eth0
warning: eth0: post-up cmd '/sbin/ip route add default via 2001:620:5ca1:160::1 /src 2001:620:5ca1:160::45 dev eth0' failed (RTNETLINK answers: Invalid argument)<<<<<<
```

Running `ifup` a second time on eth0 successfully installs the route.

There are two ways you can work around this issue.

- Add a sleep 2 to the eth0 interface in /etc/network/interfaces:

```
cumulus@switch:~$ net add interface eth0 ipv6 address 2001:620:5cal:160::45/64 post-up /bin/sleep 2s
cumulus@switch:~$ net add interface eth0 post-up /sbin/ip route add default via 2001:620:5cal:160::1 src 2001:620:5call:160::45 dev eth0
```

- Exclude the `src` parameter to the `ip route add` that causes the need for the delay. If the `src` parameter is removed, the route is added correctly.

```
cumulus@switch:~$ net add interface eth0 post-up /sbin/ip route add default via 2001:620:5cal:160::1 dev eth0
```

```
cumulus@switch:~$ ifdown eth0
Stopping NTP server: ntpd.
Starting NTP server: ntpd.
cumulus@switch:~$ ip -6 r s
cumulus@switch:~$ ifup eth0
Stopping NTP server: ntpd.
Starting NTP server: ntpd.
cumulus@switch:~$ ip -6 r s
2001:620:5cal:160::/64 dev eth0 proto kernel metric 256
fe80::/64 dev eth0 proto kernel metric 256
default via 2001:620:5cal:160::1 dev eth0 metric 1024
```

Related Information

- [Linux IP - ip route command](#)
- [Quagga docs - static route commands](#)

Introduction to Routing Protocols

This chapter discusses the various routing protocols, and how to configure them.

Contents

This chapter covers ...

- [Defining Routing Protocols \(see page 485\)](#)
- [Configuring Routing Protocols \(see page 485\)](#)
- [Protocol Tuning \(see page 485\)](#)

Defining Routing Protocols

A *routing protocol* dynamically computes reachability between various end points. This enables communication to work around link and node failures, and additions and withdrawals of various addresses.

IP routing protocols are typically distributed; that is, an instance of the routing protocol runs on each of the routers in a network.



Cumulus Linux does **not** support running multiple instances of the same protocol on a router.

Distributed routing protocols compute reachability between end points by disseminating relevant information and running a routing algorithm on this information to determine the routes to each end station. To scale the amount of information that needs to be exchanged, routes are computed on address prefixes rather than on every end point address.

Configuring Routing Protocols

A routing protocol needs to know three pieces of information, at a minimum:

- Who am I (my identity)
- To whom to disseminate information
- What to disseminate

Most routing protocols use the concept of a router ID to identify a node. Different routing protocols answer the last two questions differently.

The way they answer these questions affects the network design and thereby configuration. For example, in a link-state protocol such as OSPF (see [Open Shortest Path First \(OSPF\) Protocol \(see page 500\)](#)) or IS-IS, complete local information (links and attached address prefixes) about a node is disseminated to every other node in the network. Since the state that a node has to keep grows rapidly in such a case, link-state protocols typically limit the number of nodes that communicate this way. They allow for bigger networks to be built by breaking up a network into a set of smaller subnetworks (which are called areas or levels), and by advertising summarized information about an area to other areas.

Besides the two critical pieces of information mentioned above, protocols have other parameters that can be configured. These are usually specific to each protocol.

Protocol Tuning

Most protocols provide certain tunable parameters that are specific to convergence during changes.

Wikipedia defines [convergence](#) as the “state of a set of routers that have the same topological information about the network in which they operate”. It is imperative that the routers in a network have the same topological state for the proper functioning of a network. Without this, traffic can be blackholed, and thus not reach its destination. It is normal for different routers to have differing topological states during changes, but this difference should vanish as the routers exchange information about the change and recompute the forwarding paths. Different protocols converge at different speeds in the presence of changes.

A key factor that governs how quickly a routing protocol converges is the time it takes to detect the change. For example, how quickly can a routing protocol be expected to act when there is a link failure. Routing protocols classify changes into two kinds: hard changes such as link failures, and soft changes such as a peer dying silently. They're classified differently because protocols provide different mechanisms for dealing with these failures.

It is important to configure the protocols to be notified immediately on link changes. This is also true when a node goes down, causing all of its links to go down.

Even if a link doesn't fail, a routing peer can crash. This causes that router to usually delete the routes it has computed or worse, it makes that router impervious to changes in the network, causing it to go out of sync with the other routers in the network because it no longer shares the same topological information as its peers.

The most common way to detect a protocol peer dying is to detect the absence of a heartbeat. All routing protocols send a heartbeat (or "hello") packet periodically. When a node does not see a consecutive set of these hello packets from a peer, it declares its peer dead and informs other routers in the network about this. The period of each heartbeat and the number of heartbeats that need to be missed before a peer is declared dead are two popular configurable parameters.

If you configure these timers very low, the network can quickly descend into instability under stressful conditions when a router is not able to keep sending the heartbeats quickly as it is busy computing routing state; or the traffic is so much that the hellos get lost. Alternately, configuring this timer to very high values also causes blackholing of communication because it takes much longer to detect peer failures. Usually, the default values initialized within each protocol are good enough for most networks. Cumulus Networks recommends you do not adjust these settings.

Network Topology

In computer networks, *topology* refers to the structure of interconnecting various nodes. Some commonly used topologies in networks are star, hub and spoke, leaf and spine, and broadcast.

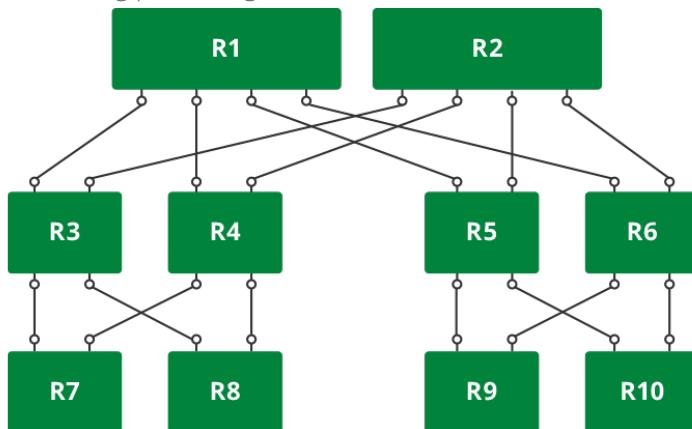
Contents

This chapter covers ...

- Clos Topologies (see page 486)
- Over-Subscribed and Non-Blocking Configurations (see page 487)
- Containing the Failure Domain (see page 487)
- Load Balancing (see page 488)

Clos Topologies

In the vast majority of modern data centers, [Clos or fat tree topology](#) is very popular. This topology is shown in the figure below. It is also commonly referred to as leaf-spine topology. We shall use this topology throughout the routing protocol guide.





This topology allows the building of networks of varying size using nodes of different port counts and/or by increasing the tiers. The picture above is a three-tiered Clos network. We number the tiers from the bottom to the top. Thus, in the picture, the lowermost layer is called tier 1 and the topmost tier is called tier 3.

The number of end stations (such as servers) that can be attached to such a network is determined by a very simple mathematical formula.

In a 2-tier network, if each node is made up of m ports, then the total number of end stations that can be connected is $m^2/2$. In more general terms, if tier-1 nodes are m -port nodes and tier-2 nodes are n -port nodes, then the total number of end stations that can be connected are $(m*n)/2$. In a three tier network, where tier-3 nodes are o -port nodes, the total number of end stations that can be connected are $(m*n*o)/2^{(number\ of\ tiers-1)}$.

Let's consider some practical examples. In many data centers, it is typical to connect 40 servers to a top-of-rack (ToR) switch. The ToRs are all connected via a set of spine switches. If a ToR switch has 64 ports, then after hooking up 40 ports to the servers, the remaining 24 ports can be hooked up to 24 spine switches of the same link speed or to a smaller number of higher link speed switches. For example, if the servers are all hooked up as 10GE links, then the ToRs can connect to the spine switches via 40G links. So, instead of connecting to 24 spine switches with 10G links, the ToRs can connect to 6 spine switches with each link being 40G. If the spine switches are also 64-port switches, then the total number of end stations that can be connected is 2560 ($40*64$) stations.

In a three tier network of 64-port switches, the total number of servers that can be connected are $(40*64*64)/2^{(3-1)} = 40960$. As you can see, this kind of topology can serve quite a large network with three tiers.

Over-Subscribed and Non-Blocking Configurations

In the above example, the network is *over-subscribed*; that is, 400G of bandwidth from end stations (40 servers * 10GE links) is serviced by only 240G of inter-rack bandwidth. The over-subscription ratio is 0.6 ($240/400$).

This can lead to congestion in the network and hot spots. Instead, if network operators connected 32 servers per rack, then 32 ports are left to be connected to spine switches. Now, the network is said to be *rearrangably non-blocking*. Now any server in a rack can talk to any other server in any other rack without necessarily blocking traffic between other servers.

In such a network, the total number of servers that can be connected are $(64*64)/2 = 2048$. Similarly, a three-tier version of the same can serve up to $(64*64*64)/4 = 65536$ servers.

Containing the Failure Domain

Traditional data centers were built using just two spine switches. This means that if one of those switches fails, the network bandwidth is cut in half, thereby greatly increasing network congestion and adversely affecting many applications. To avoid this, vendors typically try and make the spine switches resilient to failures by providing such features as dual control line cards and attempting to make the software highly available. However, as Douglas Adams famously noted, ">>>". In many cases, HA is among the top two or three causes of software failure (and thereby switch failure).

To support a fairly large network with just two spine switches also means that these switches have a large port count. This can make the switches quite expensive.

If the number of spine switches were to be merely doubled, the effect of a single switch failure is halved. With 8 spine switches, the effect of a single switch failure only causes a 12% reduction in available bandwidth.

So, in modern data centers, people build networks with anywhere from 4 to 32 spine switches.

Load Balancing

In a Clos network, traffic is load balanced across the multiple links using equal cost multi-pathing (ECMP).

Routing algorithms compute shortest paths between two end stations where shortest is typically the lowest path cost. Each link is assigned a metric or cost. By default, a link's cost is a function of the link speed. The higher the link speed, the lower its cost. A 10G link has a higher cost than a 40G or 100G link, but a lower cost than a 1G link. Thus, the link cost is a measure of its traffic carrying capacity.

In the modern data center, the links between tiers of the network are homogeneous; that is, they have the same characteristics (same speed and therefore link cost) as the other links. As a result, the first hop router can pick any of the spine switches to forward a packet to its destination (assuming that there is no link failure between the spine and the destination switch). Most routing protocols recognize that there are multiple equal-cost paths to a destination and enable any of them to be selected for a given traffic flow.

Quagga Overview

Cumulus Linux uses Cumulus Quagga, a fork of the open source routing software suite, to provide the routing protocols for dynamic routing. Cumulus Quagga runs the latest Quagga version, 0.99.23.1+cl3u2. Quagga is a fork of the [GNU Zebra](#) project.

Quagga provides many routing protocols, of which Cumulus Linux supports the following:

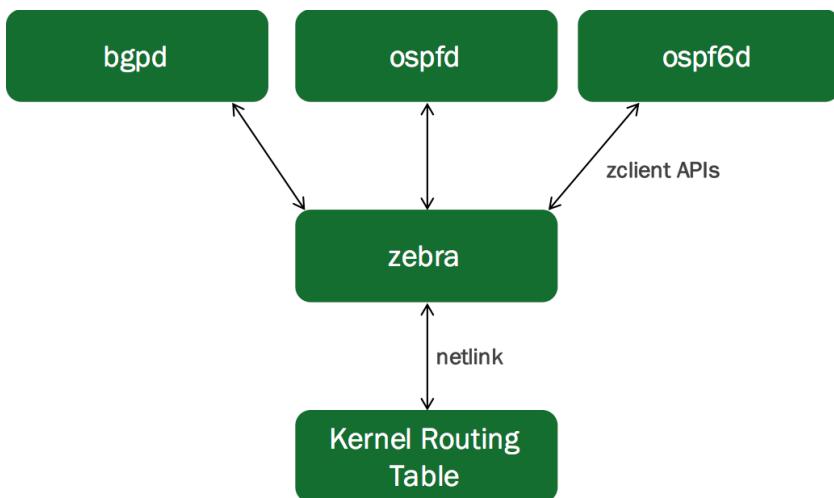
- Open Shortest Path First ([v2 \(see page 500\)](#) and [v3 \(see page 514\)](#))
- Border Gateway Protocol ([see page 516](#))

Contents

This chapter covers ...

- [Architecture \(see page 488\)](#)
- [About zebra \(see page 489\)](#)
- [Related Information \(see page 489\)](#)

Architecture





As shown in the figure above, the Quagga routing suite consists of various protocol-specific daemons and a protocol-independent daemon called `zebra`. Each of the protocol-specific daemons are responsible for running the relevant protocol and building the routing table based on the information exchanged.

It is not uncommon to have more than one protocol daemon running at the same time. For example, at the edge of an enterprise, protocols internal to an enterprise (called IGP for Interior Gateway Protocol) such as [OSPF](#) (see page 500) or RIP run alongside the protocols that connect an enterprise to the rest of the world (called EGP or Exterior Gateway Protocol) such as [BGP](#) (see page 516).

About `zebra`

`zebra` is the daemon that resolves the routes provided by multiple protocols (including static routes specified by the user) and programs these routes in the Linux kernel via `netlink` (in Linux). `zebra` does more than this, of course. The [Quagga documentation](#) defines `zebra` as the IP routing manager for `quagga` that "provides kernel routing table updates, interface lookups, and redistribution of routes between different routing protocols."

Related Information

- www.quagga.net/
- packages.debian.org/quagga

Configuring Quagga

This section provides an overview of configuring Cumulus Quagga, the Cumulus Networks-enhanced version of the Quagga routing software package that provides a suite of routing protocols so you can configure routing on your switch.

Contents

This chapter covers ...

- Configuring Cumulus Quagga (see page 489)
 - Enabling and Starting Cumulus Quagga (see page 490)
 - Understanding Integrated Configurations (see page 490)
 - Restoring the Default Configuration (see page 491)
- Interface IP Addresses and VRFs (see page 492)
- Using the Quagga vtysh Modal CLI (see page 492)
- Reloading the Quagga Configuration (see page 497)
 - Debugging (see page 498)
- Related Information (see page 498)

Configuring Cumulus Quagga

Cumulus Quagga does not start by default in Cumulus Linux. Before you run Cumulus Quagga, make sure all you have enabled relevant daemons that you intend to use — `zebra`, `bgpd`, `ospfd`, `ospf6d` or `pimd` — in the `/etc/quagga/daemons` file.



! Cumulus Networks has not tested RIP, RIPv6, IS-IS and Babel.

The `zebra` daemon must always be enabled. The others you can enable according to how you plan to route your network — using [BGP \(see page 516\)](#) for example, instead of [OSPF \(see page 500\)](#).

Before you start Cumulus Quagga, you need to enable the corresponding daemons. Edit the `/etc/quagga/daemons` file and set to `yes` each daemon you are enabling. For example, to enable BGP, set both `zebra` and `bgpd` to `yes`:

```
zebra=yes (* this one is mandatory to bring the others up)
bgpd=yes
ospf6d=no
ospf6d=no
ripd=no
ripngd=no
isisd=no
babeld=no
pimd=no
```

Enabling and Starting Cumulus Quagga

Once you enable the appropriate daemons, then you need to enable and start the Cumulus Quagga service.

```
cumulus@switch:~$ sudo systemctl enable quagga.service
cumulus@switch:~$ sudo systemctl start quagga.service
```

✓ All the routing protocol daemons (`bgpd`, `ospf6d`, `ospf6d`, `ripd`, `ripngd`, `isisd` and `pimd`) are dependent on `zebra`. When you start `quagga`, `systemd` determines whether `zebra` is running; if `zebra` is not running, it starts `zebra`, then starts the dependent service, such as `bgpd`.

In general, if you restart a service, its dependent services also get restarted. For example, running `systemctl restart quagga.service` restarts any of the routing protocol daemons that are enabled and running.

For more information on the `systemctl` command and changing the state of daemons, read [Managing Application Daemons \(see page 163\)](#).

Understanding Integrated Configurations

By default in Cumulus Linux, Quagga saves the configuration of all daemons in a single integrated configuration file, `Quagga.conf`.

You can disable this mode by running the following command in the `vtysh` Quagga CLI (see page 492):

```
cumulus@switch:~$ sudo vtysh
```

```
switch# configure terminal  
switch(config)# no service integrated-vtysh-config
```

To enable the integrated configuration file mode again, run:

```
switch(config)# service integrated-vtysh-config
```

If you disable the integrated configuration mode, Quagga saves each daemon-specific configuration file in a separate file. At a minimum for a daemon to start, that daemon must be enabled and its daemon-specific configuration file must be present, even if that file is empty.

You save the current configuration by running:

```
switch# write mem  
Building Configuration...  
Integrated configuration saved to /etc/quagga/Quagga.conf  
[OK]  
switch# exit  
cumulus@switch:~$
```



You can use `write file` instead of `write mem`.

When the integrated configuration mode disabled, the output looks like this:

```
switch# write mem  
Building Configuration...  
Configuration saved to /etc/quagga/zebra.conf  
Configuration saved to /etc/quagga/bgpd.conf  
[OK]
```

Restoring the Default Configuration

If you need to restore the Quagga configuration to the default running configuration, you need to delete the `Quagga.conf` file and restart the `quagga` service. You should back up `Quagga.conf` (or any configuration files you may remove, see the note below) before proceeding.

1. Confirm `service integrated-vtysh-config` is enabled:

```
cumulus@switch:~$ net show configuration | grep integrated  
service integrated-vtysh-config
```

2. Remove `/etc/quagga/Quagga.conf`:



```
cumulus@switch:~$ sudo rm /etc/quagga/Quagga.conf
```

3. Restart Quagga:

```
cumulus@switch:~$ sudo systemctl restart quagga.service
```



If for some reason you disabled `service integrated-vtysh-config`, then you should remove all the configuration files (such as `zebra.conf` or `ospf6d.conf`) instead of `Quagga.conf` in step 2 above.

Interface IP Addresses and VRFs

Quagga inherits the IP addresses and any associated routing tables for the network interfaces from the `/etc/network/interfaces` file. This is the recommended way to define the addresses; do **not** create interfaces using Quagga. For more information, see [Configuring IP Addresses \(see page 193\)](#) and [Virtual Routing and Forwarding - VRF \(see page 574\)](#).

Using the Quagga vtysh Modal CLI

Quagga provides a CLI – `vtysh` – for configuring and displaying the state of the protocols. It is invoked by running:

```
cumulus@switch:~$ sudo vtysh
Hello, this is Quagga (version 0.99.23.1+cl3u2).
Copyright 1996-2005 Kunihiro Ishiguro, et al.

switch#
```

`vtysh` provides a Cisco-like modal CLI, and many of the commands are similar to Cisco IOS commands. By modal CLI, we mean that there are different modes to the CLI, and certain commands are only available within a specific mode. Configuration is available with the `configure terminal` command, which is invoked thus:

```
switch# configure terminal
switch(config)#
```

The prompt displays the mode the CLI is in. For example, when the interface-specific commands are invoked, the prompt changes to:

```
switch(config)# interface swp1
switch(config-if)#
```



When the routing protocol specific commands are invoked, the prompt changes to:

```
switch(config)# router ospf  
switch(config-router)#{/pre}
```

At any level, "?" displays the list of available top-level commands at that level:

```
switch(config-if)# ?  
bandwidth      Set bandwidth informational parameter  
description    Interface specific description  
end           End current mode and change to enable mode  
exit          Exit current mode and down to previous mode  
ip            IP Information  
ipv6          IPv6 Information  
isis          IS-IS commands  
link-detect   Enable link detection on interface  
list          Print command list  
mpls-te       MPLS-TE specific commands  
multicast     Set multicast flag to interface  
no             Negate a command or set its defaults  
ptm-enable    Enable neighbor check with specified topology  
quit          Exit current mode and down to previous mode  
shutdown      Shutdown the selected interface
```

?-based completion is also available to see the parameters that a command takes:

```
switch(config-if)# bandwidth ?  
<1-10000000> Bandwidth in kilobits  
switch(config-if)# ip ?  
address      Set the IP address of an interface  
irdp        Alter ICMP Router discovery preference this interface  
ospf        OSPF interface commands  
rip         Routing Information Protocol  
router      IP router interface commands
```

Displaying state can be done at any level, including the top level. For example, to see the routing table as seen by zebra, you use:

```
switch# show ip route  
Codes: K - kernel route, C - connected, S - static, R - RIP,  
      O - OSPF, I - IS-IS, B - BGP, T - Table,  
      > - selected route, * - FIB route  
B>* 0.0.0.0/0 [20/0] via fe80::4638:39ff:fe00:c, swp29, 00:11:57  
    *                  via fe80::4638:39ff:fe00:52, swp30, 00:11:57  
B>* 10.0.0.1/32 [20/0] via fe80::4638:39ff:fe00:c, swp29, 00:11:57  
    *                  via fe80::4638:39ff:fe00:52, swp30, 00:11:57  
B>* 10.0.0.11/32 [20/0] via fe80::4638:39ff:fe00:5b, swp1, 00:11:57
```

```
B>* 10.0.0.12/32 [20/0] via fe80::4638:39ff:fe00:2e, swp2, 00:11:58
B>* 10.0.0.13/32 [20/0] via fe80::4638:39ff:fe00:57, swp3, 00:11:59
B>* 10.0.0.14/32 [20/0] via fe80::4638:39ff:fe00:43, swp4, 00:11:59
C>* 10.0.0.21/32 is directly connected, lo
B>* 10.0.0.51/32 [20/0] via fe80::4638:39ff:fe00:c, swp29, 00:11:57
  *
    via fe80::4638:39ff:fe00:52, swp30, 00:11:57
B>* 172.16.1.0/24 [20/0] via fe80::4638:39ff:fe00:5b, swp1, 00:11:57
  *
    via fe80::4638:39ff:fe00:2e, swp2, 00:11:57
B>* 172.16.3.0/24 [20/0] via fe80::4638:39ff:fe00:57, swp3, 00:11:59
  *
    via fe80::4638:39ff:fe00:43, swp4, 00:11:59
```

To run the same command at a config level, you prepend do to it:

```
switch(config-router)# do show ip route
Codes: K - kernel route, C - connected, S - static, R - RIP,
       O - OSPF, I - IS-IS, B - BGP, T - Table,
       > - selected route, * - FIB route
B>* 0.0.0.0/0 [20/0] via fe80::4638:39ff:fe00:c, swp29, 00:05:17
  *
    via fe80::4638:39ff:fe00:52, swp30, 00:05:17
B>* 10.0.0.1/32 [20/0] via fe80::4638:39ff:fe00:c, swp29, 00:05:17
  *
    via fe80::4638:39ff:fe00:52, swp30, 00:05:17
B>* 10.0.0.11/32 [20/0] via fe80::4638:39ff:fe00:5b, swp1, 00:05:17
B>* 10.0.0.12/32 [20/0] via fe80::4638:39ff:fe00:2e, swp2, 00:05:18
B>* 10.0.0.13/32 [20/0] via fe80::4638:39ff:fe00:57, swp3, 00:05:18
B>* 10.0.0.14/32 [20/0] via fe80::4638:39ff:fe00:43, swp4, 00:05:18
C>* 10.0.0.21/32 is directly connected, lo
B>* 10.0.0.51/32 [20/0] via fe80::4638:39ff:fe00:c, swp29, 00:05:17
  *
    via fe80::4638:39ff:fe00:52, swp30, 00:05:17
B>* 172.16.1.0/24 [20/0] via fe80::4638:39ff:fe00:5b, swp1, 00:05:17
  *
    via fe80::4638:39ff:fe00:2e, swp2, 00:05:17
B>* 172.16.3.0/24 [20/0] via fe80::4638:39ff:fe00:57, swp3, 00:05:18
  *
    via fe80::4638:39ff:fe00:43, swp4, 00:05:18
```

Running single commands with vtysh is possible using the -c option of vtysh:

```
cumulus@switch:~$ sudo vtysh -c 'sh ip route'
Codes: K - kernel route, C - connected, S - static, R - RIP,
       O - OSPF, I - IS-IS, B - BGP, A - Babel,
       > - selected route, * - FIB route

K>* 0.0.0.0/0 via 192.168.0.2, eth0
C>* 192.0.2.11/24 is directly connected, swp1
C>* 192.0.2.12/24 is directly connected, swp2
B>* 203.0.113.30/24 [200/0] via 192.0.2.2, swp1, 11:05:10
B>* 203.0.113.31/24 [200/0] via 192.0.2.2, swp1, 11:05:10
B>* 203.0.113.32/24 [200/0] via 192.0.2.2, swp1, 11:05:10
C>* 127.0.0.0/8 is directly connected, lo
C>* 192.168.0.0/24 is directly connected, eth0
```



Running a command multiple levels down is done thus:

```
cumulus@switch:~$ sudo vtysh -c 'configure terminal' -c 'router ospf'  
-c 'area 0.0.0.1 range 10.10.10.0/24'
```

Notice that the commands also take a partial command name (for example, `sh ip route` above) as long as the partial command name is not aliased:

```
cumulus@switch:~$ sudo vtysh -c 'sh ip r'  
% Ambiguous command.
```

A command or feature can be disabled in Quagga by prepending the command with `no`. For example:

```
cumulus@switch:~$ sudo vtysh  
switch# configure terminal  
switch(config)# router ospf  
switch(config-router)# no area 0.0.0.1 range 10.10.10.0/24  
switch(config-router)# exit  
switch(config)# exit  
switch# write mem  
switch# exit  
cumulus@switch:~$
```

The current state of the configuration can be viewed using the `show running-config` command:

```
switch# show running-config  
Building configuration...  
  
Current configuration:  
!  
username cumulus nopassword  
!  
service integrated-vtysh-config  
!  
vrf mgmt  
!  
interface lo  
 link-detect  
!  
interface swp1  
 ipv6 nd ra-interval 10  
 link-detect  
!  
interface swp2  
 ipv6 nd ra-interval 10  
 link-detect
```



```
!
interface swp3
    ipv6 nd ra-interval 10
    link-detect
!
interface swp4
    ipv6 nd ra-interval 10
    link-detect
!
interface swp29
    ipv6 nd ra-interval 10
    link-detect
!
interface swp30
    ipv6 nd ra-interval 10
    link-detect
!
interface swp31
    link-detect
!
interface swp32
    link-detect
!
interface vagrant
    link-detect
!
interface eth0 vrf mgmt
    ipv6 nd suppress-ra
    link-detect
!
interface mgmt vrf mgmt
    link-detect
!
router bgp 65020
    bgp router-id 10.0.0.21
    bgp bestpath as-path multipath-relax
    bgp bestpath compare-routerid
    neighbor fabric peer-group
    neighbor fabric remote-as external
    neighbor fabric description Internal Fabric Network
    neighbor fabric capability extended-nexthop
    neighbor swp1 interface peer-group fabric
    neighbor swp2 interface peer-group fabric
    neighbor swp3 interface peer-group fabric
    neighbor swp4 interface peer-group fabric
    neighbor swp29 interface peer-group fabric
    neighbor swp30 interface peer-group fabric
!
address-family ipv4 unicast
    network 10.0.0.21/32
    neighbor fabric activate
    neighbor fabric prefix-list dc-spine in
```

```

neighbor fabric prefix-list dc-spine out
exit-address-family
!
ip prefix-list dc-spine seq 10 permit 0.0.0.0/0
ip prefix-list dc-spine seq 20 permit 10.0.0.0/24 le 32
ip prefix-list dc-spine seq 30 permit 172.16.1.0/24
ip prefix-list dc-spine seq 40 permit 172.16.2.0/24
ip prefix-list dc-spine seq 50 permit 172.16.3.0/24
ip prefix-list dc-spine seq 60 permit 172.16.4.0/24
ip prefix-list dc-spine seq 500 deny any
!
ip forwarding
ipv6 forwarding
!
line vty
!
end

```



If you attempt to configure a routing protocol that has not been started, `vtysh` silently ignores those commands.

Alternately, if you do not want to use a modal CLI to configure Quagga, you can use a suite of [Cumulus Linux-specific commands](#) (see page 498) instead.

Reloading the Quagga Configuration

If you make a change to your routing configuration, you need to reload Quagga so your changes take place. `Quagga reload` enables you to apply only the modifications you make to your Quagga configuration, synchronizing its running state with the configuration in `/etc/quagga/Quagga.conf`. This is useful for optimizing automation of Quagga in your environment or to apply changes made at runtime.



Quagga reload only applies to an integrated service configuration, where your Quagga configuration is stored in a single `Quagga.conf` file instead of one configuration file per Quagga daemon (like `zebra` or `bgpd`).

To reload your Quagga configuration after you've modified `/etc/quagga/Quagga.conf`, run:

```
cumulus@switch:~$ sudo systemctl reload quagga.service
```

Examine the running configuration and verify that it matches the config in `/etc/quagga/Quagga.conf`:

```
cumulus@switch:~$ net show configuration
```



Debugging

If the running configuration is not what you expected, please [submit a support request](#) and supply the following information:

- The current running configuration (run `net show configuration` and output the contents to a file)
- The contents of `/etc/quagga/Quagga.conf`
- The contents of `/var/log/quagga/quagga-reload.log`

Related Information

- www.nongnu.org/quagga/docs/docs-info.html#BGP
- www.nongnu.org/quagga/docs/docs-info.html#IPv6-Support
- www.nongnu.org/quagga/docs/docs-info.html#Zebra

Comparing NCLU and vtysh Commands

Using **NCLU** (see page 80) is the primary way to [configure routing](#) (see page 489) in Cumulus Linux. However, an alternative exists in the the **vtysh** modal CLI. The available commands are as follows:

Comparing vtysh and NCLU Commands

The following table compares the various Quagga commands with their Cumulus Linux NCLU counterparts.

Action	NCLU Commands	Quagga Commands
Display the routing table	<pre>cumulus@switch: ~\$ net show route</pre>	<pre>switch# show ip route</pre>
Create a new neighbor	<pre>cumulus@switch: ~\$ net add bgp autonomous- system 65002 cumulus@switch: ~\$ net add bgp neighbor 14.0.0.2 2</pre>	<pre>switch(config) # router bgp 65 002 switch(config- router)# neighbor 14.0.0 .22</pre>
Redistribute routing information from static route entries into RIP tables		

Action	NCLU Commands	Quagga Commands
	<pre>cumulus@switch: ~\$ net add bgp redistribute static</pre>	<pre>switch(config) # router bgp 65 002 switch(config-router)# redistribute static</pre>
Define a static route (see page 478)	<pre>cumulus@switch: ~\$ net add routing route 155 .1.2.20/24 bridge 45</pre>	<pre>switch(config) # ip route 155. 1.2.20/24 bridge 45</pre>
Configure an IPv6 address	<pre>cumulus@switch: ~\$ net add interface swp3 ipv6 address 3002:2123 :1234:1abc::21/64</pre>	<pre>switch(config) # int swp3 switch(config-if)# ipv6 address 3002:21 23:1234:1abc::2 1/64</pre>
Enable topology checking (PTM (see page 257))	<pre>cumulus@switch: ~\$ net add routing ptm- enable</pre>	<pre>switch(config) # ptm-enable</pre>
Configure MTU (see page 210) in IPv6 network discovery for an interface	<pre>cumulus@switch: ~\$ sudo cl-ra interface swp3 set mtu 9000</pre>	<pre>switch(config) # int swp3 switch(config-if)# ipv6 nd mtu 9000</pre>
Set the OSPF interface priority		

Action	NCLU Commands	Quagga Commands
	<pre>cumulus@switch: ~\$ net add interf ace swp3 ospf6 priority 120</pre>	<pre>switch(config) # int swp3 switch(config-i f)# ip ospf6 priority 120</pre>
Configure timing for OSPF SPF calculations	<pre>cumulus@switch: ~\$ net add ospf6 timers throttle spf 40 50 60</pre>	<pre>switch(config) # router ospf6 switch(config- ospf6)# timer throttle spf 40 50 60</pre>
Configure the OSPF Hello packet interval in number of seconds for an interface	<pre>cumulus@switch: ~\$ net add interf ace swp4 ospf6 hello-interval 60</pre>	<pre>switch(config) # int swp4 switch(config-i f)# ipv6 ospf6 hello- interval 60</pre>
Display BGP (see page 516) information	<pre>cumulus@switch: ~\$ net show bgp summary</pre>	<pre>switch# show ip bgp summary</pre>

Open Shortest Path First - OSPF - Protocol

OSPFv2 is a [link-state routing protocol](#) for IPv4. OSPF maintains the view of the network topology conceptually as a directed graph. Each router represents a vertex in the graph. Each link between neighboring routers represents a unidirectional edge. Each link has an associated weight (called cost) that is either automatically derived from its bandwidth or administratively assigned. Using the weighted topology graph, each router computes a shortest path tree (SPT) with itself as the root, and applies the results to build its forwarding table. The computation is generally referred to as *SPF computation* and the resultant tree as the *SPF tree*.

An LSA (*link-state advertisement*) is the fundamental quantum of information that OSPF routers exchange with each other. It seeds the graph building process on the node and triggers SPF computation. LSAs originated by a node are distributed to all the other nodes in the network through a mechanism called



flooding. Flooding is done hop-by-hop. OSPF ensures reliability by using link state acknowledgement packets. The set of LSAs in a router's memory is termed *link-state database* (LSDB), a representation of the network graph. Thus, OSPF ensures a consistent view of LSDB on each node in the network in a distributed fashion (eventual consistency model); this is key to the protocol's correctness.

Contents

This chapter covers ...

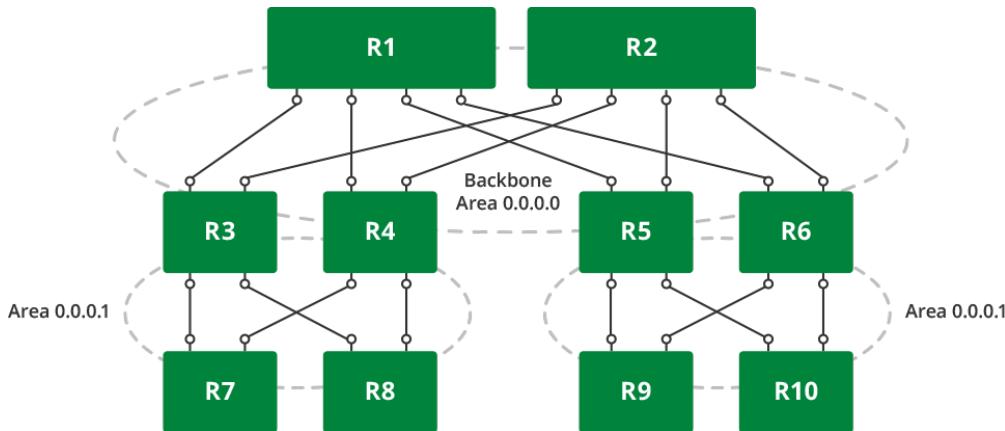
- Scalability and Areas (see page 501)
- Configuring OSPFv2 (see page 502)
 - Enabling the OSPF and Zebra Daemons (see page 502)
 - Configuring OSPF (see page 502)
 - Defining (Custom) OSPF Parameters on the Interfaces (see page 504)
 - OSP SPF Timer Defaults (see page 504)
 - Configure MD5 Authentication for OSPF Neighbors (see page 504)
- Scaling Tips (see page 505)
 - Summarization (see page 505)
 - Stub Areas (see page 506)
 - Running Multiple ospfd Instances (see page 507)
- Unnumbered Interfaces (see page 511)
- Applying a Route Map for Route Updates (see page 512)
- ECMP (see page 513)
- Topology Changes and OSPF Reconvergence (see page 513)
 - Example Configurations (see page 513)
- Debugging OSPF (see page 514)
- Related Information (see page 514)

Scalability and Areas

An increase in the number of nodes affects OSPF scalability in the following ways:

- Memory footprint to hold the entire network topology,
- Flooding performance,
- SPF computation efficiency.

The OSPF protocol advocates hierarchy as a *divide and conquer* approach to achieve high scale. The topology may be divided into areas, resulting in a two-level hierarchy. Area 0 (or 0.0.0.0), called the backbone area, is the top level of the hierarchy. Packets traveling from one non-zero area to another must go via the backbone area. As an example, the leaf-spine topology we have been referring to in the routing section can be divided into areas as follows:



Here are some points to note about areas and OSPF behavior:

- Routers that have links to multiple areas are called *area border routers* (ABR). For example, routers R3, R4, R5, R6 are ABRs in the diagram. An ABR performs a set of specialized tasks, such as SPF computation per area and summarization of routes across areas.
- Most of the LSAs have an area-level flooding scope. These include router LSA, network LSA, and summary LSA.



In the diagram, we reused the same non-zero area address. This is fine since the area address is only a scoping parameter provided to all routers within that area. It has no meaning outside the area. Thus, in the cases where ABRs do not connect to multiple non-zero areas, the same area address can be used, thus reducing the operational headache of coming up with area addresses.

Configuring OSPFv2

Configuring OSPF involves the following tasks:

- Enabling the OSPF daemon
- Enabling OSPF
- Defining (Custom) OSPF parameters on the interfaces

Enabling the OSPF and Zebra Daemons

To enable OSPF, enable the `zebra` and `ospf` daemons, as described in [Configuring Quagga](#) (see page 489), then start the Quagga service:

```
cumulus@switch:~$ sudo systemctl enable quagga.service
cumulus@switch:~$ sudo systemctl start quagga.service
```

Configuring OSPF

As discussed in [Introduction to Routing Protocols](#) (see page 484), there are three steps to the configuration:

1. Identifying the router with the router ID.



2. With whom should the router communicate?
3. What information (most notably the prefix reachability) to advertise?

There are two ways to achieve (2) and (3) in Quagga OSPF:

1. The `network` statement under `router ospf` does both. The statement is specified with an IP subnet prefix and an area address. All the interfaces on the router whose IP address matches the `network` subnet are put into the specified area. OSPF process starts bringing up peering adjacency on those interfaces. It also advertises the interface IP addresses formatted into LSAs (of various types) to the neighbors for proper reachability.

```
cumulus@switch:~$ net add ospf router-id 0.0.0.1
cumulus@switch:~$ net add ospf network 10.0.0.0/16 area 0.0.0.0
cumulus@switch:~$ net add ospf network 192.0.2.0/16 area 0.0.0.1
```

The subnets can be as coarse as possible to cover the most number of interfaces on the router that should run OSPF.

There may be interfaces where it's undesirable to bring up OSPF adjacency. For example, in a data center topology, the host-facing interfaces need not run OSPF; however the corresponding IP addresses should still be advertised to neighbors. This can be achieved using the `passive-interface` construct:

```
cumulus@switch:~$ net add ospf passive-interface swp10
cumulus@switch:~$ net add ospf passive-interface swp11
```

Or use the `passive-interface default` command to put all interfaces as passive and selectively remove certain interfaces to bring up protocol adjacency:

```
R3# configure terminal
R3(config)# router ospf
R3(config-router)# passive-interface default
R3(config-router)# no passive-interface swp1
```

2. Explicitly enable OSPF for each interface by configuring it under the interface configuration mode:

```
cumulus@switch:~$ net add interface swp1 ospf area 0.0.0.0
```

If OSPF adjacency bringup is not desired, you should configure the corresponding interfaces as passive as explained above.

This model of configuration is required for unnumbered interfaces as discussed later in this guide.

For achieving step (3) alone, the quagga configuration provides another method: *redistribution*. For example:

```
cumulus@switch:~$ net add ospf redistribute connected
```

Redistribution, however, unnecessarily loads the database with type-5 LSAs and should be limited to generating real external prefixes (for example, prefixes learned from BGP). In general, it is a good practice to generate local prefixes using `network` and/or `passive-interface` statements.



The OSPF setting `log-adjacency-changes` is enabled by default. It logs a single message when a peer transitions to/from FULL state.

Defining (Custom) OSPF Parameters on the Interfaces

There are a number of custom parameters you can define for OSPF, including:

- Network type, such as point-to-point or broadcast.
- Timer tuning, like a hello interval.
- For unnumbered interfaces (see below (see page 511)), enable OSPF.

To see the list of options, type `net add interface swp1 ospf`, then press **Tab**.

```
cumulus@switch:~$ net add interface swp1 ospf network point-to-point
cumulus@switch:~$ net add interface swp1 ospf hello-interval 5
```

The OSPF configuration is saved in `/etc/quagga/ospfd.conf`.

OSPF SPF Timer Defaults

OSPF uses the following three timers as an exponential backoff, to prevent consecutive SPFs from hammering the CPU:

- 0 ms from initial event until SPF runs
- 50 ms between consecutive SPF runs (the number doubles with each SPF, until it reaches the value of C)
- 5000 ms maximum between SPFs

Configure MD5 Authentication for OSPF Neighbors

Simple text passwords have largely been deprecated in Quagga, in favor of MD5 hash authentication; the following section covers the setup configuration for MD5 authentication on Cumulus Linux switches. You need to do this in the Quagga CLI, `vtysh`.

1. Enter the Quagga CLI on the switch:

```
cumulus@switch:~$ sudo vtysh
```

2. Change into the CLI's configuration mode:

```
switch# configure terminal
```



3. Run the `authentication` command to specify MD5 authentication is used:

```
switch(config)# ip ospf authentication message-digest
```

4. Use the `message-digest-key` command to create a key and key ID for md5:

! In the example input below, `KEYID` represents the key used to create the message digest. This must be consistent across all routers on a link.

`KEY` represents the actual message digest key, and is associated to the given `KEYID`. This value has an upper range of 16 characters. Longer strings will be truncated.

```
switch(config)# ip ospf message-digest-key KEYID md5 KEY
```

5. Save the configuration and exit `vtysh`:

```
switch(config)# exit
switch# write mem
switch# exit
cumulus@switch:~$
```



Existing MD5 authentication hashes can be removed with the `no ip ospf message-digest-key` command.

Scaling Tips

Here are some tips for how to scale out OSPF.

Summarization

By default, an ABR creates a summary (type-3) LSA for each route in an area and advertises it in adjacent areas. Prefix range configuration optimizes this behavior by creating and advertising one summary LSA for multiple routes.

To configure a range:

```
cumulus@switch:~$ sudo vtysh
switch# configure terminal
switch(config)# router ospf
switch(config-router)# area 0.0.0.1 range 30.0.0.0/8
switch(config-router)# exit
switch(config)# exit
```

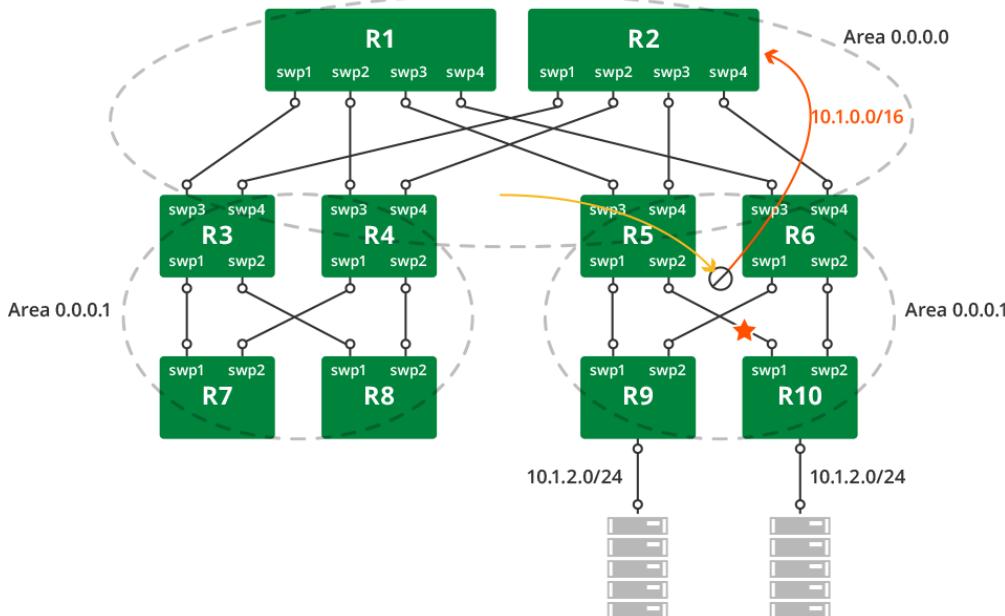
```
switch# write mem
switch# exit
cumulus@switch:~$
```



Summarize in the direction to the backbone. The backbone receives summarized routes and injects them to other areas already summarized.



Summarization can cause non-optimal forwarding of packets during failures. Here is an example scenario:



As shown in the diagram, the ABRs in the right non-zero area summarize the host prefixes as 10.1.0.0/16. When the link between R5 and R10 fails, R5 will send a worse metric for the summary route (metric for the summary route is the maximum of the metrics of intra-area routes that are covered by the summary route. Upon failure of the R5-R10 link, the metric for 10.1.2.0/24 goes higher at R5 as the path is R5-R9-R6-R10). As a result, other backbone routers shift traffic destined to 10.1.0.0/16 towards R6. This breaks ECMP and is an under-utilization of network capacity for traffic destined to 10.1.1.0/24.

Stub Areas

Nodes in an area receive and store intra-area routing information and summarized information about other areas from the ABRs. In particular, a good summarization practice about inter-area routes through prefix range configuration helps scale the routers and keeps the network stable.

Then there are external routes. External routes are the routes redistributed into OSPF from another protocol. They have an AS-wide flooding scope. In many cases, external link states make up a large percentage of the LSDB.

Stub areas alleviate this scaling problem. A stub area is an area that does not receive external route advertisements.

To configure a stub area:



```
cumulus@switch:~$ net add ospf area 0.0.0.1 stub
```

Stub areas still receive information about networks that belong to other areas of the same OSPF domain. Especially, if summarization is not configured (or is not comprehensive), the information can be overwhelming for the nodes. *Totally stubby areas* address this issue. Routers in totally stubby areas keep in their LSDB information about routing within their area, plus the default route.

To configure a totally stubby area:

```
cumulus@switch:~$ net add ospf area 0.0.0.1 stub no-summary
```

Here is a brief tabular summary of the area type differences:

Type	Behavior
Normal non-zero area	LSA types 1, 2, 3, 4 area-scoped, type 5 externals, inter-area routes summarized
Stub area	LSA types 1, 2, 3, 4 area-scoped, No type 5 externals, inter-area routes summarized
Totally stubby area	LSA types 1, 2 area-scoped, default summary, No type 3, 4, 5 LSA types allowed

Running Multiple ospfd Instances

You can configure OSPF to run multiple instances of its `ospfd` daemon. By doing so, Quagga act as if were managing multiple routers on the same switch, one for each OSPF process.

You can run multiple `ospfd` instances with OSPFv2 only, not with OSPFv3. The functional equivalent for [BGP \(see page 516\)](#) is a VRF, which provides for multiple BGP router statements, although only one instance of the `bgpd` daemon is running on the switch. Cumulus Quagga supports up to 5 instances currently, and the instance ID must be within the range of 1 through 65535.

To configure multiple `ospfd` instances, do the following:

1. Edit `/etc/quagga/daemons` and add `ospfd_instances="instance1 instance2 ..."` to the `ospfd` line, specifying an instance ID for each separate instance. For example, the following configuration has OSPF enabled with 2 `ospfd` instances, 11 and 22:

```
cumulus@switch:~$ cat /etc/quagga/daemons
zebra=yes
bgpd=no
ospfd=yes ospfd_instances="11 22"
ospf6d=no
ripd=no
ripngd=no
isisd=no
```

2. After you modify the daemons file, restart Quagga:

```
cumulus@switch:~$ sudo systemctl restart quagga
```

3. Configure each instance:

```
cumulus@switch:~$ net add interface swp1 ospf instance-id 11
cumulus@switch:~$ net add interface swp1 ospf area 0.0.0.0
cumulus@switch:~$ net add ospf router-id 1.1.1.1
cumulus@switch:~$ net add interface swp2 ospf instance-id 22
cumulus@switch:~$ net add interface swp2 ospf area 0.0.0.0
cumulus@switch:~$ net add ospf router-id 1.1.1.1
```

4. Confirm the configuration:

```
cumulus@switch:~$ net show configuration ospf

hostname zebra
log file /var/log/quagga/zebra.log
username cumulus nopassword

service integrated-vtysh-config

interface eth0
    ipv6 nd suppress-ra
    link-detect

interface lo
    link-detect

interface swp1
    ip ospf 11 area 0.0.0.0
    link-detect

interface swp2
    ip ospf 22 area 0.0.0.0
    link-detect

interface swp45
    link-detect

interface swp46
    link-detect

interface swp47
    link-detect

interface swp48
```

```

link-detect

interface swp49
link-detect

interface swp50
link-detect

interface swp51
link-detect

interface swp52
link-detect

interface vagrant
link-detect

router ospf 11
ospf router-id 1.1.1.1

router ospf 22
ospf router-id 1.1.1.1

ip forwarding
ipv6 forwarding

line vty

end

```

5. Confirm that all the OSPF instances are running:

```

cumulus@switch:~$ ps -ax | grep ospf
21135 ? S<s 0:00 /usr/lib/quagga/ospfd --daemon -A 127.
0.0.1 -n 11
21139 ? S<s 0:00 /usr/lib/quagga/ospfd --daemon -A 127.
0.0.1 -n 22
21160 ? S<s 0:01 /usr/lib/quagga/watchquagga -adz -r
/usr/sbin/servicebBquaggabBrestartbB%-s -s /usr/sbin
/servicebBquaggabBstartbB%-s -k /usr/sbin/servicebBquaggabBstopbB%
s -b bB -t 30 zebra ospfd-11 ospfd-22 pimd
22021 pts/3 S+ 0:00 grep ospf

```

Caveats

You can use the `redistribute ospf` option in your `Quagga.conf` file works with this so you can route between the instances. Specify the instance ID for the other OSPF instance. For example:



```
cumulus@switch:~$ cat /etc/quagga/Quagga.conf
hostname zebra
log file /var/log/quagga/zebra.log
username cumulus nopassword
!
service integrated-vtysh-config
!

...
!

router ospf 11
  ospf router-id 1.1.1.1
!
router ospf 22
  ospf router-id 1.1.1.1
  redistribute ospf 11
!
...
```



Don't specify a process ID unless you are using multi-instance OSPF.



If you disabled the [integrated \(see page 490\)](#) Quagga configuration, you must create a separate `ospfd` configuration file for each instance. The `ospfd.conf` file must include the instance ID in the file name. Continuing with our example, you would create `/etc/quagga/ospfd-11.conf` and `/etc/quagga/ospfd-22.conf`.

```
cumulus@switch:~$ cat /etc/quagga/ospfd-11.conf
!
hostname zebra
log file /var/log/quagga/zebra.log
username cumulus nopassword
!
service integrated-vtysh-config
!
interface eth0
  ipv6 nd suppress-ra
  link-detect
!
interface lo
  link-detect
!
interface swp1
  ip ospf 11 area 0.0.0.0
  link-detect
!
```



```
interface swp2
    ip ospf 22 area 0.0.0.0
    link-detect
!
interface swp45
    link-detect
!
interface swp46
    link-detect
!
interface swp47
    link-detect
!
interface swp48
    link-detect
!
interface swp49
    link-detect
!
interface swp50
    link-detect
!
interface swp51
    link-detect
!
interface swp52
    link-detect
!
interface vagrant
    link-detect
!
router ospf 11
    ospf router-id 1.1.1.1
!
router ospf 22
    ospf router-id 1.1.1.1
!
ip forwarding
ipv6 forwarding
!
line vty
!
```

Unnumbered Interfaces

Unnumbered interfaces are interfaces without unique IP addresses. In OSPFv2, configuring unnumbered interfaces reduces the links between routers into pure topological elements, which dramatically simplifies network configuration and reconfiguration. In addition, the routing database contains only the real networks, so the memory footprint is reduced and SPF is faster.



Unnumbered is usable for point-to-point interfaces only.



If there is a `network <network number>/<mask> area <area ID>` command present in the Quagga configuration, the `ip ospf area <area ID>` command is rejected with the error "Please remove network command first." This prevents you from configuring other areas on some of the unnumbered interfaces. You can use either the `network area` command or the `ospf area` command in the configuration, but not both.



Unless the Ethernet media is intended to be used as a LAN with multiple connected routers, we recommend configuring the interface as point-to-point. It has the additional advantage of a simplified adjacency state machine; there is no need for DR/BDR election and *LSA reflection*. See [RFC5309](#) for a more detailed discussion.

To configure an unnumbered interface, take the IP address of another interface (called the *anchor*) and use that as the IP address of the unnumbered interface:

```
cumulus@switch:~$ net add loopback lo ip address 192.0.2.1/32
cumulus@switch:~$ net add interface swp1 ip address 192.0.2.1/32
cumulus@switch:~$ net add interface swp2 ip address 192.0.2.1/32
```

These commands create the following configuration in the `/etc/network/interfaces` file:

```
auto lo
iface lo inet loopback
    address 192.0.2.1/32

auto swp1
iface swp1
    address 192.0.2.1/32

auto swp2
iface swp2
    address 192.0.2.1/32
```

To enable OSPF on an unnumbered interface:

```
cumulus@switch:~$ net add interface swp1 ospf area 0.0.0.1
```

Applying a Route Map for Route Updates

To apply a `route map` to filter route updates from Zebra into the Linux kernel:

```
cumulus@switch:$ net add routing protocol ospf route-map <route-map-name>
```

ECMP

During SPF computation for an area, if OSPF finds multiple paths with equal cost (metric), all those paths are used for forwarding. For example, in the reference topology diagram, R8 uses both R3 and R4 as next hops to reach a destination attached to R9.

Topology Changes and OSPF Reconvergence

Topology changes usually occur due to one of four events:

1. Maintenance of a router node
2. Maintenance of a link
3. Failure of a router node
4. Failure of a link

For the maintenance events, operators typically raise the OSPF administrative weight of the link(s) to ensure that all traffic is diverted from the link or the node (referred to as *costing out*). The speed of reconvergence does not matter. Indeed, changing the OSPF cost causes LSAs to be reissued, but the links remain in service during the SPF computation process of all routers in the network.

For the failure events, traffic may be lost during reconvergence; that is, until SPF on all nodes computes an alternative path around the failed link or node to each of the destinations. The reconvergence depends on layer 1 failure detection capabilities and at the worst case *DeadInterval* OSPF timer.

Example Configurations

Example configuration for event 1, using vtysh:

```
cumulus@switch:~$ sudo vtysh
switch# configure terminal
switch(config)# router ospf
switch(config-router)# max-metric router-lsa administrative
switch(config-router)# exit
switch(config)# exit
switch# write mem
switch# exit
cumulus@switch:~$
```

Example configuration for event 2:

```
cumulus@switch:~$ net add interface swp1 ospf cost 65535
```



Debugging OSPF

[OperState](#) lists all the commands to view the operational state of OSPF.

The three most important states while troubleshooting the protocol are:

1. Neighbors, with `net show ospf neighbor`. This is the starting point to debug neighbor states (also see `tcpdump` below).
2. Database, with `net show ospf database`. This is the starting point to verify that the LSDB is, in fact, synchronized across all routers in the network. For example, sweeping through the output of `show ip ospf database router` taken from all routers in an area will ensure if the topology graph building process is complete; that is, every node has seen all the other nodes in the area.
3. Routes, with `net show route ospf`. This is the outcome of SPF computation that gets downloaded to the forwarding table, and is the starting point to debug, for example, why an OSPF route is not being forwarded correctly.

[Debugging-OSPF](#) lists all of the OSPF debug options.

Using `zebra` under `vtysh`:

```
cumulus@switch:~$ sudo vtysh
switch# show [zebra]

IOBJECT := { events | status | timers }
OOBJECT := { interface | redistribute }
POBJECT := { all | dd | hello | ls-ack | ls-request | ls-update }
ZOBJECT := { all | events | kernel | packet | rib |
```

Using `tcpdump` to capture OSPF packets:

```
cumulus@switch:~$ sudo tcpdump -v -i swp1 ip proto ospf
```

Related Information

- Bidirectional forwarding detection (see page 549) (BFD) and OSPF
- en.wikipedia.org/wiki/Open_Shortest_Path_First
- www.nongnu.org/quagga/docs/docs-info.html#OSPFv2
- Perlman, Radia (1999). Interconnections: Bridges, Routers, Switches, and Internetworking Protocols (2 ed.). Addison-Wesley.
- Moy, John T. OSPF: Anatomy of an Internet Routing Protocol. Addison-Wesley.

Open Shortest Path First v3 - OSPFv3 - Protocol

OSPFv3 is a revised version of OSPFv2 to support the IPv6 address family. Refer to [Open Shortest Path First \(OSPF\) Protocol \(see page 500\)](#) for a discussion on the basic concepts, which remain the same between the two versions.



OSPFv3 has changed the formatting in some of the packets and LSAs either as a necessity to support IPv6 or to improve the protocol behavior based on OSPFv2 experience. Most notably, v3 defines a new LSA, called intra-area prefix LSA to separate out the advertisement of stub networks attached to a router from the router LSA. It is a clear separation of node topology from prefix reachability and lends itself well to an optimized SPF computation.



IETF has defined extensions to OSPFv3 to support multiple address families (that is, both IPv6 and IPv4). Quagga (see page 488) does not support it yet.

Contents

This chapter covers ...

- [Configuring OSPFv3 \(see page 515\)](#)
- [Unnumbered Interfaces \(see page 515\)](#)
- [Debugging OSPF \(see page 516\)](#)
- [Related Information \(see page 516\)](#)

Configuring OSPFv3

Configuring OSPFv3 involves the following tasks:

1. Enabling the `zebra` and `ospf6` daemons, as described in [Configuring Quagga \(see page 489\)](#) then start the Quagga service:

```
cumulus@switch:~$ sudo systemctl enable quagga.service
cumulus@switch:~$ sudo systemctl start quagga.service
```

2. Enabling OSPF6 and map interfaces to areas:

```
cumulus@switch:~$ net add ospf6 router-id 0.0.0.1
cumulus@switch:~$ net add ospf6 interface swp1 area 0.0.0.0
cumulus@switch:~$ net add ospf6 interface swp2 area 0.0.0.1
```

3. Defining (custom) OSPF6 parameters on the interfaces, such as:

- a. Network type (such as point-to-point, broadcast)
- b. Timer tuning (for example, hello interval)

```
cumulus@switch:~$ net add interface swp1 ospf6 network point-to-
point
cumulus@switch:~$ net add interface swp1 ospf6 hello-interval 5
```

The OSPFv3 configuration is saved in `/etc/quagga/ospf6d.conf`.



Unnumbered Interfaces

Unlike OSPFv2, OSPFv3 intrinsically supports unnumbered interfaces. Forwarding to the next hop router is done entirely using IPv6 link local addresses. Therefore, you are not required to configure any global IPv6 address to interfaces between routers.

Debugging OSPF

See [Debugging OSPF \(see page 514\)](#) for OSPFv2 for the troubleshooting discussion. The equivalent commands are:

```
cumulus@switch:~$ net show ospf6 neighbor [detail|drchoice]
cumulus@switch:~$ net show ospf6 database [adv-
router|detail|dump|internal|linkstate-id|self-originated]
cumulus@switch:~$ net show route ospf6
```

Another helpful command is `net show ospf6 spf tree`. It dumps the node topology as computed by SPF to help visualize the network view.

Related Information

- Bidirectional forwarding detection ([see page 549](#)) (BFD) and OSPF
- en.wikipedia.org/wiki/Open_Shortest_Path_First
- www.nongnu.org/quagga/docs/docs-info.html#OSPFv3

Border Gateway Protocol - BGP

BGP is the routing protocol that runs the Internet. It is an increasingly popular protocol for use in the data center as it lends itself well to the rich interconnections in a Clos topology. Specifically:

- It does not require routing state to be periodically refreshed unlike OSPF.
- It is less chatty than its link-state siblings. For example, a link or node transition can result in a bestpath change, causing BGP to send updates.
- It is multi-protocol and extensible.
- There are many robust vendor implementations.
- The protocol is very mature and comes with many years of operational experience.

This [IETF draft](#) provides further details of the use of BGP within the data center.

Contents

This chapter covers ...

- Autonomous System Number (ASN) ([see page 518](#))
- eBGP and iBGP ([see page 518](#))
- Route Reflectors ([see page 518](#))
- ECMP with BGP ([see page 519](#))
 - Maximum Paths ([see page 519](#))



- BGP for Both IPv4 and IPv6 (see page 520)
- Configuring BGP (see page 520)
- Using BGP Unnumbered Interfaces (see page 521)
 - BGP and Extended Next-hop Encoding (see page 521)
 - Configuring BGP Unnumbered Interfaces (see page 521)
 - Managing Unnumbered Interfaces (see page 523)
 - How traceroute Interacts with BGP Unnumbered Interfaces (see page 524)
 - Advanced: Understanding How Next-hop Fields Are Set (see page 525)
 - Limitations (see page 526)
- BGP add-path (see page 526)
 - BGP add-path RX (see page 526)
 - BGP add-path TX (see page 528)
- Fast Convergence Design Considerations (see page 530)
 - Specifying the Interface Name in the neighbor Command (see page 530)
- Using Peer Groups to Simplify Configuration (see page 531)
- Configuring BGP Dynamic Neighbors (see page 531)
- Configuring BGP Peering Relationships across Switches (see page 532)
- Configuring MD5-enabled BGP Neighbors (see page 534)
 - Manually Configuring an MD5-enabled BGP Neighbor (see page 535)
- Configuring BGP TTL Security (see page 536)
- Configuration Tips (see page 538)
 - BGP Advertisement Best Practices (see page 538)
 - Utilizing Multiple Routing Tables and Forwarding (see page 538)
 - Using BGP Community Lists (see page 538)
 - Additional Default Settings (see page 539)
 - Configuring BGP Neighbor maximum-prefixes (see page 539)
- Troubleshooting BGP (see page 539)
 - Debugging Tip: Logging Neighbor State Changes (see page 542)
 - Troubleshooting Link-local Addresses (see page 542)
- Enabling Read-only Mode (see page 545)
- Applying a Route Map for Route Updates (see page 545)
 - Filtering Routes from BGP into Zebra (see page 546)
 - Filtering Routes from Zebra into the Linux Kernel (see page 546)
- Protocol Tuning (see page 546)
 - Converging Quickly On Link Failures (see page 546)
 - Converging Quickly On Soft Failures (see page 546)
 - Reconnecting Quickly (see page 547)
 - Advertisement Interval (see page 547)

- [Caveats and Errata \(see page 548\)](#)
 - [ttl-security Issue \(see page 548\)](#)
 - [Related Information \(see page 549\)](#)

Autonomous System Number (ASN)

One of the key concepts in BGP is an *autonomous system number* or ASN. An [autonomous system](#) is defined as a set of routers under a common administration. Since BGP was originally designed to peer between independently managed enterprises and/or service providers, each such enterprise is treated as an autonomous system, responsible for a set of network addresses. Each such autonomous system is given a unique number called its ASN. ASNs are handed out by a central authority (ICANN). However, ASNs between 64512 and 65535 are reserved for private use. Using BGP within the data center relies on either using this number space or else using the single ASN you own.

The ASN is central to how BGP builds a forwarding topology. A BGP route advertisement carries with it not only the originator's ASN, but also the list of ASNs that this route advertisement has passed through. When forwarding a route advertisement, a BGP speaker adds itself to this list. This list of ASNs is called the *AS path*. BGP uses the AS path to detect and avoid loops.

ASNs were originally 16-bit numbers, but were later modified to be 32-bit. Quagga supports both 16-bit and 32-bit ASNs, but most implementations still run with 16-bit ASNs.

eBGP and iBGP

When BGP is used to peer between autonomous systems, the peering is referred to as *external BGP* or eBGP. When BGP is used within an autonomous system, the peering used is referred to as *internal BGP* or iBGP. eBGP peers have different ASNs while iBGP peers have the same ASN.

While the heart of the protocol is the same when used as eBGP or iBGP, there is a key difference in the protocol behavior between use as eBGP and iBGP: an iBGP speaker does not forward routing information learned from one iBGP peer to another iBGP peer to prevent loops. eBGP prevents loops using the AS_Path attribute.

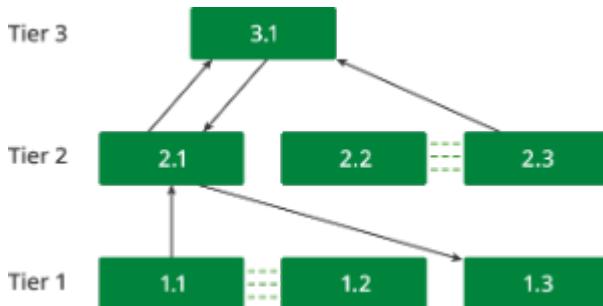
Therefore, all iBGP speakers need to be peered with each other in a full mesh. In a large network, this requirement can quickly become unscalable. The most popular method to scale iBGP networks is to introduce a *route reflector*.

Route Reflectors

Route reflectors are quite easy to understand in a Clos topology. In a two-tier Clos network, the leaf (or tier 1) switches are the only ones connected to end stations. Subsequently, this means that the spines themselves do not have any routes to announce. They're merely **reflecting** the routes announced by one leaf to the other leaves. Thus, the spine switches function as route reflectors while the leaf switches serve as route reflector clients.

In a three-tier network, the tier 2 nodes (or mid-tier spines) act as both route reflector servers and route reflector clients. They act as route reflectors because they announce the routes learned from the tier 1 nodes to other tier 1 nodes and to tier 3 nodes. They also act as route reflector clients to the tier 3 nodes, receiving routes learned from other tier 2 nodes. Tier 3 nodes act only as route reflectors.

In the following illustration, tier 2 node 2.1 is acting as a route reflector server, announcing the routes between tier 1 nodes 1.1 and 1.2 to tier 1 node 1.3. It is also a route reflector client, learning the routes between tier 2 nodes 2.2 and 2.3 from the tier 3 node, 3.1.



⚠ Configuring route-reflector-client Requires Specific Order

When configuring a route to be a route reflector client, the Quagga configuration must be specified in a specific order; otherwise, the router will not be a route reflector client.

The `net add bgp neighbor <IPv4/IPv6> route-reflector-client` command must be done after the `net add bgp neighbor <IPv4/IPv6> activate` command; otherwise, the `route-reflector-client` command is ignored. For example:

```
cumulus@switch:~$ net add bgp ipv4 unicast neighbor 14.0.0.9
activate
cumulus@switch:~$ net add bgp neighbor 14.0.0.9 next-hop-self
cumulus@switch:~$ net add bgp neighbor 14.0.0.9 route-
reflector-client >>> Must be after activate
cumulus@switch:~$ net add bgp neighbor 2001:ded:beef:2::1
remote-as 65000
cumulus@switch:~$ net add bgp ipv6 unicast redistribute
connected
cumulus@switch:~$ net add bgp maximum-paths ibgp 4
cumulus@switch:~$ net add bgp neighbor 2001:ded:beef:2::1
activate
cumulus@switch:~$ net add bgp neighbor 2001:ded:beef:2::1 next-
hop-self
cumulus@switch:~$ net add bgp neighbor 2001:ded:beef:2::1
route-reflector-client >>> Must be after activate
```

ECMP with BGP

If a BGP node hears a prefix **p** from multiple peers, it has all the information necessary to program the routing table to forward traffic for that prefix **p** through all of these peers. Thus, BGP supports equal-cost multipathing (ECMP).

In order to perform ECMP in BGP, you may need to configure `net add bgp bestpath as-path multipath-relax` (if you're using eBGP).

Maximum Paths

In Cumulus Linux, the BGP `maximum-paths` setting is enabled by default, so multiple routes are already installed. The default setting is 64 paths.



BGP for Both IPv4 and IPv6

Unlike OSPF, which has separate versions of the protocol to announce IPv4 and IPv6 routes, BGP is a multi-protocol routing engine, capable of announcing both IPv4 and IPv6 prefixes. It supports announcing IPv4 prefixes over an IPv4 session and IPv6 prefixes over an IPv6 session. It also supports announcing prefixes of both these address families over a single IPv4 session or over a single IPv6 session.

Configuring BGP

A basic BGP configuration looks like the following. However, the rest of this chapter discusses how to configure various other features, from unnumbered interfaces to route maps.

1. Enable the BGP and Zebra daemons, `zebra` and `bgpd`, then enable the Quagga service and start it, as described in [Configuring Quagga \(see page 489\)](#).
2. Identify the BGP node by assigning an ASN and `router-id`:

```
cumulus@switch:~$ net add bgp autonomous-system 65000
cumulus@switch:~$ net add bgp router-id 10.0.0.1
```

3. Specify to whom it must disseminate routing information:

```
cumulus@switch:~$ net add bgp neighbor 10.0.0.2 remote-as
external
```

If it is an iBGP session, the `remote-as` is the same as the local AS:

```
cumulus@switch:~$ net add bgp neighbor 10.0.0.2 remote-as
internal
```

Specifying the peer's IP address allows BGP to set up a TCP socket with this peer, but it doesn't distribute any prefixes to it, unless it is explicitly told that it must via the `activate` command:

```
cumulus@switch:~$ net add bgp ipv4 unicast neighbor 10.0.0.2
activate
cumulus@switch:~$ net add bgp ipv6 unicast neighbor 2001:db8:
0002::0a00:0002 activate
```

As you can see, `activate` has to be specified for each address family that is being announced by the BGP session.

4. Specify some properties of the BGP session:

```
cumulus@switch:~$ net add bgp neighbor 10.0.0.2 next-hop-self
```

If this is a route-reflector client, it can be specified as follows:



```
cumulus@switchRR:~$ net add bgp neighbor 10.0.0.1 route-reflector-client
```



It is node *switchRR*, the route reflector, on which the peer is specified as a client.

5. Specify what prefixes to originate:

```
cumulus@switch:~$ net add bgp ipv4 unicast network 192.0.2.0/24
cumulus@switch:~$ net add bgp ipv4 unicast network 203.0.113.1/24
```

Using BGP Unnumbered Interfaces

Unnumbered interfaces are interfaces without unique IP addresses. In BGP, you configure unnumbered interfaces using *extended next-hop encoding* (ENHE), which is defined by [RFC 5549](#). BGP unnumbered interfaces provide a means of advertising an IPv4 route with an IPv6 next-hop. Prior to RFC 5549, an IPv4 route could be advertised only with an IPv4 next-hop.

BGP unnumbered interfaces are particularly useful in deployments where IPv4 prefixes are advertised through BGP over a section without any IPv4 address configuration on links. As a result, the routing entries are also IPv4 for destination lookup and have IPv6 next-hops for forwarding purposes.

BGP and Extended Next-hop Encoding

Once enabled and active, BGP makes use of the available IPv6 next-hops for advertising any IPv4 prefixes. BGP learns the prefixes, calculates the routes and installs them in IPv4 AFI to IPv6 AFI format. However, ENHE in Cumulus Linux does not install routes into the kernel in IPv4 prefix to IPv6 next-hop format. For link-local peerings enabled by dynamically learning the other end's link-local address using IPv6 neighbor discovery router advertisements, an IPv6 next-hop is converted into an IPv4 link-local address and a static neighbor entry is installed for this IPv4 link-local address with the MAC address derived from the link-local address of the other end.



It is assumed that the IPv6 implementation on the peering device will use the MAC address as the interface ID when assigning the IPv6 link-local address, as suggested by RFC 4291.

Configuring BGP Unnumbered Interfaces

Configuring a BGP unnumbered interface requires enabling IPv6 neighbor discovery router advertisements. The `interval` you specify is measured in seconds, and defaults to 600 seconds. Extended next-hop encoding is sent only for the link-local address peerings:

```
cumulus@switch:~$ net add bgp autonomous-system 65020
cumulus@switch:~$ net add bgp router-id 10.0.0.21
cumulus@switch:~$ net add bgp bestpath as-path multipath-relax
```



```
cumulus@switch:~$ net add bgp bestpath compare-routerid
cumulus@switch:~$ net add bgp neighbor fabric peer-group
cumulus@switch:~$ net add bgp neighbor fabric remote-as external
cumulus@switch:~$ net add bgp neighbor fabric description Internal
Fabric Network
cumulus@switch:~$ net add bgp neighbor fabric capability extended-
nexthop
cumulus@switch:~$ net add bgp neighbor swp1 interface peer-group
fabric
cumulus@switch:~$ net add bgp neighbor swp2 interface peer-group
fabric
cumulus@switch:~$ net add bgp neighbor swp3 interface peer-group
fabric
cumulus@switch:~$ net add bgp neighbor swp4 interface peer-group
fabric
cumulus@switch:~$ net add bgp neighbor swp29 interface peer-group
fabric
cumulus@switch:~$ net add bgp neighbor swp30 interface peer-group
fabric
```

These commands create the following configuration in the `/etc/quagga/Quagga.conf` file:

```
router bgp 65020
bgp router-id 10.0.0.21
bgp bestpath as-path multipath-relax
bgp bestpath compare-routerid
neighbor fabric peer-group
neighbor fabric remote-as external
neighbor fabric description Internal Fabric Network
neighbor fabric capability extended-nexthop
neighbor swp1 interface peer-group fabric
neighbor swp2 interface peer-group fabric
neighbor swp3 interface peer-group fabric
neighbor swp4 interface peer-group fabric
neighbor swp29 interface peer-group fabric
neighbor swp30 interface peer-group fabric
!
```

Notice above, for an unnumbered configuration, you can use a single command to configure a neighbor and attach it to a [peer group \(see page 531\)](#) (making sure to substitute for the interface and peer group below):

```
cumulus@switch:~$ net add bgp neighbor <swpX> interface peer-group
<group name>
```



Managing Unnumbered Interfaces

All the relevant BGP commands are now capable of showing IPv6 next-hops and/or the interface name for any IPv4 prefix:

```
cumulus@switch:~$ net show bgp

show bgp ipv4 unicast
=====
BGP table version is 6, local router ID is 10.0.0.11
Status codes: s suppressed, d damped, h history, * valid, > best, =
multipath,
          i internal, r RIB-failure, S Stale, R Removed
Origin codes: i - IGP, e - EGP, ? - incomplete
      Network          Next Hop          Metric LocPrf Weight Path
* > 10.0.0.11/32    0.0.0.0                  0          32768 ?
* > 10.0.0.12/32    swp51                   0 65020 65012 ?
*=             swp52                   0 65020 65012 ?
* > 10.0.0.21/32    swp51                   0 65020 ?
* > 10.0.0.22/32    swp52                   0 65020 ?
* > 172.16.1.0/24   0.0.0.0                  0          32768 i
* > 172.16.2.0/24   swp51                   0 65020 65012 i
*=             swp52                   0 65020 65012 i
Total number of prefixes 6

show bgp ipv6 unicast
=====
No BGP network exists
```

Quagga RIB commands are also modified:

```
cumulus@switch:~$ net show route
RIB entry for route
=====
Codes: K - kernel route, C - connected, S - static, R - RIP,
       O - OSPF, I - IS-IS, B - BGP, P - PIM, T - Table,
       > - selected route, * - FIB route
K>* 0.0.0.0/0 via 192.168.0.254, eth0
C>* 10.0.0.11/32 is directly connected, lo
B>* 10.0.0.12/32 [20/0] via fe80::4638:39ff:fe00:5c, swp51, 1d01h04m
   *           via fe80::4638:39ff:fe00:2b, swp52, 1d01h04m
B>* 10.0.0.21/32 [20/0] via fe80::4638:39ff:fe00:5c, swp51, 1d01h04m
B>* 10.0.0.22/32 [20/0] via fe80::4638:39ff:fe00:2b, swp52, 1d01h04m
C>* 172.16.1.0/24 is directly connected, br0
B>* 172.16.2.0/24 [20/0] via fe80::4638:39ff:fe00:5c, swp51, 1d01h04m
   *           via fe80::4638:39ff:fe00:2b, swp52, 1d01h04m
C>* 192.168.0.0/24 is directly connected, eth0
```

The following commands show how the IPv4 link-local address 169.254.0.1 is used to install the route and static neighbor entry to facilitate proper forwarding without having to install an IPv4 prefix with IPv6 next-hop in the kernel:

```
cumulus@switch:~$ net show route 10.0.0.12
RIB entry for 10.0.0.12
=====
Routing entry for 10.0.0.12/32
    Known via "bgp", distance 20, metric 0, best
    Last update 1d01h06m ago
    * fe80::4638:39ff:fe00:5c, via swp51
    * fe80::4638:39ff:fe00:2b, via swp52

FIB entry for 10.0.0.12
=====
10.0.0.12 proto zebra metric 20
    nexthop via 169.254.0.1 dev swp51 weight 1 onlink
    nexthop via 169.254.0.1 dev swp52 weight 1 onlink
```

You can use this `iproute2` command to display more neighbor information:

```
cumulus@switch:~$ ip neighbor
192.168.0.254 dev eth0 lladdr 44:38:39:00:00:5f REACHABLE
169.254.0.1 dev swp52 lladdr 44:38:39:00:00:2b PERMANENT
169.254.0.1 dev swp51 lladdr 44:38:39:00:00:5c PERMANENT
fe80::4638:39ff:fe00:2b dev swp52 lladdr 44:38:39:00:00:2b router
REACHABLE
fe80::4638:39ff:fe00:5c dev swp51 lladdr 44:38:39:00:00:5c router
REACHABLE
```

How traceroute Interacts with BGP Unnumbered Interfaces

Every router or end host must have an IPv4 address in order to complete a `traceroute` of IPv4 addresses. In this case, the IPv4 address used is that of the loopback device.

Even if ENHE is not used in the data center, link addresses are not typically advertised. This is because:

- Link addresses take up valuable FIB resources. In a large Clos environment, the number of such addresses can be quite large.
- Link addresses expose an additional attack vector for intruders to use to either break in or engage in DDOS attacks.

Therefore, assigning an IP address to the loopback device is essential.

Advanced: Understanding How Next-hop Fields Are Set

This section describes how the IPv6 next-hops are set in the MP_REACH_NLRI (multiprotocol reachable NLRI) initiated by the system, which applies whether IPv6 prefixes or IPv4 prefixes are exchanged with ENHE. There are two main aspects to determine — how many IPv6 next-hops are included in the MP_REACH_NLRI (since the RFC allows either one or two next-hops) and the values of the next-hop(s). This section also describes how a received MP_REACH_NLRI is handled as far as processing IPv6 next-hops.

- Whether peering to a global IPv6 address or link-local IPv6 address, the determination whether to send one or two next-hops is as follows:
 1. If reflecting the route, two next-hops are sent only if the peer has `nexthop-local unchanged` configured and the attribute of the received route has an IPv6 link-local next-hop; otherwise, only one next-hop is sent.
 2. Otherwise (if it's not reflecting the route), two next-hops are sent if explicitly configured (`nexthop-local unchanged`) or the peer is directly connected (that is, either peering is on link-local address or the global IPv4 or IPv6 address is *directly connected*) and the route is either a local/self-originated route or the peer is an eBGP peer.
 3. In all other cases, only one next-hop gets sent, unless an outbound route map adds another next-hop.
- `route-map` can impose two next-hops in scenarios where Cumulus Linux would only send one next-hop — by specifying `set ipv6 nexthop link-local`.
- For all routes to eBGP peers and self-originated routes to iBGP peers, the global next-hop (first value) is the peering address of the local system. If the peering is on the link-local address, this is the global IPv6 address on the peering interface, if present; otherwise, it is the link-local IPv6 address on the peering interface.
- For other routes to iBGP peers (eBGP to iBGP or reflected), the global next-hop will be the global next-hop in the received attribute.



If this address were a link-local IPv6 address, it would get reset so that the link-local IPv6 address of the eBGP peer is not passed along to an iBGP peer, which most likely may be on a different link.

- `route-map` and/or the peer configuration can change the above behavior. For example, `route-map` can set the global IPv6 next-hop or the peer configuration can set it to `self` — which is relevant for *iBGP* peers. The route map or peer configuration can also set the next-hop to `unchanged`, which ensures the source IPv6 global next-hop is passed around — which is relevant for *eBGP* peers.
- Whenever two next-hops are being sent, the link-local next-hop (the second value of the two) is the link-local IPv6 address on the peering interface unless it is due to `nh-local-unchanged` or `route-map` has set the link-local next-hop.
- Network administrators cannot set **martian values** for IPv6 next-hops in `route-map`. Also, global and link-local next-hops are validated to ensure they match the respective address types.
- In a received update, a martian check is imposed for the IPv6 global next-hop. If the check fails, it gets treated as an implicit withdraw.
- If two next-hops are received in an update and the second next-hop is not a link-local address, it gets ignored and the update is treated as if only one next-hop was received.



- Whenever two next-hops are received in an update, the second next-hop is used to install the route into `zebra`. As per the previous point, it is already assured that this is a link-local IPv6 address. Currently, this is assumed to be reachable and is not registered with NHT.
- When `route-map` specifies the next-hop as `peer-address`, the global IPv6 next-hop as well as the link-local IPv6 next-hop (if it's being sent) is set to the *peering address*. If the peering is on a link-local address, the former could be the link-local address on the peering interface, unless there is a global IPv6 address present on this interface.

The above rules imply that there are scenarios where a generated update has two IPv6 next-hops, and both of them are the IPv6 link-local address of the peering interface on the local system. If you are peering with a switch or router that is not running Cumulus Linux and expects the first next-hop to be a global IPv6 address, a route map can be used on the sender to specify a global IPv6 address. This conforms with the recommendations in the Internet draft [draft-kato-bgp-ipv6-link-local-00.txt](#), "BGP4+ Peering Using IPv6 Link-local Address".

Limitations

- Interface-based peering with separate IPv4 and IPv6 sessions is not supported.
- ENHE is sent for IPv6 link-local peerings only.
- If an IPv4 /30 or /31 IP address is assigned to the interface IPv4 peering will be used over IPv6 link-local peering.
- If the default router lifetime in the generated IPv6 route advertisements (RA) is set to 0, the receiving Quagga instance will drop the RA if it is on a Cumulus Linux **2.5.z** switch. To work around this issue, either:
 - Explicitly configure the switch to advertise a router lifetime of 0, unless a value is specifically set by the operator — with the assumption that the host is running Cumulus Linux 3.y.z version of Quagga. When hosts see an IPv6 RA with a router lifetime of 0, they won't make that router a default router.
 - Use the `sysctl` on the host — `net.ipv6.conf.all.accept_ra_defrtr`. However, this requires applying this setting on all hosts, which may mean many hosts, especially if Quagga is run on the hosts.

BGP add-path

In Cumulus Linux 3.0 and later, BGP and static routing (IPv4 and IPv6) are supported within a VRF context. For more information, refer to [Virtual Routing and Forwarding - VRF \(see page 574\)](#).

BGP add-path RX

BGP add-path RX allows BGP to receive multiple paths for the same prefix. A path identifier is used so that additional paths do not override previously advertised paths. No additional configuration is required for BGP add-path RX.



BGP advertises the add-path RX capability by default. Add-Path TX requires an administrator to enable it. Enabling TX resets the session.

To view the existing capabilities, run `net show bgp neighbor`. They can be seen listed in the subsection *Add Path*; below *Neighbor capabilities*:



```
cumulus@leaf01:~$ net show bgp neighbor
BGP neighbor on swp51: fe80::4638:39ff:fe00:5c, remote AS 65020,
local AS 65011, external link
Hostname: spine01
Member of peer-group fabric for session parameters
  BGP version 4, remote router ID 10.0.0.21
  BGP state = Established, up for 1d01h15m
  Last read 00:00:00, Last write 1d01h15m
  Hold time is 3, keepalive interval is 1 seconds
  Configured hold time is 3, keepalive interval is 1 seconds
Neighbor capabilities:
  4 Byte AS: advertised and received
  AddPath:
    IPv4 Unicast: RX advertised IPv4 Unicast and received
    Extended nexthop: advertised and received
    Address families by peer:
      IPv4 Unicast
      Route refresh: advertised and received(old & new)
      Address family IPv4 Unicast: advertised and received
      Hostname Capability: advertised and received
      Graceful Restart Capabilty: advertised and received
      Remote Restart timer is 120 seconds
    Address families by peer:
      none
...
...
```

The example output above shows that additional BGP paths can be sent and received (TX and RX are advertised). It also shows that the BGP neighbor, fe80::4638:39ff:fe00:5c, supports both.

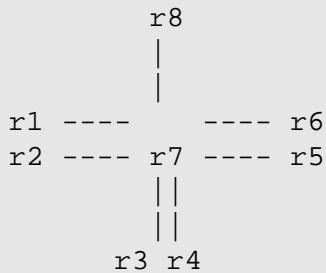
To view the current additional paths, run `net show bgp <network>`. The example output shows an additional path that has been added by the TX node for receiving. Each path has a unique AddPath ID.

```
cumulus@leaf01:~$ net show bgp 10.0.0.12
BGP routing table entry for 10.0.0.12/32
Paths: (2 available, best #1, table Default-IP-Routing-Table)
  Advertised to non peer-group peers:
    spine01(swp51) spine02(swp52)
    65020 65012
      fe80::4638:39ff:fe00:5c from spine01(swp51) (10.0.0.21)
        (fe80::4638:39ff:fe00:5c) (used)
        Origin incomplete, localpref 100, valid, external, multipath,
        bestpath-from-AS 65020, best
        AddPath ID: RX 0, TX 6
        Last update: Wed Nov 16 22:47:00 2016
    65020 65012
      fe80::4638:39ff:fe00:2b from spine02(swp52) (10.0.0.22)
        (fe80::4638:39ff:fe00:2b) (used)
        Origin incomplete, localpref 100, valid, external, multipath
        AddPath ID: RX 0, TX 3
```

Last update: Wed Nov 16 22:47:00 2016

BGP add-path TX

AddPath TX allows BGP to advertise more than just the bestpath for a prefix. Consider the following topology:



In this topology:

- r1 and r2 are in AS 100
- r3 and r4 are in AS 300
- r5 and r6 are in AS 500
- r7 is in AS 700
- r8 is in AS 800
- r7 learns 1.1.1.1/32 from r1, r2, r3, r4, r5, and r6. Among these r7 picks the path from r1 as the bestpath for 1.1.1.1/32

The example below configures the r7 session to advertise the bestpath learned from each AS. In this case, this means a path from AS 100, a path from AS 300, and a path from AS 500. The `net show bgp 1.1.1.1/32` from r7 has "bestpath-from-AS 100" so the user can see what the bestpath is from each AS:

```

cumulus@r7:~$ net add bgp autonomous-system 700
cumulus@r7:~$ net add bgp neighbor 192.0.2.2 addpath-tx-bestpath-per-
AS
  
```

The output below shows the result on r8:

```

cumulus@r8:~$ net show bgp 1.1.1.1/32
BGP routing table entry for 1.1.1.1/32
Paths: (3 available, best #3, table Default-IP-Routing-Table)
  Advertised to non peer-group peers:
    r7(10.7.8.1)
    700 100
      10.7.8.1 from r7(10.7.8.1) (10.0.0.7)
        Origin IGP, localpref 100, valid, external
        Community: 1:1
        AddPath ID: RX 2, TX 4
  
```



```
Last update: Thu Jun 2 00:57:14 2016

700 300
 10.7.8.1 from r7(10.7.8.1) (10.0.0.7)
    Origin IGP, localpref 100, valid, external
    Community: 3:3
    AddPath ID: RX 4, TX 3
    Last update: Thu Jun 2 00:57:14 2016

700 500
 10.7.8.1 from r7(10.7.8.1) (10.0.0.7)
    Origin IGP, localpref 100, valid, external, bestpath-from-AS
700, best
    Community: 5:5
    AddPath ID: RX 6, TX 2
    Last update: Thu Jun 2 00:57:14 2016
```

The example below shows the results if r7 is configured to advertise all paths to r8:

```
cumulus@r7:~$ net add bgp autonomous-system 700
cumulus@r7:~$ net add bgp neighbor 192.0.2.2 addpath-tx-all-paths
```

The output below shows the result on r8:

```
cumulus@r8:~$ net show bgp 1.1.1.1/32
BGP routing table entry for 1.1.1.1/32
Paths: (3 available, best #3, table Default-IP-Routing-Table)
  Advertised to non peer-group peers:
    r7(10.7.8.1)
  700 100
    10.7.8.1 from r7(10.7.8.1) (10.0.0.7)
      Origin IGP, localpref 100, valid, external
      Community: 1:1
      AddPath ID: RX 2, TX 4
      Last update: Thu Jun 2 00:57:14 2016

  700 300
    10.7.8.1 from r7(10.7.8.1) (10.0.0.7)
      Origin IGP, localpref 100, valid, external
      Community: 3:3
      AddPath ID: RX 4, TX 3
      Last update: Thu Jun 2 00:57:14 2016

  700 500
    10.7.8.1 from r7(10.7.8.1) (10.0.0.7)
      Origin IGP, localpref 100, valid, external, bestpath-from-AS
  700, best
      Community: 5:5
      AddPath ID: RX 6, TX 2
```



Fast Convergence Design Considerations

Without getting into the why (see the IETF draft cited in Useful Links below that talks about BGP use within the data center), we strongly recommend the following use of addresses in the design of a BGP-based data center network:

- Use of interface addresses: Set up BGP sessions only using interface-scoped addresses. This allows BGP to react quickly to link failures.
- Use of next-hop-self: Every BGP node says that it knows how to forward traffic to the prefixes it is announcing. This reduces the requirement to announce interface-specific addresses and thereby reduces the size of the forwarding table.

Specifying the Interface Name in the neighbor Command

When you are configuring BGP for the neighbors of a given interface, you can specify the interface name instead of its IP address. All the other `neighbor` command options remain the same.

This is equivalent to BGP peering to the link-local IPv6 address of the neighbor on the given interface. The link-local address is learned via IPv6 neighbor discovery router advertisements.

Consider the following example configuration in the `/etc/quagga/Quagga.conf` file:

```
router bgp 65000
  bgp router-id 10.0.0.1
  neighbor swp1 interface
  neighbor swp1 remote-as internal
  neighbor swp1 next-hop-self
!
  address-family ipv6
  neighbor swp1 activate
  exit-address-family
```

You create the above configuration with the following NCLU commands:

```
cumulus@switch:~$ net add bgp autonomous-system 65000
cumulus@switch:~$ net add bgp router-id 10.0.0.1
cumulus@switch:~$ net add bgp neighbor swp1 interface
cumulus@switch:~$ net add bgp neighbor swp1 remote-as internal
cumulus@switch:~$ net add bgp neighbor swp1 next-hop-self
cumulus@switch:~$ net add bgp ipv6 unicast neighbor swp1 activate
```



By default, Cumulus Linux sends IPv6 neighbor discovery router advertisements. Cumulus Networks recommends you adjust the router advertisement's interval to a shorter value (`net add interface <interface> ipv6 nd ra-interval <interval>`) to address scenarios when nodes come up and miss router advertisement processing to relay the neighbor's link-local address to BGP. The `interval` is measured in seconds and defaults to 600 seconds.



Using Peer Groups to Simplify Configuration

When there are many peers to connect to, the amount of redundant configuration becomes overwhelming. For example, repeating the `activate` and `next-hop-self` commands for even 60 neighbors makes for a very long configuration file. Using `peer-group` addresses this problem.

Instead of specifying properties of each individual peer, Quagga allows for defining one or more peer groups and associating all the attributes common to that peer session to a peer group. A peer needs to be attached to a peer group only once, when it then inherits all address families activated for that peer group.

After doing this, the only task is to associate an IP address with a peer group. Here is an example of defining and using peer groups:

```
cumulus@switch:~$ net add bgp neighbor tier-2 peer-group
cumulus@switch:~$ net add bgp ipv4 unicast
cumulus@switch:~$ net add bgp neighbor tier-2 activate
cumulus@switch:~$ net add bgp neighbor tier-2 next-hop-self
cumulus@switch:~$ net add bgp neighbor 10.0.0.2 peer-group tier-2
cumulus@switch:~$ net add bgp neighbor 192.0.2.2 peer-group tier-2
```



BGP peer-group restrictions have been replaced with update-groups, which dynamically examine all peers, and group them if they have the same outbound policy.

Configuring BGP Dynamic Neighbors

The *BGP dynamic neighbor* feature provides BGP peering to a group of remote neighbors within a specified range of IPv4 or IPv6 addresses for a BGP peer group. You can configure each range as a subnet IP address.

You configure dynamic neighbors using the `bgp listen range <IP address> peer-group <GROUP>` command. Once they are configured, a BGP speaker can listen for and form peer relationships with any neighbor in the IP address range and mapped to a peer group.

```
cumulus@switch:~$ net add bgp autonomous-system 65001
cumulus@switch:~$ net add bgp listen range 10.1.1.0/24 peer-group
SPINE
```

You can limit the number of dynamic peers by specifying that limit in the `bgp listen limit` command:

```
cumulus@switch:~$ net add bgp listen limit 5
```

Collectively, a sample configuration for IPv4 would look like this:

```
cumulus@switch:~$ net add bgp autonomous-system 65001
```

```
cumulus@switch:~$ net add bgp neighbor SPINE peer-group
cumulus@switch:~$ net add bgp neighbor SPINE remote-as 65000
cumulus@switch:~$ net add bgp listen limit 5
cumulus@switch:~$ net add bgp listen range 10.1.1.0/24 peer-group
SPINE
```

These commands produce an IPv4 configuration that looks like this:

```
router bgp 65001
neighbor SPINE peer-group
neighbor SPINE remote-as 65000
bgp listen limit 5
bgp listen range 10.1.1.0/24 peer-group SPINE
```

Configuring BGP Peering Relationships across Switches

A BGP peering relationship is typically initiated with the `neighbor x.x.x.x remote-as [internal|external]` command.

Specifying *internal* signifies an iBGP peering; that is, the neighbor will only create or accept a connection with the specified neighbor if the remote peer AS number matches this BGP's AS number.

Specifying *external* signifies an eBGP peering; that is, the neighbor will only create a connection with the neighbor if the remote peer AS number does **not** match this BGP AS number.

You can make this distinction using the `neighbor` command or the `peer-group` command.

In general, use the following syntax with the `neighbor` command:

```
cumulus@switch:~$ net add bgp neighbor [<IP address>|<BGP
peer>|<interface>] remote-as [<value>|internal|external]
```

Some example configurations follow.

To connect to **the same AS** using the `neighbor` command, modify your configuration similar to the following:

```
cumulus@switch:~$ net add bgp autonomous-system 500
cumulus@switch:~$ net add bgp neighbor 192.168.1.2 remote-as
internal
```

These commands create the following configuration snippet:

```
router bgp 500
neighbor 192.168.1.2 remote-as internal
```



To connect to a **different AS** using the `neighbor` command, modify your configuration similar to the following:

```
cumulus@switch:~$ net add bgp autonomous-system 500
cumulus@switch:~$ net add bgp neighbor 192.168.1.2 remote-as
external
```

These commands create the following configuration snippet:

```
router bgp 500
neighbor 192.168.1.2 remote-as external
```

To connect to **the same AS** using the `peer-group` command, modify your configuration similar to the following:

```
cumulus@switch:~$ net add bgp autonomous-system 500
cumulus@switch:~$ net add bgp neighbor swp1 interface
cumulus@switch:~$ net add bgp neighbor IBGP peer-group
cumulus@switch:~$ net add bgp neighbor IBGP remote-as internal
cumulus@switch:~$ net add bgp neighbor swp1 interface peer-group
IBGP
cumulus@switch:~$ net add bgp neighbor 192.0.2.3 peer-group IBGP
cumulus@switch:~$ net add bgp neighbor 192.0.2.4 peer-group IBGP
```

These commands create the following configuration snippet:

```
router bgp 500
neighbor swp1 interface
neighbor IBGP peer-group
neighbor IBGP remote-as internal
neighbor swp1 peer-group IBGP
neighbor 192.0.2.3 peer-group IBGP
neighbor 192.0.2.4 peer-group IBGP
```

To connect to a **different AS** using the `peer-group` command, modify your configuration similar to the following:

```
cumulus@switch:~$ net add bgp autonomous-system 500
```

```
cumuluss@switch:~$ net add bgp neighbor swp2 interface
cumuluss@switch:~$ net add bgp neighbor EBGP peer-group
cumuluss@switch:~$ net add bgp neighbor EBGP remote-as external
cumuluss@switch:~$ net add bgp neighbor 192.0.2.2 peer-group EBGP
cumuluss@switch:~$ net add bgp neighbor swp2 interface peer-group
EBGP
cumuluss@switch:~$ net add bgp neighbor 192.0.2.4 peer-group EBGP
```

These commands create the following configuration snippet:

```
router bgp 500
neighbor swp2 interface
neighbor EBGP peer-group
neighbor EBGP remote-as external
neighbor 192.0.2.2 peer-group EBGP
neighbor swp2 peer-group EBGP
neighbor 192.0.2.4 peer-group EBGP
```

Configuring MD5-enabled BGP Neighbors

The following sections outline how to configure an MD5-enabled BGP neighbor. Each process assumes that Quagga is used as the routing platform, and consists of two switches (AS 65011 and AS 65020), connected by the link 10.0.0.100/30, with the following configurations:

switch1

```
cumulus@leaf01:~$ net show bgp summary
show bgp ipv4 unicast summary
=====
BGP router identifier 10.0.0.11, local AS number 65011 vrf-id 0
BGP table version 6
RIB entries 11, using 1320 bytes of memory
Peers 2, using 36 KiB of memory
Peer groups 1, using 56 bytes of memory
Neighbor          V      AS MsgRcvd MsgSent     TblVer  InQ OutQ Up
/Down  State/PfxRcd
spine01(swp51)   4 65020    93587    93587        0      0      0
1d02h00m          3
spine02(swp52)   4 65020    93587    93587        0      0      0
1d02h00m          3
Total number of neighbors 2

show bgp ipv6 unicast summary
=====
No IPv6 neighbor is configured
```

**switch2**

```
cumulus@spine01:~$ net show bgp summary
show bgp ipv4 unicast summary
=====
BGP router identifier 10.0.0.21, local AS number 65020 vrf-id 0
BGP table version 5
RIB entries 9, using 1080 bytes of memory
Peers 4, using 73 KiB of memory
Peer groups 1, using 56 bytes of memory
Neighbor          V          AS MsgRcvd MsgSent   TblVer  InQ OutQ Up
/Down  State/PfxRcd
leaf01(swp1)    4 65011      782      782       0     0     0 00:12:54
2
leaf02(swp2)    4 65012      781      781       0     0     0 00:12:53
2
swp3            4 0          0        0       0     0     0 never
Idle
swp4            4 0          0        0       0     0     0 never
Idle
Total number of neighbors 4

show bgp ipv6 unicast summary
=====
No IPv6 neighbor is configured
```

Manually Configuring an MD5-enabled BGP Neighbor

1. SSH into leaf01.
2. Configure the password for the neighbor:

```
cumulus@leaf01:~$ net add bgp neighbor 10.0.0.102 password
mypassword
```

3. Confirm the configuration has been implemented with the `net show bgp summary` command:

```
cumulus@leaf01:~$ net show bgp summary
show bgp ipv4 unicast summary
=====
BGP router identifier 10.0.0.11, local AS number 65011 vrf-id 0
BGP table version 18
RIB entries 11, using 1320 bytes of memory
Peers 2, using 36 KiB of memory
Peer groups 1, using 56 bytes of memory
Neighbor          V          AS MsgRcvd MsgSent   TblVer  InQ OutQ Up
/Down  State/PfxRcd
```

```

spine01(swp51) 4 65020    96144    96146      0   0   0 00:30:
29            3
spine02(swp52) 4 65020    96209    96217      0   0   0
1d02h44m      3
Total number of neighbors 2

show bgp ipv6 unicast summary
=====
No IPv6 neighbor is configured

```

4. SSH into spine01.
5. Configure the password for the neighbor:

```

cumulus@spine01:~$ net add bgp neighbor 10.0.0.101 password
mypassword

```

6. Confirm the configuration has been implemented with the `net show bgp summary` command:

```

cumulus@spine01:~$ net show bgp summary
show bgp ipv4 unicast summary
=====
BGP router identifier 10.0.0.21, local AS number 65020 vrf-id 0
BGP table version 5
RIB entries 9, using 1080 bytes of memory
Peers 4, using 73 KiB of memory
Peer groups 1, using 56 bytes of memory
Neighbor          V          AS MsgRcvd MsgSent     TblVer  InQ OutQ
Up/Down  State/PfxRcd
leaf01(swp1)    4 65011    782    782      0   0   0 00:12:
54            2
leaf02(swp2)    4 65012    781    781      0   0   0 00:12:
53            2
swp3           4   0       0       0       0   0   0
never        Idle
swp4           4   0       0       0       0   0   0
never        Idle
Total number of neighbors 4

show bgp ipv6 unicast summary
=====
No IPv6 neighbor is configured

```

Configuring BGP TTL Security

The steps below cover how to configure BGP ttl security on Cumulus Linux, using a leaf (leaf01), and spine (spine01) for the example output:

1. SSH into leaf01 and configure it for TTL security:



```
cumulus@leaf01:~$ net add bgp autonomous-system 65000
cumulus@leaf01:~$ net add bgp neighbor [spine01-IP] ttl-security
hops [value]
```

2. SSH into spine01 and configure it for TTL security:

```
cumulus@spine01:~$ net add bgp autonomous-system 65001
cumulus@spine01:~$ net add bgp neighbor [leaf01-IP] ttl-security
hops [value]
```

BGP TTL security is now configured. To review the resulting configuration, run the `show ip bgp neighbor` command.

Example net show bgp neighbor output

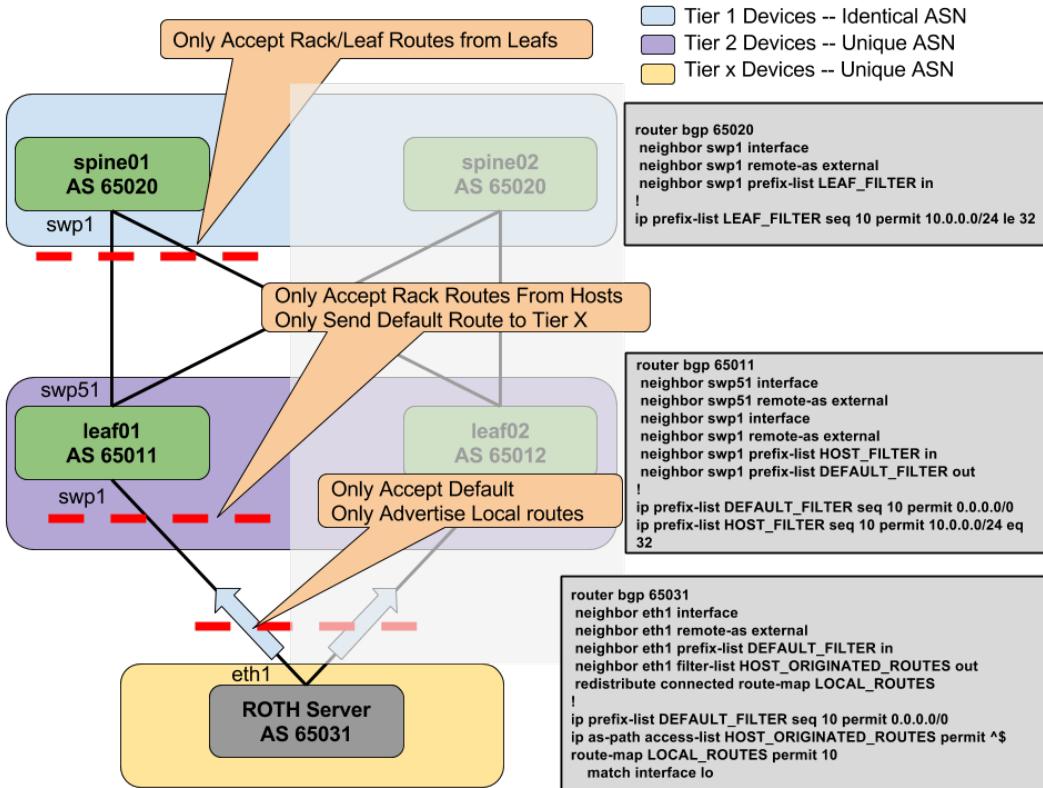
```
cumulus@spine01:~$ net show bgp neighbor
BGP neighbor on swp1: fe80::4638:39ff:fe00:5b, remote AS 65011, local
AS 65020, external link
Hostname: leaf01
Member of peer-group fabric for session parameters
  BGP version 4, remote router ID 0.0.0.0
  BGP state = Connect
  Last read 19:47:43, Last write 20:27:13
  Hold time is 3, keepalive interval is 1 seconds
  Configured hold time is 3, keepalive interval is 1 seconds
Message statistics:
  Inq depth is 0
  Outq depth is 0
          Sent        Rcvd
  Opens:          2          2
  Notifications: 0          2
  Updates:        12         14
  Keepalives:     5224       5222
  Route Refresh: 0          0
  Capability:    0          0
  Total:          5238       5240
Minimum time between advertisement runs is 0 seconds
For address family: IPv4 Unicast
  fabric peer-group member
  Not part of any update group
  Community attribute sent to this neighbor(both)
  Inbound path policy configured
  Outbound path policy configured
  Incoming update prefix filter list is *dc-spine
  Outgoing update prefix filter list is *dc-spine
  0 accepted prefixes
  Connections established 2; dropped 2
```

```
Last reset 19:47:42, due to NOTIFICATION received (Cease/Other Configuration Change)
BGP Connect Retry Timer in Seconds: 3
Next connect timer due in 0 seconds
Read thread: on Write thread: on
```

Configuration Tips

BGP Advertisement Best Practices

Limiting the exchange of routing information at various parts in the network is a best practice you should follow. The following image illustrates one way you can do so in a typical Clos architecture:



Utilizing Multiple Routing Tables and Forwarding

You can run multiple routing tables (one for in-band/data plane traffic and one for out-of-band /management plane traffic) on the same switch using [management VRF](#) (see page 593) (multiple routing tables and forwarding).

Using BGP Community Lists

You can use [community lists](#) to define a BGP community to tag one or more routes. You can then use the communities to apply route policy on either egress or ingress.

The BGP community list can be either *standard* or *expanded*. The standard BGP community list is a pair of values (such as 100:100) that can be tagged on a specific prefix and advertised to other neighbors or applied on route ingress. Alternately, it can be one of four BGP default communities:



- *internet*: a BGP community that matches all routes
- *local-AS*: a BGP community that restrict routes to your confederation's sub-AS
- *no-advertise*: a BGP community that isn't advertised to anyone
- *no-export*: a BGP community that isn't advertised to the eBGP peer

An expanded BGP community list takes a regular expression of communities matches the listed communities.

When the neighbor receives the prefix, it examines the community value and takes action accordingly, such as permitting or denying the community member in the routing policy.

Here's an example of standard community list filter:

```
cumulus@switch:~$ net add routing community-list standard COMMUNITY1  
permit 100:100
```

You can apply the community list to a route map to define the routing policy:

```
cumulus@switch:~$ net add bgp table-map ROUTE-MAP1
```

Additional Default Settings

Other default settings not discussed in detail in this chapter include the following; they're all enabled by default:

- `bgp deterministic-med`, which ensures path ordering no longer impacts bestpath selection.
- `bgp show-hostname`, which displays the hostname in show command output.
- `bgp network import-check`, which enables the advertising of the BGP network in IGP.

Configuring BGP Neighbor maximum-prefixes

The maximum number of route announcements, or prefixes, allowed by a BGP neighbor can be configured using the `maximum-prefixes` command in the CLI. Replace the `PEER` input with the relevant peer, and replace `NUMBER` with the maximum number of prefixes desired:

```
quagga(config)# neighbor PEER maximum-prefix NUMBER
```

Troubleshooting BGP

The most common starting point for troubleshooting BGP is to view the summary of neighbors connected to and some information about these connections. A sample output of this command is as follows:

```
cumulus@switch:~$ net show bgp summary  
show bgp ipv4 unicast summary  
=====  
BGP router identifier 10.0.0.11, local AS number 65011 vrf-id 0
```

```
BGP table version 8
RIB entries 11, using 1320 bytes of memory
Peers 2, using 36 KiB of memory
Peer groups 1, using 56 bytes of memory
Neighbor          V          AS MsgRcvd MsgSent     TblVer  InQ OutQ Up
/Down  State/PfxRcd
spine01(swp51)   4 65020      549      551        0       0       0 00:09:
03              3
spine02(swp52)   4 65020      548      550        0       0       0 00:09:
02              3
Total number of neighbors 2

show bgp ipv6 unicast summary
=====
No IPv6 neighbor is configured
```



You can determine whether the sessions above are iBGP or eBGP sessions by looking at the ASNs.

It is also useful to view the routing table as defined by BGP:

```
cumulus@switch:~$ net show bgp ipv4
ERROR: Command not found
Use 'net help KEYWORD(s)' to list all options that use KEYWORD(s)
cumulus@leaf01:~$ net show bgp ipv4
    unicast : add help text
cumulus@leaf01:~$ net show bgp ipv4 unicast
BGP table version is 8, local router ID is 10.0.0.11
Status codes: s suppressed, d damped, h history, * valid, > best, =
multipath,
                i internal, r RIB-failure, S Stale, R Removed
Origin codes: i - IGP, e - EGP, ? - incomplete
      Network          Next Hop            Metric LocPrf Weight Path
*> 10.0.0.11/32      0.0.0.0                  0        32768 ?
*= 10.0.0.12/32      swp52                   0  65020 65012 ?
*>
*> 10.0.0.21/32      swp51                   0  65020 ?
*> 10.0.0.22/32      swp52                   0  65020 ?
*> 172.16.1.0/24      0.0.0.0                  0        32768 i
*= 172.16.2.0/24      swp52                   0  65020 65012 i
*>
Total number of prefixes 6
```

A more detailed breakdown of a specific neighbor can be obtained using `net show bgp neighbor <neighbor>`:

```
cumulus@switch:~$ net show bgp neighbor swp51
```



```
BGP neighbor on swp51: fe80::4638:39ff:fe00:5c, remote AS 65020,
local AS 65011, external link
Hostname: spine01
Member of peer-group fabric for session parameters
BGP version 4, remote router ID 10.0.0.21
BGP state = Established, up for 00:11:30
Last read 00:00:00, Last write 00:11:26
Hold time is 3, keepalive interval is 1 seconds
Configured hold time is 3, keepalive interval is 1 seconds
Neighbor capabilities:
  4 Byte AS: advertised and received
  AddPath:
    IPv4 Unicast: RX advertised IPv4 Unicast and received
    Extended nexthop: advertised and received
    Address families by peer:
      IPv4 Unicast
      Route refresh: advertised and received(old & new)
      Address family IPv4 Unicast: advertised and received
      Hostname Capability: advertised and received
      Graceful Restart Capabilty: advertised and received
      Remote Restart timer is 120 seconds
    Address families by peer:
      none
Graceful restart informations:
  End-of-RIB send: IPv4 Unicast
  End-of-RIB received: IPv4 Unicast
Message statistics:
  Inq depth is 0
  Outq depth is 0
          Sent        Rcvd
  Opens:           1           1
  Notifications:  0           0
  Updates:         7           6
  Keepalives:     690         689
  Route Refresh:  0           0
  Capability:    0           0
  Total:          698         696
Minimum time between advertisement runs is 0 seconds
For address family: IPv4 Unicast
  fabric peer-group member
  Update group 1, subgroup 1
  Packet Queue length 0
  Community attribute sent to this neighbor(both)
  Inbound path policy configured
  Outbound path policy configured
  Incoming update prefix filter list is *dc-leaf-in
  Outgoing update prefix filter list is *dc-leaf-out
  3 accepted prefixes
  Connections established 1; dropped 0
  Last reset never
  Local host: fe80::4638:39ff:fe00:5b, Local port: 48424
  Foreign host: fe80::4638:39ff:fe00:5c, Foreign port: 179
```

```
Nexthop: 10.0.0.11
Nexthop global: fe80::4638:39ff:fe00:5b
Nexthop local: fe80::4638:39ff:fe00:5b
BGP connection: shared network
BGP Connect Retry Timer in Seconds: 3
Estimated round trip time: 3 ms
Read thread: on Write thread: off
```

To see the details of a specific route such as from whom it was received, to whom it was sent, and so forth, use the `net show bgp <ip address/prefix>` command:

```
cumulus@leaf01:~$ net show bgp 10.0.0.11/32
BGP routing table entry for 10.0.0.11/32
Paths: (1 available, best #1, table Default-IP-Routing-Table)
  Advertised to non peer-group peers:
    spine01(swp51) spine02(swp52)
  Local
    0.0.0.0 from 0.0.0.0 (10.0.0.11)
      Origin incomplete, metric 0, localpref 100, weight 32768,
      valid, sourced, bestpath-from-AS Local, best
      AddPath ID: RX 0, TX 9
      Last update: Fri Nov 18 01:48:17 2016
```

This shows that the routing table prefix seen by BGP is 10.0.0.11/32, that this route was advertised to two neighbors, and that it was not heard by any neighbors.

Debugging Tip: Logging Neighbor State Changes

It is very useful to log the changes that a neighbor goes through to troubleshoot any issues associated with that neighbor. This is done using the `log-neighbor-changes` command, which is enabled by default.

The output is sent to the specified log file, usually `/var/log/quagga/bgpd.log`, and looks like this:

```
2016/07/08 10:12:06.572827 BGP: %NOTIFICATION: sent to neighbor
10.0.0.2 6/3 (Cease/Peer Unconfigured) 0 bytes
2016/07/08 10:12:06.572954 BGP: Notification sent to neighbor
10.0.0.2: type 6/3
2016/07/08 10:12:16.682071 BGP: %ADJCHANGE: neighbor 192.0.2.2 Up
2016/07/08 10:12:16.682660 BGP: %ADJCHANGE: neighbor 10.0.0.2 Up
```

Troubleshooting Link-local Addresses

To verify that quagga learned the neighboring link-local IPv6 address via the IPv6 neighbor discovery router advertisements on a given interface, use the `show interface <if-name>` command. If `ipv6_nd suppress-ra` isn't enabled on both ends of the interface, then `Neighbor address(s):` should have the other end's link-local address. That is the address that BGP would use when BGP is enabled on that interface.



IPv6 route advertisements (RAs) are automatically enabled on an interface with IPv6 addresses, so the step `no ipv6 nd suppress-ra` is no longer needed for BGP unnumbered. The timer interval for RAs remains 600s, which may need to be adjusted to bring up peers quickly.

Use vtysh to verify the configuration:

```
cumulus@switch:~$ sudo vtysh

Hello, this is Quagga (version 0.99.23.1+cl3u2).
Copyright 1996-2005 Kunihiro Ishiguro, et al.

R7# show interface swp1
Interface swp1 is up, line protocol is up
  Link ups:      0    last: (never)
  Link downs:   0    last: (never)
  PTM status: disabled
  vrf: Default-IP-Routing-Table
  index 4 metric 0 mtu 1500
  flags: <UP,BROADCAST,RUNNING,MULTICAST>
  HWaddr: 44:38:39:00:00:5c
  inet6 fe80::4638:39ff:fe00:5c/64
    ND advertised reachable time is 0 milliseconds
    ND advertised retransmit interval is 0 milliseconds
    ND router advertisements are sent every 10 seconds
    ND router advertisements lifetime tracks ra-interval
    ND router advertisement default router preference is medium
    Hosts use stateless autoconfig for addresses.
  Neighbor address(s):
    inet6 fe80::4638:39ff:fe00:5b/128
```

Instead of the IPv6 address, the peering interface name is displayed in the `show ip bgp summary` command and wherever else applicable:

```
cumulus@switch:~$ net show bgp summary
BGP router identifier 10.0.0.21, local AS number 65020 vrf-id 0
BGP table version 15
RIB entries 17, using 2040 bytes of memory
Peers 6, using 97 KiB of memory
Peer groups 1, using 56 bytes of memory

Neighbor          V     AS MsgRcvd MsgSent TblVer  InQ OutQ Up/Down
State/PfxRcd
leaf01(swp1)      4 65011    2834    2843      0      0      0 02:21:
35              2
leaf02(swp2)      4 65012    2834    2844      0      0      0 02:21:
36              2
leaf03(swp3)      4 65013    2834    2843      0      0      0 02:21:
35              2
```

```

leaf04(swp4)    4 65014    2834    2844      0    0    0 02:21:
36              2
edge01(swp29)   4 65051    8509    8505      0    0    0 02:21:
37              3
edge01(swp30)   4 65051    8506    8503      0    0    0 02:21:
35              3

```

Total number of neighbors 6

Most of the show commands can take the interface name instead of the IP address, if that level of specificity is needed:

```

cumulus@leaf01:~$ net show bgp neighbor
  fabric  : BGP neighbor or peer-group
  swp51   : BGP neighbor or peer-group
  swp52   : BGP neighbor or peer-group
<ENTER>

```

```

cumulus@leaf01:~$ net show bgp neighbor swp51
BGP neighbor on swp51: fe80::4638:39ff:fe00:5c, remote AS 65020,
local AS 65011, external link
Hostname: spine01
Member of peer-group fabric for session parameters
  BGP version 4, remote router ID 0.0.0.0
  BGP state = Connect
  Last read 20:16:21, Last write 20:55:51
  Hold time is 30, keepalive interval is 10 seconds
  Configured hold time is 30, keepalive interval is 10 seconds
Message statistics:
  Inq depth is 0
  Outq depth is 0
          Sent        Rcvd
  Opens:           1           1
  Notifications:  1           0
  Updates:         7           6
  Keepalives:     2374       2373
  Route Refresh:  0           0
  Capability:    0           0
  Total:          2383       2380
Minimum time between advertisement runs is 5 seconds
For address family: IPv4 Unicast
  fabric peer-group member
  Not part of any update group
  Community attribute sent to this neighbor(both)
  Inbound path policy configured
  Outbound path policy configured
  Incoming update prefix filter list is *dc-leaf-in
  Outgoing update prefix filter list is *dc-leaf-out

```



```
0 accepted prefixes
Connections established 1; dropped 1
Last reset 20:16:20, due to NOTIFICATION sent (Cease/Other
Configuration Change)
BGP Connect Retry Timer in Seconds: 3
Next connect timer due in 1 seconds
Read thread: on Write thread: on
```

Enabling Read-only Mode

You can enable read-only mode for when the BGP process restarts or when the BGP process is cleared using `clear ip bgp *`. When enabled, read-only mode begins as soon as the first peer reaches its *established* state and a timer for `<max-delay>` seconds is started.

While in read-only mode, BGP doesn't run best-path or generate any updates to its peers. This mode continues until:

- All the configured peers, except the shutdown peers, have sent an explicit EOR (End-Of-RIB) or an implicit EOR. The first keep-alive after BGP has reached the established state is considered an implicit EOR. If the `<establish-wait>` option is specified, then BGP will wait for peers to reach the established state from the start of the `update-delay` until the `<establish-wait>` period is over; that is, the minimum set of established peers for which EOR is expected would be peers established during the `establish-wait` window, not necessarily all the configured neighbors.
- The `max-delay` period is over.

Upon reaching either of these two conditions, BGP resumes the decision process and generates updates to its peers.

To enable read-only mode:

```
cumulus@switch:$ net add bgp update-delay <max-delay in 0-3600
seconds> [<establish-wait in 1-3600 seconds>]
```

The default `<max-delay>` is 0 — the feature is off by default.

Use output from `show ip bgp summary` for information about the state of the update delay.

This feature can be useful in reducing CPU/network usage as BGP restarts/clears. It's particularly useful in topologies where BGP learns a prefix from many peers. Intermediate best paths are possible for the same prefix as peers get established and start receiving updates at different times. This feature is also valuable if the network has a high number of such prefixes.

Applying a Route Map for Route Updates

There are two ways you can apply `route maps` for BGP:

- By filtering routes from BGP into Zebra
- By filtering routes from Zebra into the Linux kernel



Filtering Routes from BGP into Zebra

For the first way, you can apply a route map on route updates from BGP to Zebra. All the applicable match operations are allowed, such as match on prefix, next-hop, communities, and so forth. Set operations for this attach-point are limited to metric and next-hop only. Any operation of this feature does not affect BGPs internal RIB.

Both IPv4 and IPv6 address families are supported. Route maps work on multi-paths as well. However, the metric setting is based on the best path only.

To apply a route map to filter route updates from BGP into Zebra:

```
cumulus@switch:$ net add bgp table-map <route-map-name>
```

Filtering Routes from Zebra into the Linux Kernel

To apply a route map to filter route updates from Zebra into the Linux kernel:

```
cumulus@switch:$ net add routing protocol bgp route-map <route-map-name>
```

Protocol Tuning

Converging Quickly On Link Failures

In the Clos topology, we recommend that you only use interface addresses to set up peering sessions. This means that when the link fails, the BGP session is torn down immediately, triggering route updates to propagate through the network quickly. This requires the following commands be enabled for all links: `link-detect` and `ttl-security hops <hops>`. `ttl-security hops` specifies how many hops away the neighbor is. For example, in a Clos topology, every peer is at most 1 hop away.



See Caveats and Errata below for information regarding `ttl-security hops`.

Here is an example:

```
cumulus@switch:~$ net add bgp neighbor 10.0.0.2 ttl-security hops 1
```

Converging Quickly On Soft Failures

It is possible that the link is up, but the neighboring BGP process is hung or has crashed. If a BGP process crashes, Quagga's `watchquagga` daemon, which monitors the various `quagga` daemons, will attempt to restart it. If the process is also hung, `watchquagga` will attempt to restart the process. BGP itself has a keepalive timer that is exchanged between neighbors. By default, this keepalive timer is set to 3 seconds. This time can be increased to a higher number, which decreases CPU load, especially in the presence of a

lot of neighbors. `keepalive-time` is the periodicity with which the keepalive message is sent. `hold-time` specifies how many keepalive messages can be lost before the connection is considered invalid. It is usually set to 3 times the keepalive time, so it defaults to 9 seconds. Here is an example of changing these timers:

```
cumulus@switch:~$ net add bgp neighbor swp51 timers 10 30
```

The following display snippet shows that the default values have been modified for this neighbor:

```
cumulus@switch:~$ net show bgp neighbor swp51
BGP neighbor on swp51: fe80::4638:39ff:fe00:5c, remote AS 65020,
local AS 65011, external link
Hostname: spine01
Member of peer-group fabric for session parameters
  BGP version 4, remote router ID 0.0.0.0
  BGP state = Connect
  Last read 00:00:13, Last write 00:39:43
  Hold time is 30, keepalive interval is 10 seconds
  Configured hold time is 30, keepalive interval is 10 seconds
  ...
  ...
```

Reconnecting Quickly

A BGP process attempts to connect to a peer after a failure (or on startup) every `connect-time` seconds. By default, this is 10 seconds. To modify this value, use:

```
cumulus@switch:~$ net add bgp neighbor swp51 timers connect 30
```

This command has to be specified per each neighbor, peer-group doesn't support this option in quagga.

Advertisement Interval

BGP by default chooses stability over fast convergence. This is very useful when routing for the Internet. For example, unlike link-state protocols, BGP typically waits for a duration of `advertisement-interval` seconds between sending consecutive updates to a neighbor. This ensures that an unstable neighbor flapping routes won't be propagated throughout the network. By default, this is set to 0 seconds for both eBGP and iBGP sessions, which allows for very fast convergence. You can modify this as follows:

```
cumulus@switch:~$ net add bgp neighbor swp51 advertisement-interval 5
```

The following output shows the modified value:

```
cumulus@switch:~$ net show bgp neighbor swp51
BGP neighbor on swp51: fe80::4638:39ff:fe00:5c, remote AS 65020,
local AS 65011, external link
```



```
Hostname: spine01
Member of peer-group fabric for session parameters
BGP version 4, remote router ID 0.0.0.0
BGP state = Connect
Last read 00:04:37, Last write 00:44:07
Hold time is 30, keepalive interval is 10 seconds
Configured hold time is 30, keepalive interval is 10 seconds
Message statistics:
  Inq depth is 0
  Outq depth is 0
          Sent      Rcvd
Opens:           1           1
Notifications:   1           0
Updates:         7           6
Keepalives:     2374        2373
Route Refresh:   0           0
Capability:     0           0
Total:          2383        2380
Minimum time between advertisement runs is 5 seconds
...
```



This command is not supported with peer-groups.

See this [IETF draft](#) for more details on the use of this value.

Caveats and Errata

ttl-security Issue

Enabling `ttl-security` does not cause the hardware to be programmed with the relevant information. This means that frames will come up to the CPU and be dropped there. It is recommended that you use the `net add acl` command to explicitly add the relevant entry to hardware.

For example, you can configure a file, like `/etc/cumulus/acl/policy.d/01control_plane_bgp.rules`, with a rule like this for TTL:

```
INGRESS_INTF = swp1
INGRESS_CHAIN = INPUT, FORWARD

[iptables]
-A $INGRESS_CHAIN --in-interface $INGRESS_INTF -p tcp --dport bgp
-m ttl --ttl 255 POLICE --set-mode pkt --set-rate 2000 --set-burst
1000
-A $INGRESS_CHAIN --in-interface $INGRESS_INTF -p tcp --dport bgp DROP
```



For more information about ACLs, see [Netfilter \(ACLs\)](#) (see page 121).



Related Information

- Bidirectional forwarding detection (see page 549) (BFD) and BGP
- Wikipedia entry for BGP (includes list of useful RFCs)
- Quagga online documentation for BGP (may not be up to date)
- IETF draft discussing BGP use within data centers

Bidirectional Forwarding Detection - BFD

Bidirectional Forwarding Detection (BFD) provides low overhead and rapid detection of failures in the paths between two network devices. It provides a unified mechanism for link detection over all media and protocol layers. Use BFD to detect failures for IPv4 and IPv6 single or multihop paths between any two network devices, including unidirectional path failure detection.



Cumulus Linux does not support demand mode in BFD.

Contents

This chapter covers ...

- Using BFD Multihop Routed Paths (see page 549)
- BFD Parameters (see page 549)
- Configuring BFD (see page 550)
 - BFD in BGP (see page 550)
 - BFD in OSPF (see page 552)
 - OSPF Show Commands (see page 552)
 - Scripts (see page 554)
 - Echo Function (see page 554)
- Troubleshooting BFD (see page 555)

Using BFD Multihop Routed Paths

BFD multihop sessions are built over arbitrary paths between two systems, which results in some complexity that does not exist for single hop sessions. Here are some best practices for using multihop paths:

- **Spoofing:** To avoid spoofing with multihop paths, configure `max_hop_cnt` (maximum hop count) for each peer, which limits the number of hops for a BFD session. All BFD packets exceeding the max hop count will be dropped.
- **Demultiplexing:** Since multihop BFD sessions can take arbitrary paths, demultiplex the initial BFD packet based on the source/destination IP address pair. Use Quagga, which monitors connectivity to the peer, to determine the source/destination IP address pairs.

Multihop BFD sessions are supported for both IPv4 and IPv6 peers. See below for more details.



BFD Parameters

You can configure the following BFD parameters for both IPv4 and IPv6 sessions:

- The required minimum interval between the received BFD control packets.
- The minimum interval for transmitting BFD control packets.
- The detection time multiplier.

Configuring BFD

You configure BFD one of two ways: by specifying the configuration in the [PTM topology.dot file \(see page 257\)](#), or using [Quagga \(see page 488\)](#). However, the topology file has some limitations:

- The `topology.dot` file supports creating BFD IPv4 and IPv6 single hop sessions only; you cannot specify IPv4 or IPv6 multihop sessions in the topology file.
- The topology file supports BFD sessions for only link-local IPv6 peers; BFD sessions for global IPv6 peers discovered on the link will not be created.



You cannot specify BFD multihop sessions in the `topology.dot` file since you cannot specify the source and destination IP address pairs in that file. Use [Quagga \(see page 489\)](#) to configure multihop sessions.

The Quagga CLI can track IPv4 and IPv6 peer connectivity — both single hop and multihop, and both link-local IPv6 peers and global IPv6 peers — using BFD sessions without needing the `topology.dot` file. Use Quagga to register multihop peers with PTM and BFD as well as for monitoring the connectivity to the remote BGP multihop peer. Quagga can dynamically register and unregister both IPv4 and IPv6 peers with BFD when the BFD-enabled peer connectivity is established or de-established, respectively. Also, you can configure BFD parameters for each BGP or OSPF peer using Quagga.



The BFD parameter configured in the topology file is given higher precedence over the client-configured BFD parameters for a BFD session that has been created by both topology file and client (Quagga).



BFD requires an IP address for any interface on which it is configured. The neighbor IP address for a single hop BFD session must be in the ARP table before BFD can start sending control packets.

BFD in BGP

For Quagga when using **BGP**, neighbors are registered and de-registered with [PTM \(see page 257\)](#) dynamically when you enable BFD in BGP:

```
cumulus@switch:~$ net add bgp neighbor <bgp peer> bfd
```

You can configure BFD parameters for each BGP neighbor. For example:



BFD in BGP

```
cumulus@switch:~$ net add bgp neighbor <bgp peer> bfd 4 400 400
```

To see neighbor information in BGP, including BFD status, run `net show bgp neighbor <interface>`.

Show BGP Neighbor

```
cumulus@switch:~$ net show bgp neighbor swp51
BGP neighbor on swp51: fe80::4638:39ff:fe00:5c, remote AS 65020,
local AS 65011, external link
Hostname: spine01
Member of peer-group fabric for session parameters
  BGP version 4, remote router ID 10.0.0.21
  BGP state = Established, up for 03:09:17
  Last read 00:00:01, Last write 03:09:13
  Hold time is 3, keepalive interval is 1 seconds
  Configured hold time is 3, keepalive interval is 1 seconds
Neighbor capabilities:
  4 Byte AS: advertised and received
  AddPath:
    IPv4 Unicast: RX advertised IPv4 Unicast and received
    Extended nexthop: advertised and received
    Address families by peer:
      IPv4 Unicast
      Route refresh: advertised and received(old & new)
      Address family IPv4 Unicast: advertised and received
      Hostname Capability: advertised and received
      Graceful Restart Capabilty: advertised and received
      Remote Restart timer is 120 seconds
    Address families by peer:
      none
  Graceful restart informations:
    End-of-RIB send: IPv4 Unicast
    End-of-RIB received: IPv4 Unicast
  Message statistics:
    Inq depth is 0
    Outq depth is 0
          Sent          Rcvd
    Opens:            1            1
    Notifications:  0            0
    Updates:         5            6
    Keepalives:     11352        11351
    Route Refresh:   0            0
    Capability:     0            0
    Total:          11358        11358
  Minimum time between advertisement runs is 0 seconds
  For address family: IPv4 Unicast
    fabric peer-group member
    Update group 1, subgroup 1
```

```

Packet Queue length 0
Community attribute sent to this neighbor(both)
Inbound path policy configured
Outbound path policy configured
Incoming update prefix filter list is *dc-leaf-in
Outgoing update prefix filter list is *dc-leaf-out
3 accepted prefixes
Connections established 1; dropped 0
Last reset never
Local host: fe80::4638:39ff:fe00:5b, Local port: 48390
Foreign host: fe80::4638:39ff:fe00:5c, Foreign port: 179
Nexthop: 10.0.0.11
Nexthop global: fe80::4638:39ff:fe00:5b
Nexthop local: fe80::4638:39ff:fe00:5b
BGP connection: shared network
BGP Connect Retry Timer in Seconds: 3
Read thread: on Write thread: off

```



The `add neighbor` commands recognize the neighbor IP or the interface name (`swp#`).

BFD in OSPF

For Quagga using **OSFP**, neighbors are registered and de-registered dynamically with [PTM \(see page 257\)](#) when you enable or disable BFD in OSPF. A neighbor is registered with BFD when two-way adjacency is established and deregistered when adjacency goes down if the BFD is enabled on the interface. The BFD configuration is per interface and any IPv4 and IPv6 neighbors discovered on that interface inherit the configuration.

BFD in OSPF

```
cumulus@switch:~$ net add interface swp1 ospf6 bfd 5 500 500
```

OSPF Show Commands

The BFD lines at the end of each code block shows the corresponding IPv6 or IPv4 OSPF interface or neighbor information.

Show IPv6 OSPF Interface

```
cumulus@switch:~$ net show ospf6 interface swp2s0
swp2s0 is up, type BROADCAST
  Interface ID: 4
  Internet Address:
    inet : 11.0.0.21/30
    inet6: fe80::4638:39ff:fe00:6c8e/64
  Instance ID 0, Interface MTU 1500 (autodetect: 1500)
```



```
MTU mismatch detection: enabled
Area ID 0.0.0.0, Cost 10
State PointToPoint, Transmit Delay 1 sec, Priority 1
Timer intervals configured:
  Hello 10, Dead 40, Retransmit 5
DR: 0.0.0.0 BDR: 0.0.0.0
Number of I/F scoped LSAs is 2
  0 Pending LSAs for LSUpdate in Time 00:00:00 [thread off]
  0 Pending LSAs for LSAck in Time 00:00:00 [thread off]
BFD: Detect Mul: 3, Min Rx interval: 300, Min Tx interval: 300
```

Show IPv6 OSPF Neighbor

```
cumulus@switch:~$ net show ospf6 neighbor detail
Neighbor 0.0.0.4%swp2s0
  Area 0.0.0 via interface swp2s0 (ifindex 4)
    His IfIndex: 3 Link-local address: fe80::202:ff:fe00:a
    State Full for a duration of 02:32:33
    His choice of DR/BDR 0.0.0.0/0.0.0.0, Priority 1
    DbDesc status: Slave SeqNum: 0x76000000
    Summary-List: 0 LSAs
    Request-List: 0 LSAs
    Retrans-List: 0 LSAs
    0 Pending LSAs for DbDesc in Time 00:00:00 [thread off]
    0 Pending LSAs for LSReq in Time 00:00:00 [thread off]
    0 Pending LSAs for LSUpdate in Time 00:00:00 [thread off]
    0 Pending LSAs for LSAck in Time 00:00:00 [thread off]
  BFD: Type: single hop
    Detect Mul: 3, Min Rx interval: 300, Min Tx interval: 300
    Status: Up, Last update: 0:00:00:20
```

Show IPv4 OSPF Interface

```
cumulus@switch:~$ net show ospf interface swp2s0
swp2s0 is up
  ifindex 4, MTU 1500 bytes, BW 0 Kbit <UP,BROADCAST,RUNNING,
  MULTICAST>
    Internet Address 11.0.0.21/30, Area 0.0.0.0
    MTU mismatch detection:enabled
    Router ID 0.0.0.3, Network Type POINTOPOINT, Cost: 10
    Transmit Delay is 1 sec, State Point-To-Point, Priority 1
    No designated router on this network
    No backup designated router on this network
    Multicast group memberships: OSPFAllRouters
    Timer intervals configured, Hello 10s, Dead 40s, Wait 40s,
    Retransmit 5
      Hello due in 7.056s
    Neighbor Count is 1, Adjacent neighbor count is 1
```

```
BFD: Detect Mul: 5, Min Rx interval: 500, Min Tx interval: 500
```

Show IPv4 OSPF Neighbor

```
cumulus@switch:~$ net show ospf neighbor detail
Neighbor 0.0.0.4, interface address 11.0.0.22
  In the area 0.0.0.0 via interface swp2s0
  Neighbor priority is 1, State is Full, 5 state changes
  Most recent state change statistics:
    Progressive change 3h59m04s ago
  DR is 0.0.0.0, BDR is 0.0.0.0
  Options 2 *|-|-|---|E|*
  Dead timer due in 38.501s
  Database Summary List 0
  Link State Request List 0
  Link State Retransmission List 0
  Thread Inactivity Timer on
  Thread Database Description Retransmission off
  Thread Link State Request Retransmission on
  Thread Link State Update Retransmission on
  BFD: Type: single hop
    Detect Mul: 5, Min Rx interval: 500, Min Tx interval: 500
    Status: Down, Last update: 0:00:01:29
```

Scripts

`ptmd` executes scripts at `/etc/ptm.d/bfd-sess-down` and `/etc/ptm.d/bfd-sess-up` for when BFD sessions go down or up, running `bfd-sess-down` when a BFD session goes down and running `bfd-sess-up` when a BFD session goes up.

You should modify these default scripts as needed.

Echo Function

Cumulus Linux supports the *echo function* for IPv4 single hops only, and with the a synchronous operating mode only (Cumulus Linux does not support demand mode).

You use the echo function primarily to test the forwarding path on a remote system. To enable the echo function, set `echoSupport` to 1 in the topology file.

Once the echo packets are looped by the remote system, the BFD control packets can be sent at a much lower rate. You configure this lower rate by setting the `slowMinTx` parameter in the topology file to a non-zero value of milliseconds.

You can use more aggressive detection times for echo packets since the round-trip time is reduced because they are accessing the forwarding path. You configure the detection interval by setting the `echoMinRx` parameter in the topology file to a non-zero value of milliseconds; the minimum setting is 50 milliseconds. Once configured, BFD control packets are sent out at this required minimum echo Rx interval. This indicates to the peer that the local system can loop back the echo packets. Echo packets are transmitted if the peer supports receiving echo packets.

About the Echo Packet

BFD echo packets are encapsulated into UDP packets over destination and source UDP port number 3785. The BFD echo packet format is vendor-specific and has not been defined in the RFC. BFD echo packets that originate from Cumulus Linux are 8 bytes long and have the following format:

0	1	2	3
Version	Length	Reserved	
My Discriminator			

Where:

- **Version** is the version of the BFD echo packet.
- **Length** is the length of the BFD echo packet.
- **My Discriminator** is a non-zero value that uniquely identifies a BFD session on the transmitting side. When the originating node receives the packet after being looped back by the receiving system, this value uniquely identifies the BFD session.

Transmitting and Receiving Echo Packets

BFD echo packets are transmitted for a BFD session only when the peer has advertised a non-zero value for the required minimum echo Rx interval (the `echoMinRx` setting) in the BFD control packet when the BFD session starts. The transmit rate of the echo packets is based on the peer advertised echo receive value in the control packet.

BFD echo packets are looped back to the originating node for a BFD session only if locally the `echoMinRx` and `echoSupport` are configured to a non-zero values.

Using Echo Function Parameters

You configure the echo function by setting the following parameters in the topology file at the global, template and port level:

- **echoSupport:** Enables and disables echo mode. Set to 1 to enable the echo function. It defaults to 0 (disable).
- **echoMinRx:** The minimum interval between echo packets the local system is capable of receiving. This is advertised in the BFD control packet. When the echo function is enabled, it defaults to 50. If you disable the echo function, this parameter is automatically set to 0, which indicates the port or the node cannot process or receive echo packets.
- **slowMinTx:** The minimum interval between transmitting BFD control packets when the echo packets are being exchanged.

Troubleshooting BFD

To troubleshoot BFD, use `ptmctl -b`. For more information, see [Prescriptive Topology Manager - PTM](#) (see page 257).



Equal Cost Multipath Load Sharing - Hardware ECMP

Cumulus Linux supports hardware-based [equal cost multipath](#) (ECMP) load sharing. ECMP is enabled by default in Cumulus Linux. Load sharing occurs automatically for all routes with multiple next hops installed. ECMP load sharing supports both IPv4 and IPv6 routes.

Contents

This chapter covers ...

- Understanding Equal Cost Routing (see page 556)
- Understanding ECMP Hashing (see page 557)
 - Using cl-ecmpcalc to Determine the Hash Result (see page 557)
 - cl-ecmpcalc Limitations (see page 558)
 - ECMP Hash Buckets (see page 558)
- Resilient Hashing (see page 560)
 - Resilient Hash Buckets (see page 561)
 - Removing Next Hops (see page 561)
 - Adding Next Hops (see page 563)
 - Configuring Resilient Hashing (see page 563)
- Caveats and Errata (see page 564)

Understanding Equal Cost Routing

ECMP operates only on equal cost routes in the Linux routing table.

In this example, the 10.1.1.0/24 route has two possible next hops that have been installed in the routing table:

```
$ ip route show 10.1.1.0/24
10.1.1.0/24 proto zebra metric 20
nexthop via 192.168.1.1 dev swp1 weight 1 onlink
nexthop via 192.168.2.1 dev swp2 weight 1 onlink
```

For routes to be considered equal they must:

- Originate from the same routing protocol. Routes from different sources are not considered equal. For example, a static route and an OSPF route are not considered for ECMP load sharing.
- Have equal cost. If two routes from the same protocol are unequal, only the best route is installed in the routing table.



As of Cumulus Linux 3.0, the BGP `maximum-paths` setting is enabled, so multiple routes are installed by default. See the [ECMP section \(see page 519\)](#) of the BGP chapter for more information.



Understanding ECMP Hashing

Once multiple routes are installed in the routing table, a hash is used to determine which path a packet follows.

Cumulus Linux hashes on the following fields:

- IP protocol
- Ingress interface
- Source IPv4 or IPv6 address
- Destination IPv4 or IPv6 address

For TCP/UDP frames, Cumulus Linux also hashes on:

- Source port
- Destination port

ECMP Hash Fields					
Source IP	Destination IP	Layer 4 Protocol	Source Port	Destination Port	Payload

To prevent out of order packets, ECMP hashing is done on a per-flow basis, which means that all packets with the same source and destination IP addresses and the same source and destination ports always hash to the same next hop. ECMP hashing does not keep a record of flow states.

ECMP hashing does not keep a record of packets that have hashed to each next hop and does not guarantee that traffic sent to each next hop is equal.

Using `cl-ecmpcalc` to Determine the Hash Result

Since the hash is deterministic and always provides the same result for the same input, you can query the hardware and determine the hash result of a given input. This is useful when determining exactly which path a flow takes through a network.

On Cumulus Linux, use the `cl-ecmpcalc` command to determine a hardware hash result.

In order to use `cl-ecmpcalc`, all fields that are used in the hash must be provided. This includes ingress interface, layer 3 source IP, layer 3 destination IP, layer 4 source port and layer 4 destination port.

```
$ sudo cl-ecmpcalc -i swp1 -s 10.0.0.1 -d 10.0.0.1 -p tcp --sport 2000
0 --dport 80
ecmpcalc: will query hardware
swp3
```

If any field is omitted, `cl-ecmpcalc` fails.

```
$ sudo cl-ecmpcalc -i swp1 -s 10.0.0.1 -d 10.0.0.1 -p tcp
ecmpcalc: will query hardware
usage: cl-ecmpcalc [-h] [-v] [-p PROTOCOL] [-s SRC] [--sport SPORT] [-
d DST]
                  [--dport DPORT] [--vid VID] [-i IN_INTERFACE]
```

```

[--sportid SPORTID] [--smodid SMODID] [-o
OUT_INTERFACE]
[--dportid DPORTID] [--dmodid DMODID] [--hardware]
[--nohardware] [-hs HASHSEED]
[-hf HASHFIELDS [HASHFIELDS ...]]
[--hashfunction {crc16-ccitt,crc16-bisync}] [-e
EGRESS]
[-c MCOUNT]
cl-ecmpcalc: error: --sport and --dport required for TCP and UDP
frames

```

cl-ecmpcalc Limitations

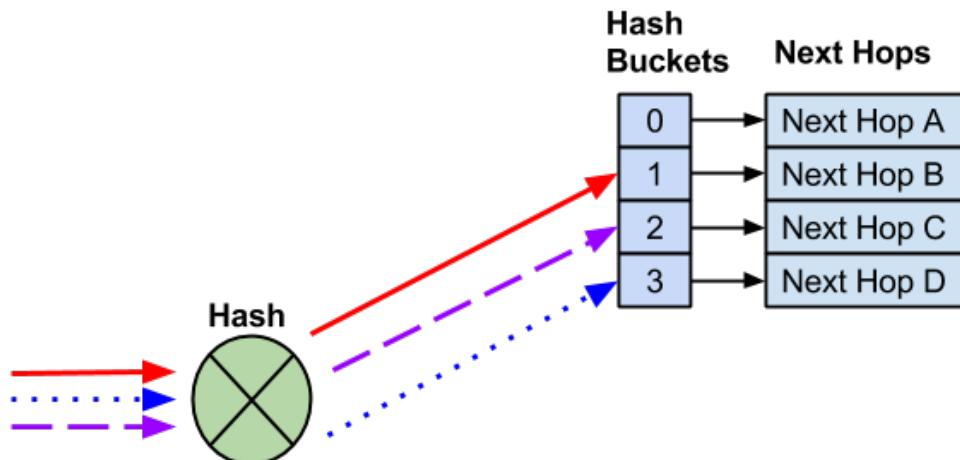
`cl-ecmpcalc` can only take input interfaces that can be converted to a single physical port in the port tab file, like the physical switch ports (swp). Virtual interfaces like bridges, bonds, and subinterfaces are not supported.

`cl-ecmpcalc` is supported only on switches with the Spectrum, Tomahawk, Trident II+ and Trident II chipsets.

ECMP Hash Buckets

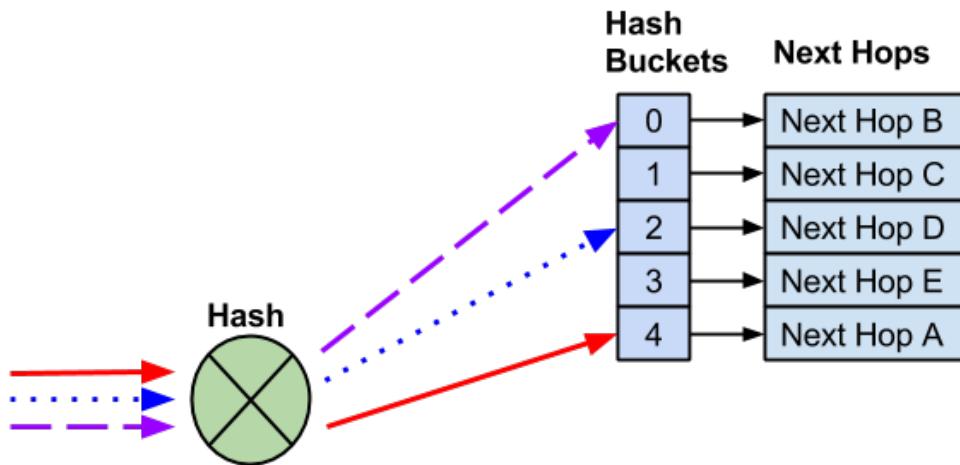
When multiple routes are installed in the routing table, each route is assigned to an ECMP *bucket*. When the ECMP hash is executed the result of the hash determines which bucket gets used.

In the following example, 4 next hops exist. Three different flows are hashed to different hash buckets. Each next hop is assigned to a unique hash bucket.



Adding a Next Hop

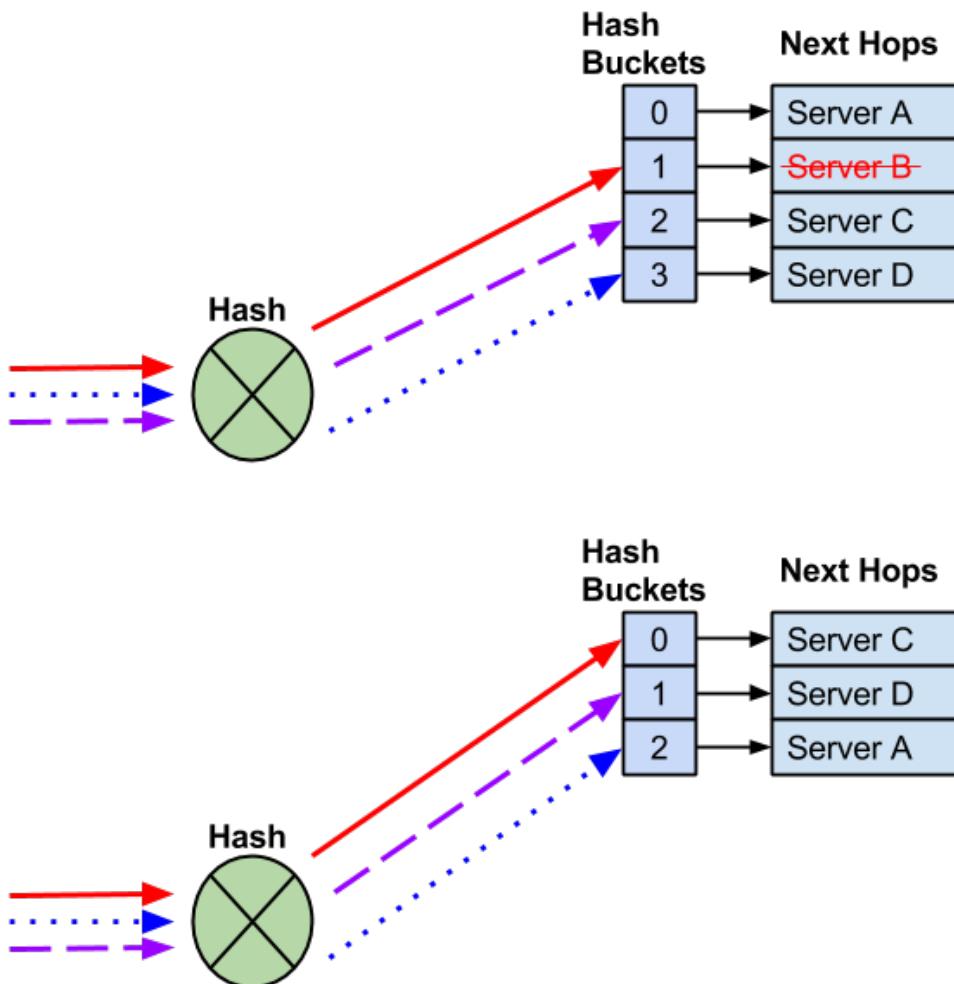
When a next hop is added, a new hash bucket is created. The assignment of next hops to hash buckets, as well as the hash result, may change when additional next hops are added.



A new next hop is added and a new hash bucket is created. As a result, the hash and hash bucket assignment changed, causing the existing flows to be sent to different next hops.

Removing a Next Hop

When a next hop is removed, the remaining hash bucket assignments may change, again, potentially changing the next hop selected for an existing flow.



A next hop fails and the next hop and hash bucket are removed. The remaining next hops may be reassigned.

In most cases, the modification of hash buckets has no impact on traffic flows as traffic is being forward to a single end host. In deployments where multiple end hosts are using the same IP address (anycast), *resilient hashing* must be used.

Resilient Hashing

In Cumulus Linux when a next hop fails or is removed from an ECMP pool, the hashing or hash bucket assignment can change. For deployments where there is a need for flows to always use the same next hop, like TCP anycast deployments, this can create session failures.

The ECMP hash performed with resilient hashing is exactly the same as the default hashing mode. Only the method in which next hops are assigned to hash buckets differs.

Resilient hashing supports both IPv4 and IPv6 routes.

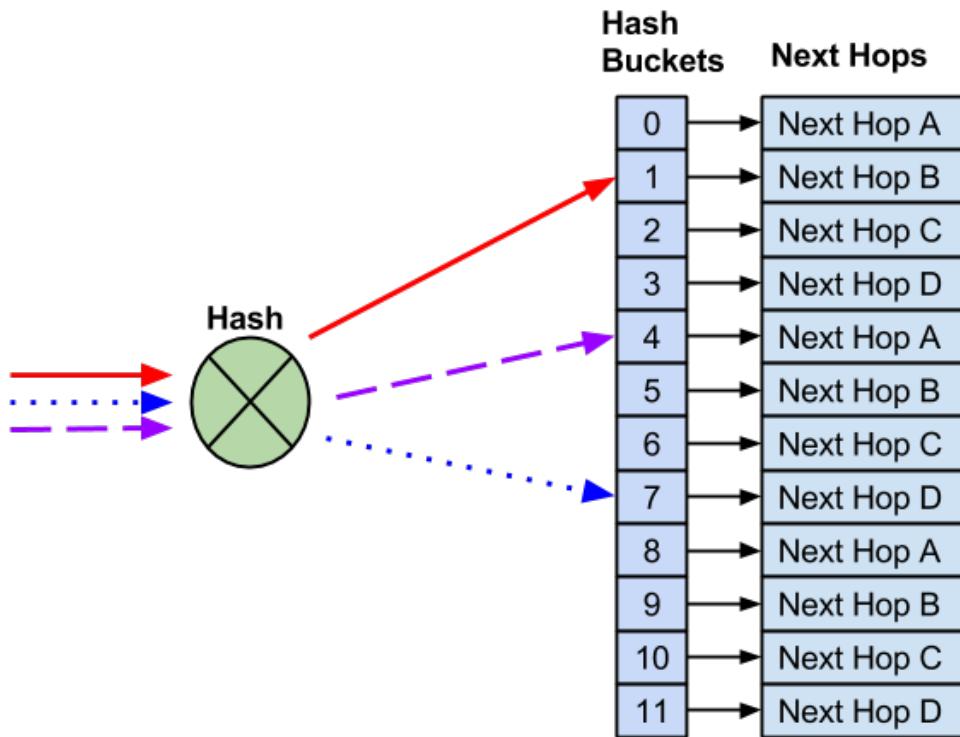
Resilient hashing is not enabled by default. See below for steps on configuring it.



Resilient hashing prevents disruptions when new next hops are removed. It does not prevent disruption when next hops are added.

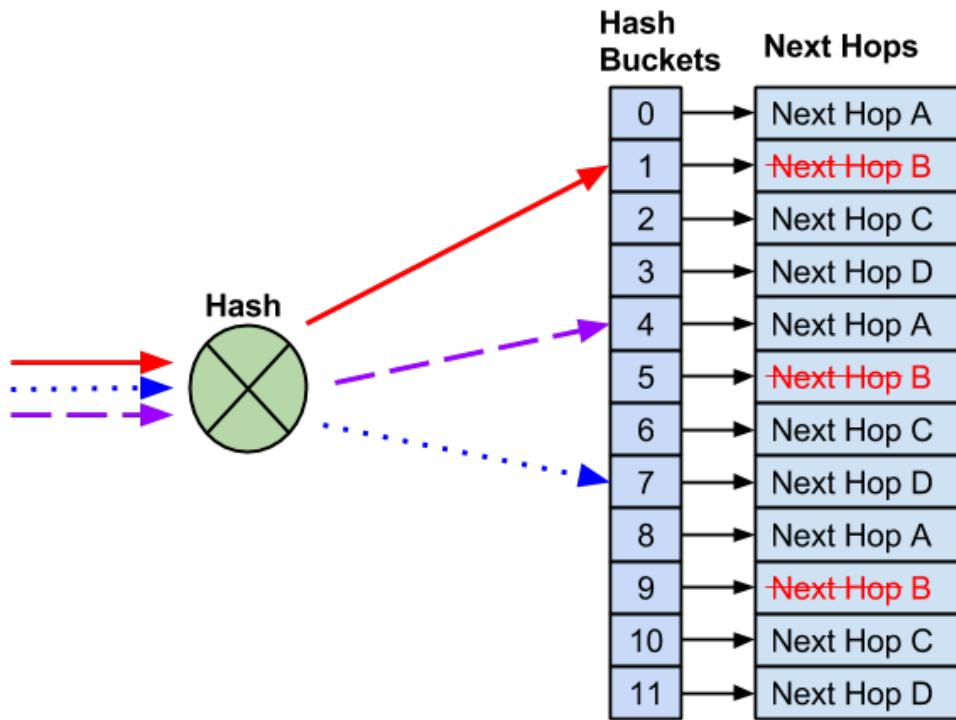
Resilient Hash Buckets

When resilient hashing is configured, a fixed number of buckets are defined. Next hops are then assigned in round robin fashion to each of those buckets. In this example, 12 buckets are created and four next hops are assigned.

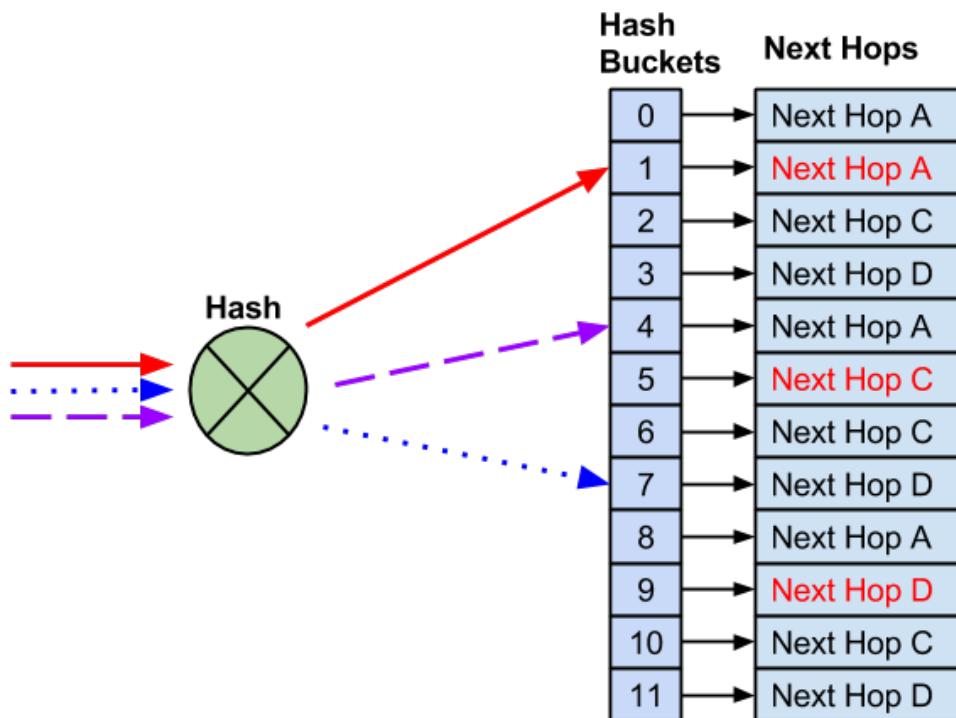


Removing Next Hops

Unlike default ECMP hashing, when a next hop needs to be removed, the number of hash buckets does not change.



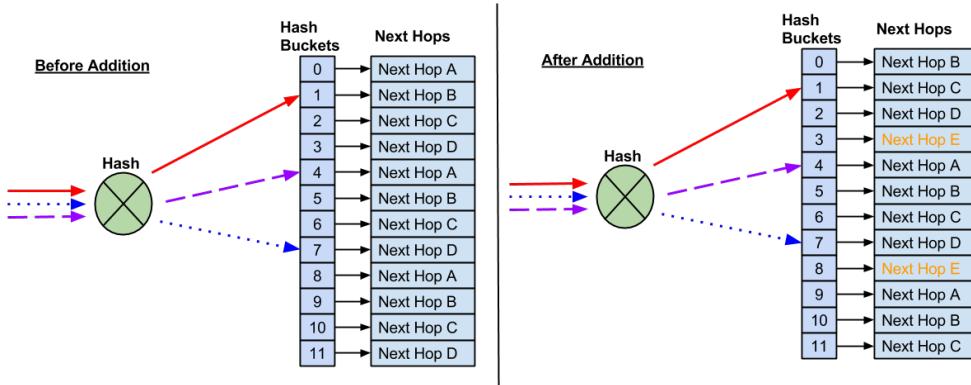
With 12 buckets assigned and four next hops, instead of reducing the number of buckets — which would impact flows to known good hosts — the remaining next hops replace the failed next hop.



After the failed next hop is removed, the remaining next hops are installed as replacements. This prevents impact to any flows that hash to working next hops.

Adding Next Hops

Resilient hashing does not prevent possible impact to existing flows when new next hops are added. Due to the fact there are a fixed number of buckets, a new next hop requires reassigning next hops to buckets.



As a result, some flows may hash to new next hops, which can impact anycast deployments.

Configuring Resilient Hashing

Resilient hashing is not enabled by default. When resilient hashing is enabled, 65,536 buckets are created to be shared among all ECMP groups. An ECMP group is a list of unique next hops that are referenced by multiple ECMP routes.



An ECMP route counts as a single route with multiple next hops. The following example is considered to be a single ECMP route:

```
$ ip route show 10.1.1.0/24
10.1.1.0/24 proto zebra metric 20
nexthop via 192.168.1.1 dev swp1 weight 1 onlink
nexthop via 192.168.2.1 dev swp2 weight 1 onlink
```

All ECMP routes must use the same number of buckets (the number of buckets cannot be configured per ECMP route).

The number of buckets can be configured as 64, 128, 256, 512 or 1024; the default is 128:

Number of Hash Buckets	Number of Supported ECMP Groups
64	1024
128	512



Number of Hash Buckets	Number of Supported ECMP Groups
256	256
512	128
1024	64

A larger number of ECMP buckets reduces the impact on adding new next hops to an ECMP route. However, the system supports fewer ECMP routes. If the maximum number of ECMP routes have been installed, new ECMP routes log an error and are not installed.

To enable resilient hashing, edit `/etc/cumulus/datapath/traffic.conf`:

1. Enable resilient hashing:

```
# Enable resilient hashing
resilient_hash_enable = TRUE
```

2. **(Optional)** Edit the number of hash buckets:

```
# Resilient hashing flowset entries per ECMP group
# Valid values - 64, 128, 256, 512, 1024
resilient_hash_entries_ecmp = 256
```

3. Restart (see page 173) the `switchd` service:

```
cumulus@switch:~$ sudo systemctl restart switchd.service
```

Caveats and Errata

Resilient hashing is supported only on switches with the [Tomahawk](#), [Trident II+](#) and [Trident II](#) chipsets. You can run `netshow system` to determine the chipset.

Redistribute Neighbor

Redistribute neighbor provides a mechanism for IP subnets to span racks without forcing the end hosts to run a routing protocol.

The fundamental premise behind redistribute neighbor is to announce individual host /32 routes in the routed fabric. Other hosts on the fabric can then use this new path to access the hosts in the fabric. If multiple equal-cost paths (ECMP) are available, traffic can load balance across the available paths natively.

The challenge is to accurately compile and update this list of reachable hosts or neighbors. Luckily, existing commonly-deployed protocols are available to solve this problem. Hosts use [ARP](#) to resolve MAC addresses when sending to an IPv4 address. A host then builds an ARP cache table of known MAC addresses: IPv4 tuples as they receive or respond to ARP requests.



In the case of a leaf switch, where the default gateway is deployed for hosts within the rack, the ARP cache table contains a list of all hosts that have ARP'd for their default gateway. In many scenarios, this table contains all the layer 3 information that's needed. This is where redistribute neighbor comes in, as it is a mechanism of formatting and syncing this table into the routing protocol.

Contents

This chapter covers ...

- Availability (see page 565)
- Target Use Cases and Best Practices (see page 565)
- How It Works (see page 566)
- Configuration Steps (see page 566)
 - Configuring the Leaf(s) (see page 566)
 - Configuring the Host(s) (see page 569)
- Known Limitations (see page 570)
 - TCAM Route Scale (see page 570)
 - Possible Uneven Traffic Distribution (see page 570)
 - Silent Hosts Never Receive Traffic (see page 571)
 - Support for IPv4 Only (see page 571)
 - VRFs Are not Supported (see page 571)
- Troubleshooting (see page 571)
 - Verification (see page 573)

Availability

Redistribute neighbor is distributed as `python-rdnbrd`.

Target Use Cases and Best Practices

Redistribute neighbor was created with these use cases in mind:

- Virtualized clusters
- Hosts with service IP addresses that migrate between racks
- Hosts that are dual connected to two leaf nodes without using proprietary protocols such as [MLAG \(see page 300\)](#)
- Anycast services needing dynamic advertisement from multiple hosts

Cumulus Networks recommends following these guidelines with redistribute neighbor:

- Use a single logical connection from each host to each leaf.
- A host can connect to one or more leafs. Each leaf advertises the /32 it sees in its neighbor table.
- A host-bound bridge/VLAN should be local to each switch only.
- Leaf switches with redistribute neighbor enabled should be directly connected to the hosts.
- IP addressing must be non-overlapping, as the host IPs are directly advertised into the routed fabric.

- Run redistribute neighbor on Linux-based hosts primarily; other host operating systems may work, but Cumulus Networks has not actively tested any at this stage.

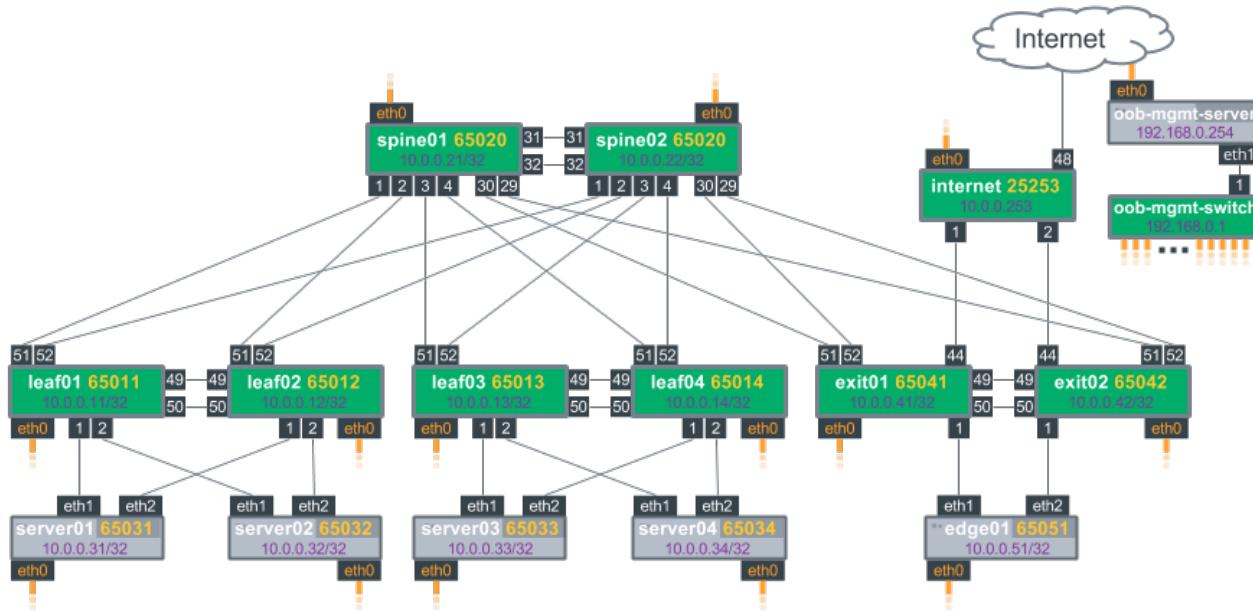
How It Works

Redistribute neighbor works as follows:

- The leaf/ToR switches learn about connected hosts when the host sends an ARP request or ARP reply.
- An entry for the host is added to the kernel neighbor table of each leaf switch.
- The redistribute neighbor daemon, `rdbnbd`, monitors the kernel neighbor table and creates a /32 route for each neighbor entry. This /32 route is created in kernel table 10.
- Quagga is configured to import routes from kernel table 10.
- A route-map is used to control which routes from table 10 are imported.
- In Quagga these routes are imported as *table* routes.
- BGP, OSPF and so forth are then configured to redistribute the table 10 routes.

Configuration Steps

The following configuration steps are based on the [reference topology](#) set forth by Cumulus Networks. Here is a diagram of the topology:



Configuring the Leaf(s)

The following steps demonstrate how to configure leaf01, but the same steps can be applied to any of the leafs.

- Configure the host facing ports, using the same IP address on both host-facing interfaces as well as a /32 prefix. In this case, swp1 and swp2 are configured as they are the ports facing server01 and server02:



```
cumulus@leaf01:~$ net add loopback lo address 10.0.0.11/32
cumulus@leaf01:~$ net add interface swp1 address 10.0.0.11/32
cumulus@leaf01:~$ net add interface swp2 address 10.0.0.11/32
```

The commands produce the following configuration in the /etc/network/interfaces file:

```
auto lo
iface lo inet loopback
    address 10.0.0.11/32

auto swp1
iface swp1
    address 10.0.0.11/32

auto swp2
iface swp2
    address 10.0.0.11/32
```

2. Enable the daemon so it starts at bootup:

```
cumulus@leaf01:~$ sudo systemctl enable rdnbrd.service
```

3. Start the daemon:

```
cumulus@leaf01:~$ sudo systemctl restart rdnbrd.service
```

4. Configure routing:

- Add the table as routes into the local routing table using the Quagga vtysh shell:

```
cumulus@leaf01:~$ sudo vtysh
leaf01(config)# conf t
leaf01(config)# ip import-table 10
```

- Define a route-map that matches on the host-facing interfaces:

```
leaf01(config)# route-map REDIST_NEIGHBOR permit 10
leaf01(config-route-map)# match interface swp1
leaf01(config-route-map)# route-map REDIST_NEIGHBOR permit 20
leaf01(config-route-map)# match interface swp2
```

- Apply that route-map to routes imported into *table*:



```
leaf01(config)# ip protocol table route-map REDIST_NEIGHBOR
```

- d. Redistribute the imported *table* routes in into the appropriate routing protocol.

BGP:

```
leaf01(config)# router bgp 65001
leaf01(config-router)# address-family ipv4 unicast
leaf01(config-router-af)# redistribute table 10
leaf01(config-router-af)# end
```

OSPF:

```
leaf01(config)# router ospf
leaf01(config-router)# redistribute table 10
```

- e. Save the configuration and exit vtysh.

```
leaf01(config)# exit
leaf01# write mem
leaf01# exit
cumulus@leaf01:~$
```

Here are the contents of /etc/quagga/Quagga.conf from the reference topology:

```
cumulus@leaf01$ cat /etc/quagga/Quagga.conf
!
interface swp51
  no ipv6 nd suppress-ra
  ipv6 nd ra-interval 3
!
interface swp52
  no ipv6 nd suppress-ra
  ipv6 nd ra-interval 3
!
ip import-table 10 distance 19
route-map rdarp permit 1
match interface swp2
route-map rdarp permit 2
match interface swp1
!
ip protocol table route-map rdarp
router bgp 65011
  bgp router-id 10.0.0.11
  network 10.0.0.11/32
  bgp bestpath as-path multipath-relax
  bgp bestpath compare-routerid
```



```
neighbor fabric peer-group
neighbor fabric description Internal Fabric Network
neighbor fabric capability extended-nexthop
neighbor fabric advertisement-interval 0
neighbor fabric timers 1 3
neighbor fabric timers connect 3
neighbor fabric remote-as external
neighbor swp51 interface v6only
neighbor swp51 peer-group fabric
neighbor swp52 interface v6only
neighbor swp52 peer-group fabric
redistribute table 10

!
address-family ipv6 unicast
neighbor fabric activate
neighbor swp51 activate
neighbor swp52 activate
exit-address-family
!
```

Configuring the Host(s)

There are a few possible host configurations that range in complexity. This document only covers the basic use case: dual-connected Linux hosts with static IP addresses assigned.

Additional host configurations will be covered in future separate knowledge base articles.

Configuring a Dual-connected Host

Configure a host with the same /32 IP address on its loopback (lo) and uplinks (in this example, eth1 and eth2). This is done so both leaf switches advertise the same /32 regardless of the interface. Cumulus Linux relies on [ECMP \(see page 556\)](#) to load balance across the interfaces southbound, and an equal cost static route (see the configuration below) for load balancing northbound.

The loopback hosts the primary service IP address(es) and to which you can bind services.

Configure the loopback and physical interfaces. Referring back to the topology diagram, server01 is connected to leaf01 via eth1 and to leaf02 via eth2. You should note:

- The loopback IP is assigned to lo, eth1 and eth2.
- The post-up ARPing is used to force the host to ARP as soon as its interface comes up. This allows the leaf to learn about the host as soon as possible.
- The post-up ip route replace is used to install a default route via one or both leaf nodes if both swp1 and swp2 are up.

```
cumulus@server01$ cat /etc/network/interfaces
# The loopback network interface
auto lo
iface lo inet loopback
auto lo:1
iface lo:1
    address 10.1.0.101/32
```



```
auto eth1
iface eth1
    address 10.1.0.101/32
    post-up for i in {1..3}; do arping -q -c 1 -w 0 -i eth1
10.0.0.11; sleep 1; done
    post-up ip route add 0.0.0.0/0 nexthop via 10.0.0.11 dev eth1
onlink nexthop via 10.0.0.12 dev eth2 onlink || true

auto eth2
iface eth2
    address 10.1.0.101/32
    post-up for i in {1..3}; do arping -q -c 1 -w 0 -i eth2
10.0.0.12; sleep 1; done
    post-up ip route add 0.0.0.0/0 nexthop via 10.0.0.11 dev eth1
onlink nexthop via 10.0.0.12 dev eth2 onlink || true
```

Installing `ifplugd`

Additionally, install and use `ifplugd`. `ifplugd` modifies the behavior of the Linux routing table when an interface undergoes a link transition (carrier up/down). The Linux kernel by default leaves routes up even when the physical interface is unavailable (NO-CARRIER).

Install `ifplugd` on the host and modify the settings in `/etc/default/ifplugd`:

```
cumulus@server01:~$ sudo apt-get update
cumulus@server01:~$ sudo apt-get install ifplugd
```

Edit `/etc/default/ifplugd` as follows, where `eth1` and `eth2` are the interface names that your host uses to connect to the leaves.

```
cumulus@server01$ cat /etc/default/ifplugd
INTERFACES="eth1 eth2"
HOTPLUG_INTERFACES=""
ARGS="-q -f -u10 -d10 -w -I"
SUSPEND_ACTION="stop"
```

For full instructions on installing `ifplugd` on Ubuntu, [follow this guide](#).

Known Limitations

TCAM Route Scale

This feature adds each ARP entry as a /32 host route into the routing table of all switches within a summarization domain. Take care to keep the number of hosts minus fabric routes under the TCAM size of the switch. Review the [Cumulus Networks datasheets](#) for up to date scalability limits of your chosen hardware platforms. If in doubt, contact Cumulus Networks support or your Cumulus Networks CSE; they will be happy to help.



Possible Uneven Traffic Distribution

Linux uses source L3 addresses only to do load balancing on most older distributions.

Silent Hosts Never Receive Traffic

Freshly provisioned hosts that have never sent traffic may not ARP for their default gateways. The post-up ARPing in /etc/network/interfaces on the host should take care of this. If the host does not ARP, then rdnbrd on the leaf cannot learn about the host.

Support for IPv4 Only

This release of redistribute neighbor supports IPv4 only.

VRFs Are not Supported

This release of redistribute neighbor does not support VRFs (see page 574).

Troubleshooting

- **How do I determine if rdnbrd (the redistribute neighbor daemon) is running?**

Use `systemctl` to check:

```
cumulus@leaf01$ systemctl status rdnbrd.service
* rdnbrd.service - Cumulus Linux Redistribute Neighbor Service
  Loaded: loaded (/lib/systemd/system/rdnbrd.service; enabled)
  Active: active (running) since Wed 2016-05-04 18:29:03 UTC; 1h
  13min ago
    Main PID: 1501 (python)
      CGroup: /system.slice/rdnbrd.service
             `--1501 /usr/bin/python /usr/sbin/rdnbrd -d
```

- **How do I change rdnbrd's default configuration?**

By editing `/etc/rdnbrd.conf` then running `systemctl restart rdnbrd.service`:

```
cumulus@leaf01$ cat /etc/rdnbrd.conf
# syslog logging level CRITICAL, ERROR, WARNING, INFO, or DEBUG
loglevel = INFO

# TX an ARP request to known hosts every keepalive seconds
keepalive = 1

# If a host does not send an ARP reply for holdtime consider the
host down
holdtime = 3

# Install /32 routes for each host into this table
route_table = 10
```

```

# Uncomment to enable ARP debugs on specific interfaces.
# Note that ARP debugs can be very chatty.
# debug_arp = swp1 swp2 swp3 br1
# If we already know the MAC for a host, unicast the ARP
request. This is
# unusual for ARP (why ARP if you know the destination MAC) but
we will be
# using ARP as a keepalive mechanism and do not want to
broadcast so many ARPs
# if we do not have to. If a host cannot handle a unicasted ARP
request, set
# the following option to False.
#
# Unicasting ARP requests is common practice (in some scenarios)
for other
# networking operating systems so it is unlikely that you will
need to set
# this to False.
unicast_arp_requests = True
cumulus@leaf01:~$ sudo systemctl restart rdnbrd.service

```

- **What is table 10? Why was table 10 chosen?**

The Linux kernel supports multiple routing tables and has the ability to utilize 0 through 255 as table IDs. However, tables 0, 253, 254 and 255 are reserved, and 1 is usually the first one utilized, so rdnbrd only allows you to specify 2-252. The number 10 was chosen for no particular reason. Feel free to set it to any value between 2-252. You can see all the tables specified here:

```

cumulus@switch$ cat /etc/iproute2/rt_tables
#
# reserved values
#
255 local
254 main
253 default
0 unspec
#
# local
#
#1 inr.ruhep

```

Read more information on [Linux route tables](#), or you can read the [Ubuntu man pages for ip route](#).

- **How do I determine that the /32 redistribute neighbor routes are being advertised to my neighbor?**

For BGP, check the advertised routes to the neighbor.

```

cumulus@leaf01:~$ sudo vtysh
Hello, this is Quagga (version 0.99.23.1+c13u2).
Copyright 1996-2005 Kunihiro Ishiguro, et al.

```



```
leaf01# show ip bgp neighbor swp51 advertised-routes
BGP table version is 5, local router ID is 10.0.0.11
Status codes: s suppressed, d damped, h history, * valid, >
best, = multipath,
          i internal, r RIB-failure, S Stale, R Removed
Origin codes: i - IGP, e - EGP, ? - incomplete

      Network          Next Hop            Metric LocPrf Weight Path
*-> 10.0.0.11/32    0.0.0.0                  0        32768 i
*-> 10.0.0.12/32    ::                      0
65020 65012 i
*-> 10.0.0.21/32    ::                      0
65020 i
*-> 10.0.0.22/32    ::                      0
65020 i

Total number of prefixes 4
```

Verification

The following workflow can be used to verify that the kernel routing table is being correctly populated, and that routes are being correctly imported/advertised:

1. Verify that ARP neighbour entries are being populated into the Kernel routing table 10.

```
cumulus@switch:~$ ip route show table 10
10.0.1.101 dev swp1 scope link
```

If these routes are not being generated, verify the following:

- That the `rdnbrd` daemon is running
- Check `/etc/rdnbrd.conf` to verify the correct table number is used

2. Verify that routes are being imported into Quagga from the kernel routing table 10.

```
cumulus@switch:~$ sudo vtysh
Hello, this is Quagga (version 0.99.23.1+cl3u2).
Copyright 1996-2005 Kunihiro Ishiguro, et al.

switch# show ip route table
Codes: K - kernel route, C - connected, S - static, R - RIP,
       O - OSPF, I - IS-IS, B - BGP, A - Babel, T - Table,
       > - selected route, * - FIB route
T[10]>* 10.0.1.101/32 [19/0] is directly connected, swp1, 01:25:
29
```

Both the `>` and `*` should be present so that table 10 routes are installed as preferred into the routing table. If the routes are not being installed, verify the following:



- The imported distance of the locally imported kernel routes using the `ip import 10 distance x` command, where X is **not** less than the administrative distance of the routing protocol. If the distance is too low, routes learned from the protocol may overwrite the locally imported routes.
 - The routes are in the kernel routing table.
3. Confirm that routes are in the BGP/OSPF database and being advertised.

```
switch# show ip bgp
```

Virtual Routing and Forwarding - VRF

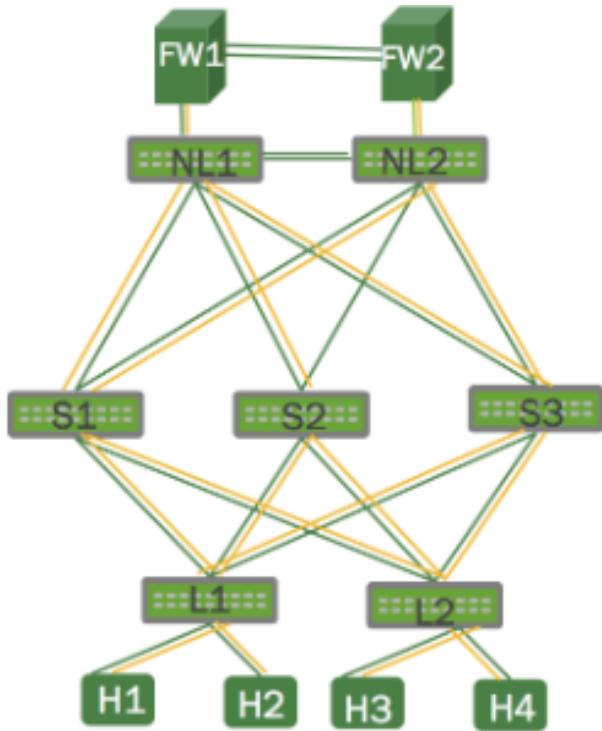
Cumulus Linux provides *virtual routing and forwarding* (VRF) to allow for the presence of multiple independent routing tables working simultaneously on the same router or switch. This permits multiple network paths without the need for multiple switches. Think of this feature as VLAN for layer 3, but unlike VLANs, there is no field in the IP header carrying it. Other implementations call this feature *VRF-Lite*.

The primary use cases for VRF in a data center are similar to VLANs at layer 2: using common physical infrastructure to carry multiple isolated traffic streams for multi-tenant environments, where these streams are allowed to cross over only at configured boundary points, typically firewalls or IDS. You can also use it to burst traffic from private clouds to enterprise networks where the burst point is at layer 3. Or you can use it in an OpenStack deployment.

VRF is fully supported in the Linux kernel, so it has the following characteristics:

- The VRF is presented as a layer 3 master network device with its own associated routing table.
- The layer 3 interfaces (VLAN interfaces, bonds, switch virtual interfaces/SVIs) associated with the VRF are enslaved to that VRF; IP rules direct FIB (forwarding information base) lookups to the routing table for the VRF device.
- The VRF device can have its own IP address, known as a *VRF-local loopback*.
- Applications can use existing interfaces to operate in a VRF context — by binding sockets to the VRF device or passing the `ifindex` using `cmsg`.
- Listen sockets used by services are VRF-global by default unless the application is configured to use a more limited scope. Connected sockets (like TCP) are then bound to the VRF domain in which the connection originates.
- Connected and local routes are placed in appropriate VRF tables.
- Neighbor entries continue to be per-interface, and you can view all entries associated with the VRF device.
- A VRF does not map to its own network namespace; however, you can nest VRFs in a network namespace.
- You can use existing Linux tools to interact with it, such as `tcpdump`.

You configure VRF by associating each subset of interfaces to a VRF routing table, and configuring an instance of the routing protocol — BGP — for each routing table.



Contents

This chapter covers ...

- Configuring VRF (see page 576)
 - Specifying a Table ID (see page 576)
 - Bringing a VRF Up after Downing It with ifdown (see page 576)
 - Using the vrf Command (see page 577)
- Quagga Operation in a VRF (see page 578)
 - Example Configuration (see page 578)
- Example Commands to Show VRF Data (see page 580)
 - Showing VRF Data Using NCLU Commands (see page 580)
 - Showing VRF Data Using Quagga Commands (see page 581)
 - Showing VRF Data Using ip Commands (see page 582)
- Using BGP Unnumbered Interfaces with VRF (see page 586)
- Using DHCP with VRF (see page 588)
 - Caveats for DHCP with VRF (see page 589)
 - Example Configuration (see page 589)
- Using ping or traceroute (see page 592)
- Caveats and Errata (see page 592)



Configuring VRF

Each routing table is called a *VRF table*, and has its own table ID. You configure VRF using [NCLU \(see page 80\)](#), then place the layer 3 interface in the VRF. You can have a maximum of 64 VRFs on a switch.

When you configure a VRF, you follow a similar process to other network interfaces. Keep in mind the following for a VRF table:

- It can have an IP address, a loopback interface for the VRF.
- Associated rules are added automatically.
- You can also add a default route to avoid skipping across tables when the kernel forwards the packet.
- Names for VRF tables can be up to 15 characters. However, you **cannot** use the name *mgmt*, as this name can **only** be used for [management VRF \(see page 593\)](#).

To configure a VRF, run:

```
cumulus@switch:~$ net add vrf red vrf-table auto  
cumulus@switch:~$ net add interface swp1 vrf red
```

These commands result in the following VRF configuration in the `/etc/network/interfaces` file:

```
auto red  
iface red  
    vrf-table auto  
  
auto swp1  
iface swp1  
    vrf red
```

Specifying a Table ID

Instead of having Cumulus Linux assign a table ID for the VRF table, you can specify your own table ID in the configuration. The table ID to name mapping is saved in `/etc/iproute2/rt_tables.d/` for name-based references. So instead of using the `auto` option above, specify the table ID like this:

```
cumulus@switch:~$ net add vrf red vrf-table 1016
```



If you do specify a table ID, it **must** be in the range of 1001 to 1255 which is reserved in Cumulus Linux for VRF table IDs.

Bringing a VRF Up after Downing It with ifdown

If you take down a VRF using `ifdown`, to bring it back up you need to do one of two things:



- Use `ifup --with-dependents <vrf>`
- Use `ifreload -a`

For example:

```
cumulus@switch:~$ sudo ifdown red
cumulus@switch:~$ sudo ifup --with-dependents red
```

Using the vrf Command

The `vrf` command returns information about VRF tables that is otherwise not available in other Linux commands, such as `iproute`. You can also use it to execute non-VRF-specific commands and perform other tasks related to VRF tables.

To get a list of VRF tables, run:

```
cumulus@switch:~$ vrf list

VRF          Table
-----
red          1016
```

To return a list of processes and PIDs associated with a specific VRF table, run `vrf task list <vrf-name>`. For example:

```
cumulus@switch:~$ vrf task list red

VRF: red
-----
dhclient      2508
sshd         2659
bash          2681
su            2702
bash          2720
vrf           2829
```

To determine which VRF table is associated with a particular PID, run `vrf task identify <pid>`. For example:

```
cumulus@switch:~$ vrf task identify 2829

red
```

Running IPv4 and IPv6 Commands in a VRF Context

You can execute non-VRF-specific Linux commands and perform other tasks against a given VRF table. This typically applies to single-use commands started from a login shell, as they affect only AF_INET and AF_INET6 sockets opened by the command that gets executed; it has no impact on netlink sockets, associated with the `ip` command.

To execute such a command against a VRF table, run `vrf task exec <vrf-name> <command>`. For example, to SSH from the switch to a device accessible through VRF `red`:

```
cumulus@switch:~$ sudo vrf task exec red ssh user@host
```

You should manage long-running services with `systemd` using the `service@vrf` notation; for example, `systemctl start ntp@mgmt`. `systemd`-based services are stopped when a VRF is deleted and started when the VRF is created. For example, restarting networking or running an `ifdown/ifup` sequence.

Quagga Operation in a VRF

In Cumulus Linux 3.0 and later, BGP and static routing (IPv4 and IPv6) are supported within a VRF context. Various Quagga routing constructs, such as routing tables, nexthops, router-id, and related processing are also VRF-aware. However, OSPFv2 and OSPFv3 are not VRF-aware and can only operate in the default VRF.

[Quagga \(see page 488\)](#) learns of VRFs provisioned on the system as well as interface attachment to a VRF through notifications from the kernel.

You can assign switch ports to each VRF table with an interface-level configuration, and BGP instances can be assigned to the table with a BGP router-level command. Note that OSPFv2 and OSPFv3 are **not** VRF-aware.

Because BGP is VRF-aware, it supports per-VRF neighbors, both iBGP and eBGP as well as numbered and unnumbered interfaces. Non-interface-based VRF neighbors are bound to the VRF, which is how you can have overlapping address spaces in different VRFs. Each VRF can have its own parameters, such as address families and redistribution. Incoming connections rely on the Linux kernel for VRF-global sockets. BGP neighbors can be tracked using [BFD \(see page 549\)](#), both for single and multiple hops. You can configure multiple [BGP \(see page 516\)](#) instances, associating each with a VRF.

As mentioned above, VRFs are provisioned through `/etc/network/interfaces`. VRFs can be pre-provisioned in Quagga too, but they become active only when configured through `/etc/network/interfaces`.

- A VRF can be pre-provisioned in Quagga by running the command `vrf vrf-name`.
- A BGP instance corresponding to a VRF can be pre-provisioned by configuring `router bgp asn vrf vrf-name`. Under this context, all existing BGP parameters can be configured - neighbors, peer-groups, address-family configuration, redistribution etc.
- Static routes (IPv4 and IPv6) can be provisioned in a VRF by specifying the VRF along with the static route configuration. For example, `ip route prefix nexthop vrf vrf-name`. The VRF has to exist for this configuration to be accepted - either already defined through `/etc/network/interfaces` or pre-provisioned in Quagga.

Example Configuration

Here's an example VRF configuration in BGP:



```
cumulus@switch:~$ net add bgp vrf vrf1012 autonomous-system 64900
cumulus@switch:~$ net add bgp vrf vrf1012 router-id 6.0.2.7
cumulus@switch:~$ net add bgp vrf vrf1012 neighbor ISL peer-group
cumulus@switch:~$ net add bgp vrf vrf1012 neighbor ISLv6 peer-group
cumulus@switch:~$ net add bgp vrf vrf1012 neighbor swp1.2 interface
v6only peer-group ISLv6
cumulus@switch:~$ net add bgp vrf vrf1012 neighbor swp1.2 remote-as
external
cumulus@switch:~$ net add bgp vrf vrf1012 neighbor swp3.2 interface
v6only peer-group ISLv6
cumulus@switch:~$ net add bgp vrf vrf1012 neighbor swp3.2 remote-as
external
cumulus@switch:~$ net add bgp vrf vrf1012 neighbor 169.254.2.18
remote-as external
cumulus@switch:~$ net add bgp vrf vrf1012 neighbor 169.254.2.18 peer-
group ISL
cumulus@switch:~$ net add bgp vrf vrf1012 ipv4 unicast network
20.7.2.0/24
cumulus@switch:~$ net add bgp vrf vrf1012 ipv4 unicast neighbor ISL
activate
cumulus@switch:~$ net add bgp vrf vrf1012 neighbor ISL route-map
ALLOW_BR2 out
cumulus@switch:~$ net add bgp vrf vrf1012 ipv6 unicast network 2003:7:
2::/125
cumulus@switch:~$ net add bgp vrf vrf1012 ipv6 unicast neighbor ISLv6
activate
cumulus@switch:~$ net add bgp vrf vrf1012 neighbor ISLv6 route-map
ALLOW_BR2_v6 out
```

These commands produce the following configuration in the /etc/quagga/Quagga.conf file:

```
router bgp 64900 vrf vrf1012
  bgp router-id 6.0.2.7
  no bgp default ipv4-unicast
  neighbor ISL peer-group
  neighbor ISLv6 peer-group
  neighbor swp1.2 interface v6only peer-group ISLv6
  neighbor swp1.2 remote-as external
  neighbor swp3.2 interface v6only peer-group ISLv6
  neighbor swp3.2 remote-as external
  neighbor 169.254.2.18 remote-as external
  neighbor 169.254.2.18 peer-group ISL
!
  address-family ipv4 unicast
    network 20.7.2.0/24
    neighbor ISL activate
    neighbor ISL route-map ALLOW_BR2 out
  exit-address-family
!
```

```

address-family ipv6 unicast
  network 2003:7:2::/125
  neighbor ISLv6 activate
  neighbor ISLv6 route-map ALLOW_BR2_v6 out
exit-address-family
!

```

Example Commands to Show VRF Data

There are a number of ways to interact with VRFs, including NCLU, vtysh (the Quagga CLI) and iproute2.

Showing VRF Data Using NCLU Commands

To show the routes in the VRF:

```

cumulus@switch:~$ net show route vrf red
RIB entry for red
=====
Codes: K - kernel route, C - connected, S - static, R - RIP,
       O - OSPF, I - IS-IS, B - BGP, T - Table,
       > - selected route, * - FIB route

C>* 169.254.2.8/30 is directly connected, swp1.2
C>* 169.254.2.12/30 is directly connected, swp2.2
C>* 169.254.2.16/30 is directly connected, swp3.2

```

To show the BGP summary for the VRF:

```

cumulus@switch:~$ net show bgp vrf red summary
BGP router identifier 6.0.2.7, local AS number 64900 vrf-id 14
BGP table version 0
RIB entries 1, using 120 bytes of memory
Peers 6, using 97 KiB of memory
Peer groups 2, using 112 bytes of memory

Neighbor          V     AS MsgRcvd MsgSent      TblVer  InQ OutQ Up/Down
State/PfxRcd
s3(169.254.2.18)
                  4 65000   102039   102040          0     0     0
3d13h03m          0
s1(169.254.2.10)
                  4 65000   102039   102040          0     0     0
3d13h03m          0
s2(169.254.2.14)
                  4 65000   102039   102040          0     0     0
3d13h03m          0

Total number of neighbors 3

```

To show BGP (IPv4) routes in the VRF:

```
cumulus@switch:~$ net show bgp vrf vrf1012
BGP table version is 0, local router ID is 6.0.2.7
Status codes: s suppressed, d damped, h history, * valid, > best, =
multipath,
                i internal, r RIB-failure, S Stale, R Removed
Origin codes: i - IGP, e - EGP, ? - incomplete

Network          Next Hop          Metric LocPrf Weight Path
20.7.2.0/24      0.0.0.0          0        32768 i

Total number of prefixes 1
```

However, to show BGP IPv6 routes in the VRF, you need to use `vtysh`, the Quagga CLI:

```
cumulus@switch:~$ sudo vtysh
switch# show bgp vrf vrf1012
BGP table version is 0, local router ID is 6.0.2.7
Status codes: s suppressed, d damped, h history, * valid, > best, =
multipath,
                i internal, r RIB-failure, S Stale, R Removed
Origin codes: i - IGP, e - EGP, ? - incomplete

Network          Next Hop          Metric LocPrf Weight Path
2003:7:2::/125   ::              0        32768 i

Total number of prefixes 1
switch# exit
cumulus@switch:~$
```

Showing VRF Data Using Quagga Commands

Show all VRFs learned by Quagga from the kernel. The table ID shows the corresponding routing table in the kernel either automatically assigned or manually defined:

```
cumulus@switch:~$ sudo vtysh
switch# show vrf
vrf vrf1012 id 14 table 1012
vrf vrf1013 id 21 table 1013
vrf vrf1014 id 28 table 1014
switch# exit
cumulus@switch:~$
```

Show VRFs configured in BGP, including the default. A non-zero ID is a VRF that has also been actually provisioned — that is, defined in `/etc/network/interfaces`:

```
cumulus@switch:~$ sudo vtysh
switch# show bgp vrfs
Type   Id      RouterId          #PeersCfg  #PeersEstb  Name
DFLT   0       6.0.0.7           0          0  Default
VRF    14      6.0.2.7           6          6  vrf1012
VRF    21      6.0.3.7           6          6  vrf1013
VRF    28      6.0.4.7           6          6  vrf1014

Total number of VRFs (including default): 4
switch# exit
cumulus@switch:~$
```

Display interfaces known to Quagga and attached to this VRF:

```
cumulus@switch:~$ sudo vtysh
switch# show interface vrf vrf1012
Interface br2 is up, line protocol is down
  PTM status: disabled
  vrf: vrf1012
  index 13 metric 0 mtu 1500
  flags: <UP,BROADCAST,MULTICAST>
  inet 20.7.2.1/24

  inet6 fe80::202:ff:fe00:a/64
  ND advertised reachable time is 0 milliseconds
  ND advertised retransmit interval is 0 milliseconds
  ND router advertisements are sent every 600 seconds
  ND router advertisements lifetime tracks ra-interval
  ND router advertisement default router preference is medium
  Hosts use stateless autoconfig for addresses.
switch# exit
cumulus@switch:~$
```

Showing VRF Data Using ip Commands

To list all VRFs provisioned, showing the VRF ID (vrf1012, vrf1013 and vrf1014 below) as well as the table ID:

```
cumulus@switch:~$ ip -d link show type vrf
14: vrf1012: <NOARP,MASTER,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast
state UNKNOWN mode DEFAULT group default qlen 1000
  link/ether 46:96:c7:64:4d:fa brd ff:ff:ff:ff:ff:ff promiscuity 0
  vrf table 1012 addrgenmode eui64
21: vrf1013: <NOARP,MASTER,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast
state UNKNOWN mode DEFAULT group default qlen 1000
  link/ether 7a:8a:29:0f:5e:52 brd ff:ff:ff:ff:ff:ff promiscuity 0
  vrf table 1013 addrgenmode eui64
```



```
28: vrf1014: <NOARP,MASTER,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast
    state UNKNOWN mode DEFAULT group default qlen 1000
        link/ether e6:8c:4d:fc:eb:b1 brd ff:ff:ff:ff:ff:ff promiscuity 0
        vrf table 1014 addrgenmode eui64
```

To list the interfaces attached to a specific VRF:

```
cumulus@switch:~$ ip -d link show vrf vrf1012
8: swp1.2@swp1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue
    master vrf1012 state UP mode DEFAULT group default
        link/ether 00:02:00:00:00:07 brd ff:ff:ff:ff:ff:ff promiscuity 0
        vlan protocol 802.1Q id 2 <REORDER_HDR>
        vrf_slave addrgenmode eui64
9: swp2.2@swp2: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue
    master vrf1012 state UP mode DEFAULT group default
        link/ether 00:02:00:00:00:08 brd ff:ff:ff:ff:ff:ff promiscuity 0
        vlan protocol 802.1Q id 2 <REORDER_HDR>
        vrf_slave addrgenmode eui64
10: swp3.2@swp3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue
    master vrf1012 state UP mode DEFAULT group default
        link/ether 00:02:00:00:00:09 brd ff:ff:ff:ff:ff:ff promiscuity 0
        vlan protocol 802.1Q id 2 <REORDER_HDR>
        vrf_slave addrgenmode eui64
11: swp4.2@swp4: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue
    master vrf1012 state UP mode DEFAULT group default
        link/ether 00:02:00:00:00:0a brd ff:ff:ff:ff:ff:ff promiscuity 0
        vlan protocol 802.1Q id 2 <REORDER_HDR>
        vrf_slave addrgenmode eui64
12: swp5.2@swp5: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue
    master vrf1012 state UP mode DEFAULT group default
        link/ether 00:02:00:00:00:0b brd ff:ff:ff:ff:ff:ff promiscuity 0
        vlan protocol 802.1Q id 2 <REORDER_HDR>
        vrf_slave addrgenmode eui64
13: br2: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc noqueue
    master vrf1012 state DOWN mode DEFAULT group default
        link/ether 00:00:00:00:00:00 brd ff:ff:ff:ff:ff:ff promiscuity 0
        bridge forward_delay 100 hello_time 200 max_age 2000 ageing_time
        30000 stp_state 0 priority 32768
        vlan_filtering 0 vlan_protocol 802.1Q bridge_id 8000.0:0:0:0:0:0
        designated_root 8000.0:0:0:0:0:0
        root_port 0 root_path_cost 0 topology_change 0
        topology_change_detected 0 hello_timer 0.00
        tcn_timer 0.00 topology_change_timer 0.00 gc_timer 202.23
        vlan_default_pvid 1 group_fwd_mask 0
        group_address 01:80:c2:00:00:00 mcast_snooping 1 mcast_router 1
        mcast_query_use_ifaddr 0 mcast_querier 0
        mcast_hash_elasticity 4096 mcast_hash_max 4096
        mcast_last_member_count 2 mcast_startup_query_count 2
        mcast_last_member_interval 100 mcast_membership_interval 26000
        mcast_querier_interval 25500
```

```
mcast_query_interval 12500 mcast_query_response_interval 1000
mcast_startup_query_interval 3125
    nf_call_iptables 0 nf_call_ip6tables 0 nf_call_arptables 0
    vrf_slave addrgenmode eui64
```

To show IPv4 routes in a VRF:

```
cumulus@switch:~$ ip route show table vrf1012
unreachable default metric 240
broadcast 20.7.2.0 dev br2 proto kernel scope link src 20.7.2.1
dead linkdown
20.7.2.0/24 dev br2 proto kernel scope link src 20.7.2.1 dead
linkdown
local 20.7.2.1 dev br2 proto kernel scope host src 20.7.2.1
broadcast 20.7.2.255 dev br2 proto kernel scope link src 20.7.2.1
dead linkdown
broadcast 169.254.2.8 dev swp1.2 proto kernel scope link src
169.254.2.9
169.254.2.8/30 dev swp1.2 proto kernel scope link src 169.254.2.9
local 169.254.2.9 dev swp1.2 proto kernel scope host src
169.254.2.9
broadcast 169.254.2.11 dev swp1.2 proto kernel scope link src
169.254.2.9
broadcast 169.254.2.12 dev swp2.2 proto kernel scope link src
169.254.2.13
169.254.2.12/30 dev swp2.2 proto kernel scope link src
169.254.2.13
local 169.254.2.13 dev swp2.2 proto kernel scope host src
169.254.2.13
broadcast 169.254.2.15 dev swp2.2 proto kernel scope link src
169.254.2.13
broadcast 169.254.2.16 dev swp3.2 proto kernel scope link src
169.254.2.17
169.254.2.16/30 dev swp3.2 proto kernel scope link src
169.254.2.17
local 169.254.2.17 dev swp3.2 proto kernel scope host src
169.254.2.17
broadcast 169.254.2.19 dev swp3.2 proto kernel scope link src
169.254.2.17
```

To show IPv6 routes in a VRF:

```
cumulus@switch:~$ ip -6 route show table vrf1012
local fe80:: dev lo proto none metric 0 pref medium
local fe80:: dev lo proto none metric 0 pref medium
local fe80:: dev lo proto none metric 0 pref medium
local fe80:: dev lo proto none metric 0 pref medium
local fe80::202:ff:fe00:7 dev lo proto none metric 0 pref medium
local fe80::202:ff:fe00:8 dev lo proto none metric 0 pref medium
```



```
local fe80::202:ff:fe00:9 dev lo proto none metric 0 pref medium
local fe80::202:ff:fe00:a dev lo proto none metric 0 pref medium
fe80::/64 dev br2 proto kernel metric 256 dead linkdown pref medium
fe80::/64 dev swp1.2 proto kernel metric 256 pref medium
fe80::/64 dev swp2.2 proto kernel metric 256 pref medium
fe80::/64 dev swp3.2 proto kernel metric 256 pref medium
ff00::/8 dev br2 metric 256 dead linkdown pref medium
ff00::/8 dev swp1.2 metric 256 pref medium
ff00::/8 dev swp2.2 metric 256 pref medium
ff00::/8 dev swp3.2 metric 256 pref medium
unreachable default dev lo metric 240 error -101 pref medium
```

To see a list of links associated with a particular VRF table, run `ip link list <vrf-name>`. For example:

```
cumulus@switch:~$ ip link list red

VRF: red
-----
swp1.10@swp1      UP          6c:64:1a:00:5a:0c <BROADCAST,
MULTICAST,UP,LOWER_UP>
swp2.10@swp2      UP          6c:64:1a:00:5a:0d <BROADCAST,
MULTICAST,UP,LOWER_UP>
```

To see a list of routes associated with a particular VRF table, run `ip route list <vrf-name>`. For example:

```
cumulus@switch:~$ ip route list red

VRF: red
-----
unreachable default metric 8192
10.1.1.0/24 via 10.10.1.2 dev swp2.10
10.1.2.0/24 via 10.99.1.2 dev swp1.10
broadcast 10.10.1.0 dev swp2.10 proto kernel scope link src
10.10.1.1
10.10.1.0/28 dev swp2.10 proto kernel scope link src 10.10.1.1
local 10.10.1.1 dev swp2.10 proto kernel scope host src 10.10.1.1
broadcast 10.10.1.15 dev swp2.10 proto kernel scope link src
10.10.1.1
broadcast 10.99.1.0 dev swp1.10 proto kernel scope link src
10.99.1.1
10.99.1.0/30 dev swp1.10 proto kernel scope link src 10.99.1.1
local 10.99.1.1 dev swp1.10 proto kernel scope host src 10.99.1.1
broadcast 10.99.1.3 dev swp1.10 proto kernel scope link src
10.99.1.1

local fe80:: dev lo proto none metric 0 pref medium
local fe80:: dev lo proto none metric 0 pref medium
```

```

local fe80::6e64:1aff:fe00:5a0c dev lo proto none metric 0 pref
medium
local fe80::6e64:1aff:fe00:5a0d dev lo proto none metric 0 pref
medium
fe80::/64 dev swp1.10 proto kernel metric 256 pref medium
fe80::/64 dev swp2.10 proto kernel metric 256 pref medium
ff00::/8 dev swp1.10 metric 256 pref medium
ff00::/8 dev swp2.10 metric 256 pref medium
unreachable default dev lo metric 8192 error -101 pref medium

```



You can also show routes in a VRF using `ip [-6] route show vrf <name>`. This command omits local and broadcast routes, which can clutter the output.

Using BGP Unnumbered Interfaces with VRF

BGP unnumbered interface configurations (see page 516) are supported with VRF. In BGP unnumbered, there are no addresses on any interface. However, debugging tools like traceroute need at least a single IP address per node as the node's source IP address. Typically, this address was assigned to the loopback device. With VRF, you need a loopback device for each VRF table since VRF is based on interfaces, not IP addresses. While Linux does not support multiple loopback devices, it does support the concept of a dummy interface, which is used to achieve the same goal.

An IP address can be associated with the VRF device, which will then act as the dummy (loopback-like) interface for that VRF.

1. Configure the BGP unnumbered configuration. The BGP unnumbered configuration is the same for a non-VRF, applied under the VRF context (`router bgp asn vrf <vrf-name>`).

```

cumulus@switch:~$ net add vrf vrf1 vrf-table auto
cumulus@switch:~$ net add vrf vrf1 ip address 6.1.0.6/32
cumulus@switch:~$ net add vrf vrf1 ipv6 address 2001:6:1::6/128
cumulus@switch:~$ net add interface swp1 link speed 10000
cumulus@switch:~$ net add interface swp1 link autoneg off
cumulus@switch:~$ net add interface swp1 vrf vrf1
cumulus@switch:~$ net add vlan 101 ip address 20.1.6.1/24
cumulus@switch:~$ net add vlan 101 ipv6 address 2001:20:1:6::1/80
cumulus@switch:~$ net add bridge bridge ports vlan101

```

These commands create the following configuration in the `/etc/network/interfaces` file:

```

auto swp1
iface swp1
    link-autoneg on
    link-speed 10000
    vrf vrf1

auto bridge

```



```
iface bridge
    bridge-ports vlan101
    bridge-vids 101
    bridge-vlan-aware yes

auto vlan101
iface vlan101
    address 20.1.6.1/24
    address 2001:20:1:6::1/80
    vlan-id 101
    vlan-raw-device bridge

auto vrf1
iface vrf1
    address 6.1.0.6/32
    address 2001:6:1::6/128
    vrf-table auto
```

Here is the Quagga BGP configuration:

```
cumulus@switch:~$ net add bgp vrf vrf1 autonomous-system 65001
cumulus@switch:~$ net add bgp vrf vrf1 bestpath as-path multipath-relax
cumulus@switch:~$ net add bgp vrf vrf1 bestpath compare-routerid
cumulus@switch:~$ net add bgp vrf vrf1 neighbor LEAF peer-group
cumulus@switch:~$ net add bgp vrf vrf1 neighbor LEAF remote-as external
cumulus@switch:~$ net add bgp vrf vrf1 neighbor LEAF capability extended-nexthop
cumulus@switch:~$ net add bgp vrf vrf1 neighbor swp1.101 interface
peer-group LEAF
cumulus@switch:~$ net add bgp vrf vrf1 neighbor swp2.101 interface
peer-group LEAF
cumulus@switch:~$ net add bgp vrf vrf1 ipv4 unicast redistribute
connected
cumulus@switch:~$ net add bgp vrf vrf1 ipv4 unicast neighbor LEAF
activate
cumulus@switch:~$ net add bgp vrf vrf1 ipv6 unicast redistribute
connected
cumulus@switch:~$ net add bgp vrf vrf1 ipv6 unicast neighbor LEAF
activate
```

These commands create the following configuration in the /etc/quagga/Quagga.conf file:

```
!
router bgp 65001 vrf vrf1
no bgp default ipv4-unicast
bgp bestpath as-path multipath-relax
```



```
bgp bestpath compare-routerid
neighbor LEAF peer-group
neighbor LEAF remote-as external
neighbor LEAF capability extended-nexthop
neighbor swp1.101 interface peer-group LEAF
neighbor swp2.101 interface peer-group LEAF
!
address-family ipv4 unicast
  redistribute connected
  neighbor LEAF activate
exit-address-family
!
address-family ipv6 unicast
  redistribute connected
  neighbor LEAF activate
exit-address-family
!
```

Using DHCP with VRF

Since you can use VRF to bind IPv4 and IPv6 sockets to non-default VRF tables, you have the ability to start DHCP servers and relays in any non-default VRF table using the `dhcpd` and `dhcrelay` services, respectively. These services must be managed by `systemd` in order to run in a VRF context; in addition, the services must be listed in `/etc/vrf/systemd.conf`. By default, this file already lists these two services, as well as others like `ntp` and `snmpd`. You can add more services as needed, such as `dhcpd6` and `dhcrelay6` for IPv6.

If you edit `/etc/vrf/systemd.conf`, run `sudo systemctl daemon-reload` to generate the `systemd` instance files for the newly added service(s). Then you can start the service in the VRF using `systemctl start <service>@<vrf-name>.service`, where `<service>` is the name of the service — such as `dhcpd` or `dhcrelay` — and `<vrf-name>` is the name of the VRF.

For example, to start the `dhcrelay` service after you configured a VRF named `blue`, run:

```
cumulus@switch:~$ sudo systemctl start dhcrelay@blue.service
```

To enable the service at boot time you should also run `systemctl enable <service>@<vrf-name>`. To continue with the previous example:

```
cumulus@switch:~$ sudo systemctl enable dhcrelay@blue.service
```

In addition, you need to create a separate default file in `/etc/default` for every instance of a DHCP server and/or relay in a non-default VRF; this is where you set the server and relay options. To run multiple instances of any of these services, you need a separate file for each instance. The files must be named as follows:

- `isc-dhcp-server-<vrf-name>`
- `isc-dhcp-server6-<vrf-name>`
- `isc-dhcp-relay-<vrf-name>`

- isc-dhcp-relay6-<vrf-name>

See the example configuration below for more details.

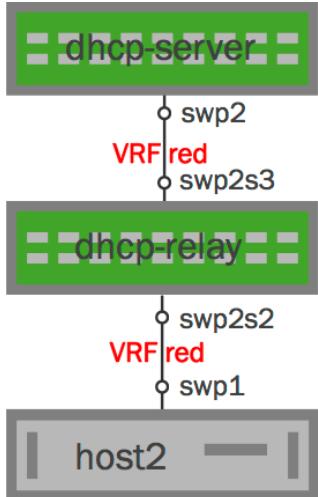
Caveats for DHCP with VRF

- Cumulus Linux does **not** support DHCP server and relay across VRFs, so the server and host cannot be in different VRF tables. In addition, the server and relay cannot be in different VRF tables.
- Typically a service running in the default VRF owns a port across all VRFs. If the VRF local instance is preferred, the global one may need to be disabled and stopped first.
- VRF is a layer 3 routing feature. It only makes sense to run programs that use AF_INET and AF_INET6 sockets in a VRF. VRF context does not affect any other aspects of the operation of a program.
- This method only works with `systemd`-based services.

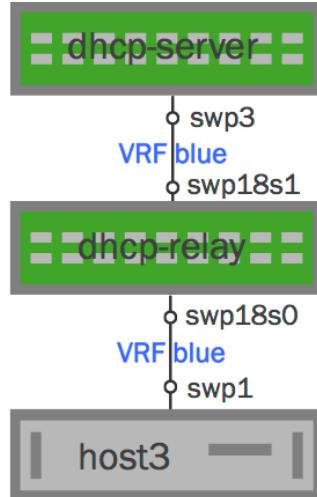
Example Configuration

In the following example, there is one IPv4 network with a VRF named *red* and one IPv6 network with a VRF named *blue*.

The IPv4 DHCP server/relay network looks like this:



The IPv6 DHCP server/relay network looks like this:



Configure each DHCP server and relay as follows:

Sample DHCP Server Configuration

1. Create the file `isc-dhcp-server-red` in `/etc/default/`. Here is sample content:

```

# Defaults for isc-dhcp-
server initscript
# sourced by /etc/init.d
/isc-dhcp-server
  
```

Sample DHCP6 Server Configuration

1. Create the file `isc-dhcp-server6-blue` in `/etc/default/`. Here is sample content:

```

# Defaults for isc-dhcp-
server initscript
# sourced by /etc/init.d
/isc-dhcp-server
  
```

```

# installed at /etc/default
/etc/dhcp-server by the
maintainer scripts
#
# This is a POSIX shell
fragment
#
# Path to dhcpcd's config
file (default: /etc/dhcp
/dhcpcd.conf).
DHCPD_CONF="-cf /etc/dhcp
/dhcpcd-red.conf"
# Path to dhcpcd's PID file
(default: /var/run/dhcpcd.
pid).
DHCPD_PID="-pf /var/run
/dhcpcd-red.pid"
# Additional options to
start dhcpcd with.
# Don't use options -cf or
-pf here; use DHCPD_CONF/
DHCPD_PID instead
#OPTIONS=""
# On what interfaces
should the DHCP server
(dhcpcd) serve DHCP
requests?
# Separate multiple
interfaces with spaces, e.
g. "eth0 eth1".
INTERFACES="swp2"

```

2. Enable the DHCP server:
cumulus@switch:~\$ sudo systemctl enable dhcpcd@red.service
3. Start the DHCP server:
cumulus@switch:~\$ sudo systemctl start dhcpcd@red.service
or
cumulus@switch:~\$ sudo systemctl restart dhcpcd@red.service
4. Check status:
cumulus@switch:~\$ sudo systemctl status dhcpcd@red.service



You can create this configuration using the `vrf` command ([see above \(see page 578\)](#) for more details):

```

# installed at /etc/default
/etc/dhcp-server by the
maintainer scripts
#
# This is a POSIX shell
fragment
#
# Path to dhcpcd's config
file (default: /etc/dhcp
/dhcpcd.conf).
DHCPD_CONF="-cf /etc/dhcp
/dhcpcd6-blue.conf"
# Path to dhcpcd's PID file
(default: /var/run/dhcpcd.
pid).
DHCPD_PID="-pf /var/run
/dhcpcd6-blue.pid"
# Additional options to
start dhcpcd with.
# Don't use options -cf or
-pf here; use DHCPD_CONF/
DHCPD_PID instead
#OPTIONS=""
# On what interfaces
should the DHCP server
(dhcpcd) serve DHCP
requests?
# Separate multiple
interfaces with spaces, e.
g. "eth0 eth1".
INTERFACES="swp3"

```

2. Enable the DHCP server:
cumulus@switch:~\$ sudo systemctl enable dhcpcd6@blue.service
3. Start the DHCP server:
cumulus@switch:~\$ sudo systemctl start dhcpcd6@blue.service
or
cumulus@switch:~\$ sudo systemctl restart dhcpcd6@blue.service
4. Check status:
cumulus@switch:~\$ sudo systemctl status dhcpcd6@blue.service



You can create this configuration using the `vrf` command ([see above \(see page 578\)](#) for more details):



```
cumulus@switch:~$ sudo  
vrf task exec red /usr  
/sbin/dhcpcd -f -q -cf /  
/etc/dhcp/dhcpcd-red.  
conf -pf /var/run/dhcpcd-  
red.pid swp2
```

```
cumulus@switch:~$ sudo  
vrf task exec blue dhcpcd  
-6 -q -cf /  
/etc/dhcp/dhcpcd6-  
blue.conf -pf /var/run  
/dhcpcd6-blue.pid swp3
```

Sample DHCP Relay Configuration

1. Create the file `isc-dhcp-relay-red` in `/etc/default/`. Here is sample content:

```
# Defaults for isc-dhcp-  
relay initscript  
# sourced by /etc/init.d  
/isc-dhcp-relay  
# installed at /etc/default  
/isc-dhcp-relay by the  
maintainer scripts  
#  
# This is a POSIX shell  
fragment  
#  
# What servers should the  
DHCP relay forward  
requests to?  
SERVERS="102.0.0.2"  
# On what interfaces  
should the DHCP relay  
(dhrelay) serve DHCP  
requests?  
# Always include the  
interface towards the DHCP  
server.  
# This variable requires a  
-i for each interface  
configured above.  
# This will be used in the  
actual dhcrelay command  
# For example, "-i eth0 -i  
eth1"  
INTF_CMD="-i swp2s2 -i  
swp2s3"  
# Additional options that  
are passed to the DHCP  
relay daemon?  
OPTIONS=""
```

Sample DHCP6 Relay Configuration

1. Create the file `isc-dhcp-relay6-blue` in `/etc/default/`. Here is sample content:

```
# Defaults for isc-dhcp-  
relay initscript  
# sourced by /etc/init.d  
/isc-dhcp-relay  
# installed at /etc/default  
/isc-dhcp-relay by the  
maintainer scripts  
#  
# This is a POSIX shell  
fragment  
#  
# What servers should the  
DHCP relay forward  
requests to?  
#SERVERS="103.0.0.2"  
# On what interfaces  
should the DHCP relay  
(dhrelay) serve DHCP  
requests?  
# Always include the  
interface towards the DHCP  
server.  
# This variable requires a  
-i for each interface  
configured above.  
# This will be used in the  
actual dhcrelay command  
# For example, "-i eth0 -i  
eth1"  
INTF_CMD="-l swp18s0 -u  
swp18s1"  
# Additional options that  
are passed to the DHCP  
relay daemon?
```

2. Enable the DHCP relay:

```
cumulus@switch:~$ sudo systemctl
enable dhcrelay@red.service
```

3. Start the DHCP relay:

```
cumulus@switch:~$ sudo systemctl
start dhcrelay@red.service
or
cumulus@switch:~$ sudo systemctl
restart dhcrelay@red.service
```

4. Check status:

```
cumulus@switch:~$ sudo systemctl
status dhcrelay@red.service
```



You can create this configuration using the `vrf` command ([see above \(see page 578\)](#) for more details):

```
cumulus@switch:~$ sudo
vrf task exec red /usr
/sbin/dhcrelay -d -q -i /
swp2s2 -i swp2s3
102.0.0.2
```

```
OPTIONS="-pf /var/run
/dhcrelay6@blue.pid"
```

2. Enable the DHCP relay:

```
cumulus@switch:~$ sudo systemctl
enable dhcrelay6@blue.service
```

3. Start the DHCP relay:

```
cumulus@switch:~$ sudo systemctl
start dhcrelay6@blue.service
or
cumulus@switch:~$ sudo systemctl
restart dhcrelay6@blue.service
```

4. Check status:

```
cumulus@switch:~$ sudo systemctl
status dhcrelay6@blue.service
```



You can create this configuration using the `vrf` command ([see above \(see page 578\)](#) for more details):

```
cumulus@switch:~$ sudo
vrf task exec blue /usr
/sbin/dhcrelay -d -q -6 -
l /
swp18s0 -u swp18s1 -
pf /var/run
/dhcrelay6@blue.pid
```

Using ping or traceroute

If you wish to use `ping` or `traceroute` on a VRF, use the `-I <vrf>` flag for `ping` and `-i <vrf>` for `traceroute`.

```
cumulus@switch:~$ ping -I blue
```

Or:

```
cumulus@switch:~$ sudo traceroute -i blue
```

Caveats and Errata

- The Penguin Computing Arctica 4804IP switch does not support VRFs.



- While there is a fixed limit of 64 VRF devices, Cumulus Networks has validated up to 20 VRF devices on a switch at one time.
- Table selection based on the incoming interface only; currently, packet attributes or output-interface-based selection are not available.
- BGP (IPv4/IPv6) is the only routing protocol supported currently. There is no support for OSPF (IPv4 /IPv6) at this time.
- Setting the router ID outside of BGP via the `router-id` option causes all BGP instances to get the same router ID. If you want each BGP instance to have its own router ID, specify the `router-id` under the BGP instance using `bgp router-id`. If both are specified, the one under the BGP instance overrides the one provided outside BGP.
- It is not possible to leak routes across VRFs within Quagga.
- You cannot configure [EVPN address families](#) (see page 438) within a VRF.

Management VRF

Management VRF — a subset of [VRF \(see page 574\)](#) (virtual routing tables and forwarding) — provides a separation between the out-of-band management network and the in-band data plane network. For all VRFs, the *main* routing table is the default table for all of the data plane switch ports. With management VRF, a second table, *mgmt*, is used for routing through the switch's Ethernet ports. The *mgmt* name is special cased to identify the management VRF from a data plane VRF. FIB rules are installed for DNS servers since this is the typical deployment case.

Cumulus Linux only supports eth0 as the management interface, or eth1, depending on the switch platform. The Ethernet ports are software-only parts that are not hardware accelerated by `switchd`. VLAN subinterfaces, bonds, bridges and the front panel switch ports are not supported as management interfaces.

When management VRF is enabled, logins to the switch are set into the management VRF context. IPv4 and IPv6 networking applications (for example, Ansible, Chef and `apt-get`) run by an administrator communicate out the management network by default. This default context does not impact services run through `systemd` and the `systemctl` command, and does not impact commands examining the state of the switch; for example, using the `ip` command to list links, neighbors or routes.



The management VRF configurations in this chapter contain a localhost loopback IP address (127.0.0.1/8). Putting the loopback address in the management VRF's L3 domain prevents issues with applications that expect the loopback IP address to exist in the VRF, such as NTP.

Contents

This chapter covers ...

- [Enabling Management VRF \(see page 594\)](#)
 - [Bringing the Management VRF Up after Downing It with ifdown \(see page 595\)](#)
 - [Enabling NTP \(see page 595\)](#)
 - [Enabling snmpd \(see page 596\)](#)
 - [Enabling hsflowd \(see page 597\)](#)
 - [Using ping or traceroute \(see page 598\)](#)



- OSPF and BGP (see page 598)
 - Redistributing Routes in Management VRF (see page 598)
- SNMP Traps Use eth0 Only (see page 599)
- Using SSH within a Management VRF Context (see page 599)
- Viewing the Routing Tables (see page 599)
 - Viewing a Single Route (see page 599)
- Using the mgmt Interface Class (see page 600)
- Management VRF and DNS (see page 600)
- Incompatibility with cl-ns-mgmt (see page 601)

Enabling Management VRF

To enable management VRF on eth0, complete the following steps:

Example Management VRF Configuration

The example NCLU commands below create a VRF called *mgmt*:



The management VRF must be named `mgmt` to differentiate from a data plane VRF.

```
cumulus@switch:~$ net add vrf mgmt
cumulus@switch:~$ net pending
cumulus@switch:~$ net commit
```

The NCLU commands above create the following snippets in `/etc/network/interfaces`:

```
...
auto eth0
iface eth0 inet dhcp
    vrf mgmt

...
auto mgmt
iface mgmt
    address 127.0.0.1/8
    vrf-table auto

...
```



When you commit the change for adding the management VRF, it drops all connections over eth0. This can impact any automation that may be running, such as Ansible or Puppet scripts.

Bringing the Management VRF Up after Downing It with ifdown

If you take down the management VRF using `ifdown`, to bring it back up you need to do one of two things:

- Use `ifup --with-dependents <vrft>`
- Use `ifreload -a`

For example:

```
cumulus@switch:~$ sudo ifdown mgmt
cumulus@switch:~$ sudo ifup --with-dependents mgmt
```



Running `ifreload -a` disconnects the session for any interface configured as `auto`.

Enabling NTP

To enable NTP to run in the mgmt VRF:

1. Configure the `mgmt` VRF as described in [the Enabling Management VRF section above](#) (see page 593)
2. If NTP is running, stop the service:

```
cumulus@switch:~$ sudo systemctl stop ntp.service
```



By default, NTP is running in the default VRF, and to automatically start when the system boots; the NTP service needs to be stopped, disabled, and then restarted once the mgmt VRF is configured.

3. Disable NTP from automatically starting in the default VRF:

```
cumulus@switch:~$ sudo systemctl disable ntp.service
```

4. Start NTP in the mgmt VRF.

```
cumulus@switch:~$ sudo systemctl start ntp@mgmt
```



5. Verify that NTP peers are active.

```
cumulus@switch:~$ ntpq -pn
      remote                  refid            st t when poll reach   delay
  offset  jitter
=====
=====
*38.229.71.1    204.9.54.119      2 u    42   64  377   31.275   -
  0.625   3.105
-104.131.53.252 209.51.161.238      2 u    47   64  377   16.381   -
  5.251   0.681
+45.79.10.228   200.98.196.212      2 u    44   64  377   42.998
  0.115   0.585
+74.207.240.206 127.67.113.92      2 u    43   64  377   73.240   -
  1.623   0.320
```

6. Enable ntp@mgmt so that it starts when the switch boots:

```
cumulus@switch:~$ sudo systemctl enable ntp@mgmt
```

Enabling snmpd

To enable `snmpd` to run in the mgmt VRF:

1. Configure the `mgmt` VRF as described in [the Enabling Management VRF section above](#) (see page 593)
2. Stop the `snmpd` daemon if it is running:

```
cumulus@switch:~$ sudo systemctl stop snmpd.service
```

3. Disable `snmpd` to ensure it does not start in the default VRF if the system is rebooted:

```
cumulus@switch:~$ sudo systemctl disable snmpd.service
```

4. Start `snmpd` in the the mgmt VRF:

```
cumulus@switch:~$ sudo systemctl start snmpd@mgmt
```

5. Enable `snmpd@mgmt` so it starts when the switch boots:

```
cumulus@switch:~$ sudo systemctl enable snmpd@mgmt
```



Enabling hsflowd

If you're using sFlow to monitor traffic in the mgmt VRF, you need to complete the following steps to enable it.

1. Add the `hsflowd` process to the `systemd` configuration file in `/etc/vrf`. Edit `/etc/vrf/systemd.conf` in a text editor.

```
cumulus@switch:~$ sudo nano /etc/vrf/systemd.conf
# Systemd-based services that are expected to be run in a VRF
context.
#
# If changes are made to this file run systemctl daemon-reload
# to re-generate systemd files.
chef-client
collectd
dhcpcd
dhcrelay
hsflowd  <<< Add this line
ntp
puppet
snmpd
snmptrapd
ssh
zabbix-agent
```

2. Stop, disable, reload, reenable and restart the `hsflowd` service.

```
cumulus@switch:~$ sudo systemctl stop hsflowd.service
cumulus@switch:~$ sudo systemctl disable hsflowd.service
cumulus@switch:~$ sudo systemctl daemon-reload
cumulus@switch:~$ sudo systemctl enable hsflowd@mgmt.service
cumulus@switch:~$ sudo systemctl start hsflowd@mgmt.service
```

3. Verify that the `hsflowd` service is running in the mgmt VRF.

```
cumulus@switch:~$ ps aux | grep flow
root      7294  0.0  0.4  81320  2108 ?          Ssl   22:22   0:00
/usr/sbin/hsflowd
cumulus    7906  0.0  0.4  12728   2056 pts/0      S+   22:34   0:00
grep flow
cumulus@switch:~$ vrf task identify 7294
mgmt
```



Using ping or traceroute

By default, issuing a `ping` or `traceroute` assumes the packet should be sent to the dataplane network (the main routing table). If you wish to use `ping` or `traceroute` on the management network, use the `-I` flag for `ping` and `-i` for `traceroute`.

```
cumulus@switch:~$ ping -I mgmt
```

Or:

```
cumulus@switch:~$ sudo traceroute -i mgmt
```

OSPF and BGP

In general, no changes are required for either BGP or OSPF. Quagga was updated in Cumulus Linux 3.0 to be VRF-aware and automatically sends packets based on the switch port routing table. This includes BGP peering via loopback interfaces. BGP does routing lookups in the default table. However, one modification you may consider has to do with how your routes get redistributed.

Redistributing Routes in Management VRF

Management VRF uses the `mgmt` table, including local routes. It does not affect how the routes are redistributed when using routing protocols such as OSPF and BGP.

To redistribute the routes in your network, use the `redistribute connected` command under BGP or OSPF. This enables the directly connected network out of `eth0` to be advertised to its neighbor.



This also creates a route on the neighbor device to the management network through the data plane, which may not be desired.

Cumulus Networks recommends you always use route maps to control the advertised networks redistributed by the `redistribute connected` command. For example, you can specify a route map to redistribute routes in this way (for both BGP and OSPF):

```
cumulus@leaf01:~$ net add routing route-map REDISTRIBUTE-CONNECTED  
deny 100 match interface eth0  
cumulus@leaf01:~$ net add routing route-map REDISTRIBUTE-CONNECTED  
permit 1000
```

These commands produce the following configuration snippet in the `/etc/quagga/Quagga.conf` file:

```
<routing protocol>  
redistribute connected route-map REDISTRIBUTE-CONNECTED
```

```
route-map REDISTRIBUTE-CONNECTED deny 100
  match interface eth0
!
route-map REDISTRIBUTE-CONNECTED permit 1000
```

SNMP Traps Use eth0 Only

SNMP cannot use a switch port to send data. For any SNMP traps, this traffic gets sent out to eth0. Cumulus Networks plans to support switch ports in the future.



For SNMP, this restriction only applies to traps. SNMP polling is not affected.

Using SSH within a Management VRF Context

If you SSH to the switch through a switch port, it works as expected. If you need to SSH from the device out of a switch port, use `vrf exec default ssh <ip_address_of_swp_port>`. For example:

```
cumulus@switch:~$ sudo vrf exec default ssh 10.23.23.2 10.3.3.3
```

Viewing the Routing Tables

When you look at the routing table with `ip route show`, you are looking at the switch port (*main*) table. You can also see the dataplane routing table with `net show route vrf main`.

To look at information about eth0 (the management routing table), use `net show route vrf mgmt`.

```
cumulus@switch:~$ net show route vrf mgmt
default via 192.168.0.1 dev eth0

cumulus@switch:~$ net show route
default via 10.23.23.3 dev swp17 proto zebra metric 20
10.3.3.3 via 10.23.23.3 dev swp17
10.23.23.0/24 dev swp17 proto kernel scope link src 10.23.23.2
192.168.0.0/24 dev eth0 proto kernel scope link src 192.168.0.11
```

Viewing a Single Route

Note that if you use `ip route get` to return information about a single route, the command resolves over the *mgmt* table by default. To get information about the route in the switching silicon, use:

```
cumulus@switch:~$ net show route <addr>
```

To get the route for any VRF, run:



```
cumulus@switch:~$ net show route vrf mgmt <addr>
```

Using the mgmt Interface Class

In `ifupdown2` interface classes (see page 186) are used to create a user-defined grouping for interfaces. The special class `mgmt` is available to separate the switch's management interfaces from the data interfaces. This allows you to manage the data interfaces by default using `ifupdown2` commands. Performing operations on the `mgmt` interfaces requires specifying the `--allow=mgmt` option, which prevents inadvertent outages on the management interfaces. Cumulus Linux by default brings up all interfaces in both the `auto` (default) class and the `mgmt` interface class when the switch boots.

You configure the management interface in `/etc/network/interfaces`. In the example below, the management interface, `eth0`, and the mgmt VRF stanzas are added to the `mgmt` interface class:

```
auto lo
iface lo inet loopback

allow-mgmt eth0
iface eth0 inet dhcp
    vrf mgmt

allow-mgmt mgmt
iface mgmt
    address 127.0.0.1/8
    vrf-table auto
```

When you run `ifupdown2` commands against the interfaces in the `mgmt` class, include `--allow=mgmt` with the commands. For example, to see which interfaces are in the `mgmt` interface class, run:

```
cumulus@switch:~$ ifquery l --allow=mgmt
eth0
mgmt
```

To reload the configurations for interfaces in the `mgmt` class, run:

```
cumulus@switch:~$ sudo ifreload --allow=mgmt
```

However, you can still bring the management interface up and down using `ifup eth0` and `ifdown eth0`.

Management VRF and DNS

Cumulus Linux supports both DHCP and static DNS entries over management VRF through IP FIB rules. These rules are added to direct lookups to the DNS addresses out of the management VRF. However, nameservers configured through DHCP are automatically updated, while statically configured nameservers (configured in `/etc/resolv.conf`) only get updated when you run `ifreload -a`.



Because DNS lookups are forced out of the management interface using FIB rules, this could affect data plane ports if there are overlapping addresses.

Incompatibility with cl-ns-mgmt



Management VRF has replaced the management namespace functionality in Cumulus Linux. The management namespace feature (via the `cl-ns-mgmt` utility) has been deprecated, and the `cl-ns-mgmt` command has been removed.

Protocol Independent Multicast - PIM

Protocol Independent Multicast (PIM) is a multicast control plane protocol, that advertises multicast sources and receivers over a routed layer 3 network. Layer 3 multicast relies on PIM to advertise information about multicast capable routers and the location of multicast senders and receivers. For this reason, multicast cannot be sent through a routed network without PIM.

PIM has two modes of operation: Sparse Mode (PIM-SM) and Dense Mode (PIM-DM).



Cumulus Linux only supports PIM Sparse Mode.

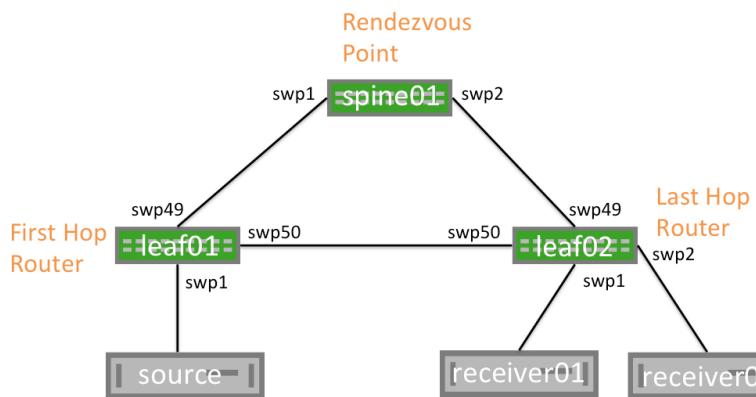
Contents

This chapter covers ...

- PIM Overview (see page 602)
 - PIM Messages (see page 603)
 - PIM Neighbors (see page 605)
- PIM Sparse Mode (PIM-SM) (see page 605)
 - Any-source Multicast Routing (see page 606)
 - PIM Null-Register (see page 609)
- Configuration (see page 609)
 - Getting Started (see page 609)
 - Multicast Source Discovery Protocol (MSDP) (see page 613)
 - Verifying PIM (see page 614)
- Troubleshooting PIM (see page 618)
 - FHR Stuck in Registering Process (see page 618)
 - No *,G Is Built on LHR (see page 619)
 - No mroute Created on FHR (see page 620)
 - No S,G on RP for an Active Group (see page 621)
 - No mroute Entry Present in Hardware (see page 621)
 - Verify MSDP Session State (see page 622)

- View the Active Sources (see page 622)
- Caveats and Errata (see page 622)

PIM Overview



Network Element	Description
First Hop Router (FHR)	The FHR is the first router attached closest to the source. The FHR is responsible for the PIM register process. Each multicast source will have a single FHR.
Last Hop Router (LHR)	The LHR is the last router in the path, attached to an interested multicast receiver. There is a single LHR for each network subnet with an interested receiver, however multicast groups can have multiple LHRs throughout the network.
Rendezvous Point (RP)	The RP allows for the discovery of multicast sources and multicast receivers. The RP is responsible for sending PIM Register Stop messages to FHRs.
PIM Shared Tree (RP Tree) or (*, G) Tree	The Shared Tree is the multicast tree rooted at the RP. When receivers wish to join a multicast group, messages are sent along the shared tree towards the RP.
PIM Shortest Path Tree (SPT) or (S,G) Tree	The SPT is the multicast tree rooted at the multicast source for a given group. Each multicast source will have a unique SPT. The SPT may match the RP Tree, but this is not a requirement. The SPT represents the most efficient way to send multicast traffic from a source to the interested receivers.
Outgoing Interface (OIF)	The Outgoing interface indicates the interface a PIM or multicast packet should be sent on. OIFs are the interfaces towards the multicast receivers.



Network Element	Description
Incoming Interface (IIF)	The Incoming Interface indicates the interface a PIM or multicast packet should be received on. IIFs can be the interfaces towards the multicast, or towards the RP.
Reverse Path Forwarding Interface (RPF Interface)	Reverse Path Forwarding is the unicast route towards a source or receiver.
Multicast Route (mroute)	A multicast route indicates the multicast source and multicast group as well as associated OIFs, IIFs, and RPF information.
Star-G mroute (*,G)	The (*,G) mroute represents the RP Tree. The * is a wildcard indicating any multicast source. The G is the multicast group. An example (*,G) would be (*, 239.1.2.9).
S-G mroute (S,G)	This is the mroute representing the SPT. The S is the multicast source IP. The G is the multicast group. An example (S,G) would be (10.1.1.1, 239.1.2.9).

PIM Messages

PIM Message	Description
PIM Hello	<p>PIM hellos announce the presence of a multicast router on a segment. PIM hellos are sent every 30 seconds by default.</p> <p>PIM Hello Example</p> <pre>22.1.2.2 > 224.0.0.13: PIMv2, length 34 Hello, cksum 0xfd8b (correct) Hold Time Option (1), length 2, Value: 1m45s 0x0000: 0069 LAN Prune Delay Option (2), length 4, Value: T-bit=0, LAN delay 500ms, Override interval 2500ms 0x0000: 01f4 09c4 DR Priority Option (19), length 4, Value: 1 0x0000: 0000 0001 Generation ID Option (20), length 4, Value: 0x2459b190 0x0000: 2459 b190</pre>



PIM Message	Description
PIM Join /Prune (J/P)	<p>PIM J/P messages indicate the groups that a multicast router would like to receive or no longer receive. Often PIM Join/Prune messages are described as distinct message types, but are actually a single PIM message with a list of groups to join and a second list of groups to leave. PIM J/F messages can be to join or prune from the SPT or RP trees (also called (*,G) Joins or (S,G) Joins).</p> <p>⚠️ PIM Join/Prune messages are sent to PIM neighbors on individual interfaces. Join/Prune messages are never unicast.</p> <pre>▶ Internet Protocol Version 4, Src: 10.4.1.1, Dst: 224.0.0.1 ▼ Protocol Independent Multicast 0010 = Version: 2 0011 = Type: Join/Prune (3) Reserved byte(s): 00 Checksum: 0x8dd9 [correct] ▼ PIM Options Upstream-neighbor: 10.4.1.2 Reserved byte(s): 00 Num Groups: 1 Holdtime: 210 ▼ Group 0: 239.1.1.9/32 ▼ Num Joins: 1 IP address: 104.255.224.1/32 (SWR) Num Prunes: 0</pre> <p>This PIM Join/Prune is for group 239.1.1.9, with 1 Join and 0 Prunes for the group. Join/Prune multiple groups can exist in a single packet.</p> <p>S,G Prune Example</p> <pre>21:49:59.470885 IP (tos 0x0, ttl 255, id 138, offset 0, flags [none], proto PIM (103), length 54) 22.1.2.2 > 224.0.0.13: PIMv2, length 34 Join / Prune, cksm 0xb9e5 (correct), upstream-neighbor: 22.1.2.1 1 group(s), holdtime: 3m30s group #1: 225.1.0.0, joined sources: 0, pruned sources: 1 pruned source #1: 33.1.1.1(S)</pre>
PIM Register	PIM register messages are unicast packets sent from a FHR destined to the RP to advertise a multicast group. The FHR fully encapsulates the original multicast packet in a PIM register message. The RP is responsible for decapsulating the PIM register message and forwarding along the (*,G) tree towards the receivers.



PIM Message	Description
PIM Null Register	PIM Null Register is a special type of PIM Register message where the "Null-Register" flag is set within the packet. Null Register messages are used for a FHR to signal to an RP that a source is still sending multicast traffic. Unlike normal PIM Register messages Null Register messages do not encapsulate the original data packet.
PIM Register Stop	PIM Register Stop messages are sent by an RP to the FHR to indicate that PIM Register messages should no longer be sent.
	<p>Register Stop Example</p> <pre>21:37:00.419379 IP (tos 0x0, ttl 255, id 24, offset 0, flags [none], proto PIM (103), length 38) 100.1.2.1 > 33.1.1.10: PIMv2, length 18 Register Stop, cksum 0xd8db (correct) group=225.1.0.0 source=3.1.1.1</pre>
IGMP Membership Report (IGMP Join)	IGMP Membership Reports are sent by multicast receivers to tell multicast routers of their interest in a specific multicast group. IGMP Join messages trigger PIM *G Joins. IGMP version 1 and 2 messages are sent to the All Hosts multicast address, 224.0.0.1. IGMP version 3 messages are sent to an IGMP v3 specific multicast address, 224.0.0.22.
IGMP Leave	IGMP Leaves tell a multicast router that a multicast receiver no longer wants the multicast group. IGMP Leave messages trigger PIM *G Prunes.

PIM Neighbors

When PIM is configured on an interface, PIM Hello messages are sent to the link local multicast group 224.0.0.13. Any other router configured with PIM on the segment that hears the PIM Hello messages will build a PIM neighbor with the sending device.



PIM neighbors are stateless. No confirmation of neighbor relationship is exchanged between PIM endpoints.

PIM Sparse Mode (PIM-SM)

PIM Sparse Mode (PIM-SM) is a "pull" multicast distribution method. This means that multicast traffic is only sent through the network if receivers explicitly ask for it. When a receiver "pulls" multicast traffic, the network must be periodically notified that the receiver wishes to continue the multicast stream.



This behavior is in contrast to PIM Dense Mode (PIM-DM), where traffic is flooded, and the network must be periodically notified that the receiver wishes to stop receiving the multicast stream.

PIM-SM has three configuration options: Any-source Multicast (ASM), Bi-directional Multicast (BiDir), and Source Specific Multicast (SSM):

- Any-source Multicast (ASM) is the traditional, and most commonly deployed PIM implementation. ASM relies on Rendezvous Points to connect multicast senders and receivers that then dynamically determine the shortest path through the network between source and receiver, to efficiently send multicast traffic.
- Bidirectional PIM (BiDir) forwards all traffic through the multicast Rendezvous Point (RP), rather than tracking multicast source IPs, allowing for greater scale, while resulting in inefficient forwarding of network traffic.
- Source Specific Multicast (SSM) requires multicast receivers to know exactly which source they wish to receive multicast traffic from, rather than relying on multicast Rendezvous Points. SSM requires the use of IGMPv3 on the multicast clients.



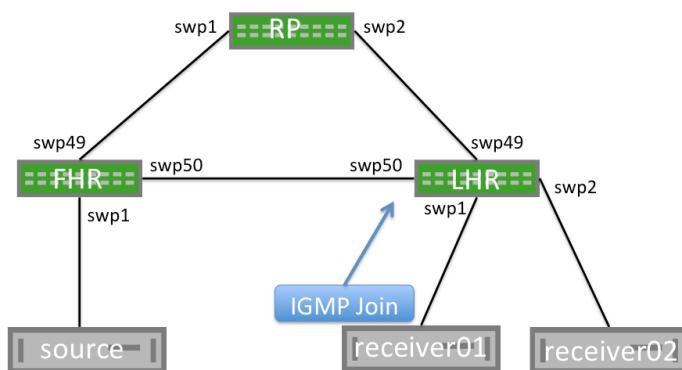
Cumulus Linux only supports Any-source Multicast. PIM SSM and PIM BiDir are not currently supported.

Any-source Multicast Routing

Multicast routing behaves differently depending on whether the source is sending before receivers request the multicast stream, or if a receiver tries to join a stream before there are any sources.

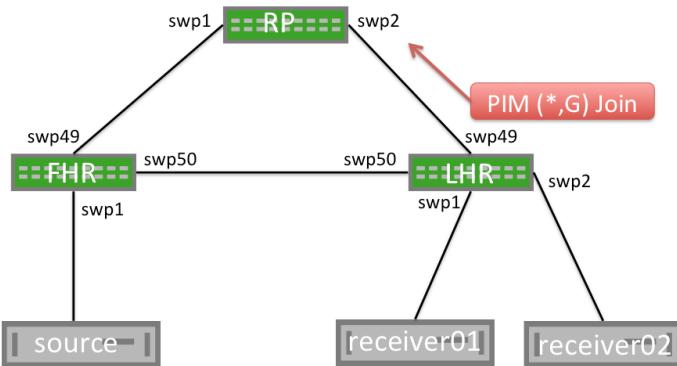
Receiver Joins First

When a receiver joins a group, an IGMP Membership Join message is sent to the IGMPv3 multicast group, 224.0.0.22. The PIM multicast router for the segment, listening to the IGMPv3 group, receives the IGMP Membership Join message, and becomes an LHR for this group.



This creates a $(*, G)$ mroute, with an OIF of the interface on which the IGMP Membership Report was received and an IIF of the RPF interface for the RP.

The LHR generates a PIM $(*,G)$ Join message, and sends it from the interface towards the RP. Each multicast router between the LHR and the RP will build a $(*,G)$ mroute with the OIF being the interface on which the PIM Join message was received and an Incoming Interface of the Reverse Path Forwarding interface for the RP.



When the RP receives the $(*, G)$ Join message, it will not send any additional PIM Join messages. The RP will maintain a $(*, G)$ state as long as the receiver wishes to receive the multicast group.

Unlike multicast receivers, multicast sources do not send IGMP (or PIM) messages to the FHR. A multicast source begins sending and the FHR will receive the traffic and build both a $(*, G)$ and an (S, G) mroute. The FHR will then begin the PIM Register process.

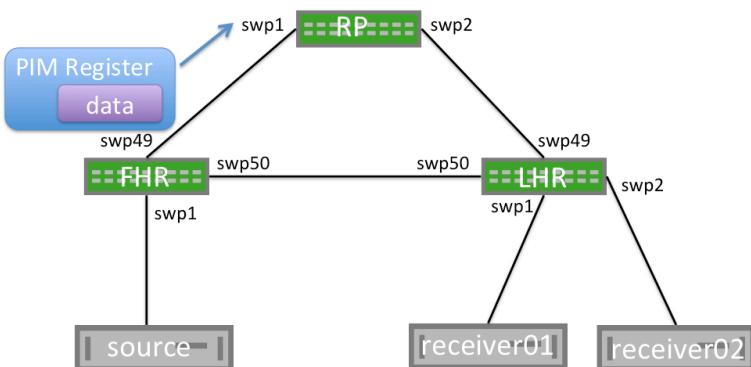
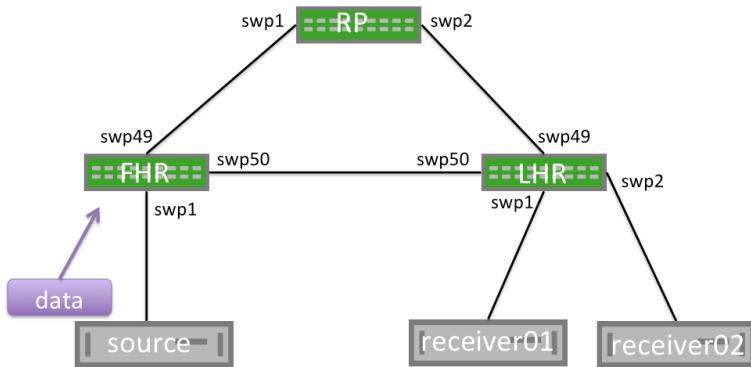
PIM Register Process

When a First Hop Router (FHR) receives a multicast data packet from a source, the FHR does not know if there are any interested multicast receivers in the network. The FHR encapsulates the data packet in a unicast PIM register message. This packet is sourced from the FHR and destined to the RP address. The RP will build an (S,G) mroute and decapsulate the multicast packet and forward it along the $(*,G)$ tree.

As the unencapsulated multicast packet travels down the $(*,G)$ tree towards the interested receivers. At the same time, the RP will send a PIM (S,G) Join towards the FHR. This will build an (S,G) state on each multicast router between the RP and FHR.

When the FHR receives a PIM (S,G) Join, it will continue encapsulating and sending PIM Register messages, but will also make a copy of the packet and send it along the (S,G) mroute.

The RP then receives the multicast packet along the (S,G) tree and sends a PIM Register Stop to the FHR to end the register process.



PIM SPT Switchover

When the LHR receives the first multicast packet, in order to efficiently forward traffic through the network, it will send a PIM (S,G) Join towards the FHR. This builds the Shortest Path Tree (SPT), or the tree that is the shortest path to the source.

When the traffic arrives over the SPT, a PIM (S,G) Prune will be sent up the Shared Tree towards the RP. This removes multicast traffic from the shared tree; multicast data will only be sent over the SPT.

The LHR will now send both (*,G) Joins and (S,G) Prune messages towards the RP.



Cumulus Linux PIM does not currently support SPT Switchover. All traffic will be forwarded along the shared tree.



Sender Starts Before Receivers Join

As previously mentioned, a multicast sender can send multicast data without any additional IGMP or PIM signaling. When the FHR receives the multicast traffic it will encapsulate it and send a PIM Register to the Rendezvous Point (RP).

When the RP receives the PIM Register, it will build an (S,G) mroute; however, there is no (*,G) mrouter and no interested receivers.

The RP will drop the PIM Register message and immediately send a PIM Register Stop message to the FHR.

Receiving a PIM Register Stop without any associated PIM Joins leaves the FHR without any outgoing interfaces. The FHR will drop this multicast traffic until a PIM Join is received.



PIM Register messages are sourced from the interface that received the multicast traffic and are destined to the RP address. The PIM Register is not sourced from the interface towards the RP.

PIM Null-Register

In order to notify the RP that multicast traffic is still flowing when the RP has no receiver, or if the RP is not on the SPT tree, the FHR will periodically send PIM Null Register messages. The FHR sends a PIM Register with the Null-Register flag set, but without any data. This special PIM Register notifies the RP that a multicast source is still sending, should any new receivers come online.

After receiving a PIM Null-Register, the RP immediately sends a PIM Register Stop to acknowledge the reception of the PIM Null Register message.

Configuration

Getting Started

The `cumulus-pim` package is included in Quagga. To configure PIM on a switch:

1. Open `/etc/quagga/daemons` in a text editor.
2. Add the following line to the end of the file to enable `pimd`, and save the file:

```
zebra=yes  
pimd=yes
```

3. Run the `sudo systemctl restart quagga` command to restart Quagga:

```
cumulus@switch:~$ sudo systemctl restart quagga
```

4. In a terminal, run the `vtysh` command to start the Quagga CLI on the switch.

```
cumulus@switch:~$ sudo vtysh  
cumulus#
```

5. Run the following commands to enable multicast routing:

```
cumulus# configure terminal  
cumulus(config)# ip multicast-routing
```

6. Run the following commands to configure the PIM interfaces:

```
cumulus# configure terminal  
cumulus(config)# int swp1  
cumulus(config-if)# ip pim sm
```



PIM must be enabled on all interfaces facing multicast sources or multicast receivers, as well as on the interface where the RP address is configured.

7. Run the following commands to enable either IGMPv2 or IGMPv3 on the interfaces with hosts attached:

```
cumulus# configure terminal  
cumulus(config)# int swp1  
cumulus(config-if)# ip igmp version 3
```



IGMP must be configured on all interfaces where multicast receivers exist.

8. Configure a group mapping for a static RP:

```
cumulus# configure terminal  
cumulus(config)# ip pim rp 192.168.0.1 224.0.0.0/4
```



Each PIM-SM enabled device must configure a static RP to a group mapping, and all PIM-SM enabled devices must have the same RP to group mapping configuration.

Complete Multicast Network Configuration Example

The following is example configuration

RP Configuration

```
RP# show run  
Building configuration...  
Current configuration:
```

```
!
log syslog
ip multicast-routing
ip pim rp 192.168.0.1 224.0.0.0/4
username cumulus nopassword
!
!
interface lo
description RP Address interface
ip ospf area 0.0.0.0
ip pim sm
!
interface swp1
description interface to FHR
ip ospf area 0.0.0.0
ip ospf network point-to-point
ip pim sm
!
interface swp2
description interface to LHR
ip ospf area 0.0.0.0
ip ospf network point-to-point
ip pim sm
!
router ospf
ospf router-id 192.168.0.1
!
line vty
!
end
```

FHR Configuration

```
FHR# show run
!
log syslog
ip multicast-routing
ip pim rp 192.168.0.1 224.0.0.0/4
username cumulus nopassword
!
interface bridge10.1
description Interface to multicast source
ip ospf area 0.0.0.0
ip ospf network point-to-point
ip pim sm
!
interface lo
ip ospf area 0.0.0.0
ip pim sm
!
```



```
interface swp49
description interface to RP
ip ospf area 0.0.0.0
ip ospf network point-to-point
ip pim sm
!
interface swp50
description interface to LHR
ip ospf area 0.0.0.0
ip ospf network point-to-point
ip pim sm
!
router ospf
ospf router-id 192.168.1.1
!
line vty
!
end
```

LHR Configuration

```
LHR# show run
!
log syslog
ip multicast-routing
ip pim rp 192.168.0.1 224.0.0.0/4
username cumulus nopassword
!
interface bridge0.1
description interface to multicast receivers
ip igmp
ip ospf area 0.0.0.0
ip ospf network point-to-point
ip pim sm
!
interface lo
ip ospf area 0.0.0.0
ip pim sm
!
interface swp49
description interface to RP
ip ospf area 0.0.0.0
ip ospf network point-to-point
ip pim sm
!
interface swp50
description interface to FHR
ip ospf area 0.0.0.0
ip ospf network point-to-point
ip pim sm
```

```
!
router ospf
  ospf router-id 192.168.2.2
!
line vty
!
end
```

Multicast Source Discovery Protocol (MSDP)

The Multicast Source Discovery Protocol (MSDP) is used to connect multiple PIM-SM multicast domains together, using the PIM-SM RPs. By configuring any cast RPs with the same IP address on multiple multicast switches (primarily on the loopback interface), the PIM-SM limitation of only one RP per multicast group is relaxed. This allows for an increase in both failover and load-balancing throughout.

When an RP discovers a new source (typically a PIM-SM register message via TCP), a source-active (SA) message is sent to each MSDP peer. The peer then determines if any receivers are interested.



Cumulus Linux MSDP support is primarily for anycast-RP configuration, rather than multiple multicast domains. Each MSDP peer must be configured in a full mesh, as SA messages are not received and re-forwarded.



Cumulus Linux currently only supports one MSDP mesh-group.

Configuration

The steps below cover configuring a Cumulus switch to use the MSDP

1. Add an anycast IP address to the loopback interface for each RP in the domain:

```
cumulus@switch:$ net add loopback lo ip address 10.1.1.1/32
cumulus@switch:$ net add loopback lo ip address 10.1.1.100/32
cumulus@switch:$ net pending
cumulus@switch:$ net commit
```

2. On every multicast switch, configure the group to RP mapping using the anycast address:

```
cumulus@switch:$ net add pim rp 100.1.1.100 224.0.0.0/4
cumulus@switch:$ net pending
cumulus@switch:$ net commit
```

3. Log into the Quagga CLI:

```
cumulus@switch:$ sudo vtysh
```

- Configure the MSDP mesh group for all active RPs:



The mesh group should include all RPs in the domain as members, with a unique address as the source. This configuration will result in MSDP peerings between all RPs.

```
switch# conf t
switch(config)# ip msdp mesh-group cumulus source 100.1.1.1
switch(config)# ip msdp mesh-group cumulus source 100.1.1.2
switch(config)# ip msdp mesh-group cumulus source 100.1.1.3
```

- Inject the anycast IP address into the domain's IGP.



If the network is unnumbered and uses unnumbered BGP as the IGP, avoid using the anycast IP address for establishing unicast or multicast peerings. For PIM-SM, ensure that the unique address is used as the PIM hello source by setting the source:

```
cumulus@switch:$ sudo vtysh
switch# conf t
switch(config)# interface lo
switch(config-if)# ip pim use-source 100.1.1.1
```

Verifying PIM



The following outputs are based on the [Cumulus Reference Topology](#) with cldemo-pim.

Source Starts First

On the FHR, an mroute is built, but the upstream state is "Prune". The FHR flag is set on the interface receiving multicast.

Use the `show ip mroute` command to review detailed output for the FHR:

```
exit01# show ip mroute
Source          Group          Proto   Input      Output     TTL
Uptime
172.16.5.105   239.1.1.1    none    br0       none      0        --:
--:--
!
exit01# show ip pim upstream
Iif Source Group State Uptime JoinTimer RSTimer KATimer RefCnt
```



```
br0 172.16.5.105 239.1.1.1 Prune 00:07:40 --:--- 00:00:36 00:02:50 1
!
exit01# show ip pim upstream-join-desired
Interface Source Group LostAssert Joins PimInclude
JoinDesired EvalJD
!
exit01# show ip pim interface
Interface State Address PIM Nbrs PIM DR FHR
br0 up 172.16.5.1 0 local 1
swp51 up 10.1.0.17 1 local 0
swp52 up 10.1.0.19 0 local 0
!
exit01# show ip pim state
Source Group IIF OIL
172.16.5.105 239.1.1.1 br0
!
exit01# show ip pim int detail
Interface : br0
State : up
Address : 172.16.5.1
Designated Router
-----
Address : 172.16.5.1
Priority : 1
Uptime : --:---:
Elections : 2
Changes : 0

FHR - First Hop Router
-----
239.1.1.1 : 172.16.5.105 is a source, uptime is 00:27:43
```

On the Spine, no mroute state is created, but the `show ip pim upstream` output includes the S,G:

```
spine01# show ip mroute
Source Group Proto Input Output TTL
Uptime
!
spine01# show ip pim upstream
Iif Source Group State Uptime
JoinTimer RSTimer KATimer RefCnt
swp30 172.16.5.105 239.1.1.1 Prune 00:00:19 --:--:
-- --:---: 00:02:46 1
```

As a receiver joins the group, the mroute Output interface on the FHR transitions from "none" to the RPF interface of the RP:

```
exit01# show ip mroute
```

```

Source          Group          Proto  Input   Output   TTL
Uptime
172.16.5.105  239.1.1.1    PIM    br0     swp51    1      00:0
5:40
!
exit01# show ip pim upstream
Iif      Source          Group          State   Uptime
JoinTimer RSTimer      KATimer      RefCnt
br0      172.16.5.105  239.1.1.1    Prune   00:48:23 --:--
-- 00:00:00 00:00:37      2
!
exit01# show ip pim upstream-join-desired
Interface Source          Group          LostAssert Joins PimInclude
JoinDesired EvalJD
swp51    172.16.5.105  239.1.1.1    no       yes     no
yes      yes
!
exit01# show ip pim state
Source          Group          IIF      OIL
172.16.5.105  239.1.1.1    br0     swp51

```

```

spine01# show ip mroute
Source          Group          Proto  Input   Output   TTL
Uptime
*              239.1.1.1    PIM    lo      swp1    1      00:0
9:59
172.16.5.105  239.1.1.1    PIM    swp30   swp1    1      00:0
9:59
!
spine01# show ip pim upstream
Iif      Source          Group          State   Uptime
JoinTimer RSTimer      KATimer      RefCnt
lo      *              239.1.1.1    Joined   00:10:01 00:00:5
9  --:---:--- --:---:--- 1
swp30    172.16.5.105  239.1.1.1    Joined   00:00:01 00:00:5
9  --:---:--- 00:02:35      1
!
spine01# show ip pim upstream-join-desired
Interface Source          Group          LostAssert Joins PimInclude
JoinDesired EvalJD
swp1    *              239.1.1.1    no       yes     no
yes      yes
!
spine01# show ip pim state
Source          Group          IIF      OIL
*              239.1.1.1    lo      swp1
172.16.5.105  239.1.1.1    swp30   swp1

```



Receiver Joins First

On the LHR attached to the receiver:

```
leaf01# show ip mroute
Source          Group          Proto  Input       Output      TTL
Uptime
*           239.2.2.2        IGMP    swp51      br0        1      00:0
1:19
!
leaf01# show ip pim local-membership
Interface Address          Source          Group          Membership
br0      172.16.1.1        *           239.2.2.2      INCLUDE
!
leaf01# show ip pim state
Source          Group          IIF     OIL
*           239.2.2.2        swp51     br0
!
leaf01# show ip pim upstream
Iif      Source          Group          State          Uptime
JoinTimer RSTimer   KATimer   RefCnt
swp51    *           239.2.2.2      Joined        00:02:07 00:00:5
3  ---:---  ---:---      1
!
leaf01# show ip pim upstream-join-desired
Interface Source          Group          LostAssert  Joins PimInclude
JoinDesired EvalJD
br0      *           239.2.2.2      no          no     yes
yes      yes
!
leaf01# show ip igmp groups
Interface Address          Group          Mode   Timer      Srcs V Uptime
br0      172.16.1.1        239.2.2.2    EXCL 00:04:02   1 3 00:04:1
2
!
leaf01# show ip igmp sources
Interface Address          Group          Source          Timer Fwd
Uptime
br0      172.16.1.1        239.2.2.2      *            03:54     Y 00
:04:21
```

On the RP:

```
spine01# show ip mroute
Source          Group          Proto  Input       Output      TTL
Uptime
*           239.2.2.2        PIM     lo         swp1        1      00:0
0:03
!
spine01# show ip pim state
```

```

Source          Group          IIF      OIL
*              239.2.2.2       lo       swp1
!
spine01# show ip pim upstream
Iif      Source          Group          State      Uptime
JoinTimer RSTimer      KATimer      RefCnt
lo      *              239.2.2.2     Joined    00:05:17 00:00:4
3  ---:---  ---:---      1
!
spine01# show ip pim upstream-join-desired
Interface Source          Group          LostAssert Joins PimInclude
JoinDesired EvalJD
swp1      *              239.2.2.2     no         yes     no
yes      yes

```

Troubleshooting PIM

FHR Stuck in Registering Process

When a multicast source starts, the FHR sends unicast PIM register messages from the RPF interface towards the source. After the PIM register is received by the RP, a `PIM register stop` message is sent from the RP to the FHR to end the register process. If an issue with this communication, the FHR will remain stuck in the registering process, which can result in high CPU, as PIM register packets are generated by the FHR CPU, and sent to the RP CPU.

To assess this issue:

- Review the FHR. The output interface of `pimreg` can be seen here. If this does not change to an interface within a few seconds, the FHR is likely stuck.

```

exit01# show ip mroute
Source          Group          Proto   Input        Output
TTL   Uptime
172.16.5.105  239.2.2.3      PIM     br0        pimreg      1
00:03:59

```

To troubleshoot the issue:

1. Validate that the FHR can reach the RP. If the RP and FHR can not communicate, the Registration process will fail:

```

cumulus@exit01:~$ ping 10.0.0.21 -I br0
PING 10.0.0.21 (10.0.0.21) from 172.16.5.1 br0: 56(84) bytes of
data.
^C
--- 10.0.0.21 ping statistics ---
4 packets transmitted, 0 received, 100% packet loss, time 3000ms

```

2. On the RP, use `tcpdump` to see if the PIM Register packets are arriving:



```
cumulus@spine01:~$ sudo tcpdump -i swp30
tcpdump: verbose output suppressed, use -v or -vv for full
protocol decode
listening on swp30, link-type EN10MB (Ethernet), capture size 262
144 bytes
23:33:17.524982 IP 172.16.5.1 > 10.0.0.21: PIMv2, Register,
length 66
```

3. If PIM Registration packets are being received, verify that they are seen by PIM by issuing debug pim packets from within Quagga:

```
cumulus@spine01:~$ sudo vtysh -c "debug pim packets"
PIM Packet debugging is on

cumulus@spine01:~$ sudo tail /var/log/quagga/quagga.log
2016/10/19 23:46:51 PIM: Recv PIM REGISTER packet from 172.16.5.1
to 10.0.0.21 on swp30: ttl=255 pim_version=2 pim_msg_size=64
checksum=a681
```

4. Repeat the process on the FHR to see if PIM Register Stop messages are being received on the FHR and passed to the PIM process:

```
cumulus@exit01:~$ sudo tcpdump -i swp51
23:58:59.841625 IP 172.16.5.1 > 10.0.0.21: PIMv2, Register,
length 28
23:58:59.842466 IP 10.0.0.21 > 172.16.5.1: PIMv2, Register Stop,
length 18

cumulus@exit01:~$ sudo vtysh -c "debug pim packets"
PIM Packet debugging is on

cumulus@exit01:~$ sudo tail -f /var/log/quagga/quagga.log
2016/10/19 23:59:38 PIM: Recv PIM REGSTOP packet from 10.0.0.21
to 172.16.5.1 on swp51: ttl=255 pim_version=2 pim_msg_size=18
checksum=5a39
```

No *,G Is Built on LHR

The most common reason for a *,G to not be built on a LHR is for both PIM **and** IGMP to not be enabled on an interface facing a receiver.

```
leaf01# show run
!
interface br0
  ip igmp
```



```
ip ospf area 0.0.0.0
ip pim sm
```

To troubleshoot this issue:

1. If both PIM and IGMP are enabled, ensure that IGMPv3 Joins are being sent by the receiver:

```
cumulus@leaf01:~$ sudo tcpdump -i br0 igmp
tcpdump: verbose output suppressed, use -v or -vv for full
protocol decode
listening on br0, link-type EN10MB (Ethernet), capture size 26214
 4 bytes
00:03:55.789744 IP 172.16.1.101 > igmp.mcast.net: igmp v3
report, 1 group record(s)
```

No mroute Created on FHR

To troubleshoot this issue:

1. Verify that multicast traffic is being received:

```
cumulus@exit01:~$ sudo tcpdump -i br0
tcpdump: verbose output suppressed, use -v or -vv for full
protocol decode
listening on br0, link-type EN10MB (Ethernet), capture size 26214
 4 bytes
00:11:52.944745 IP 172.16.5.105.51570 > 239.2.2.9.1000: UDP,
length 9
```

2. Verify that PIM is configured on the interface facing the source:

```
exit01# show run
!
interface br0
  ip ospf area 0.0.0.0
  ip pim sm
```

- a. If PIM is configured, verify that the RPF interface for the source matches the interface the multicast traffic is received on:

```
exit01# show ip rpf 172.16.5.105
Routing entry for 172.16.5.0/24 using Multicast RIB
  Known via "connected", distance 0, metric 0, best
    * directly connected, br0
```

3. Verify that an RP is configured for the multicast group:



```
exit01# show ip pim rp-info
RP address          group/prefix-list      OIF           I am RP
 10.0.0.21          224.0.0.0/4            swp51         no
```

No SG on RP for an Active Group

An RP will not build an mroute when there are no active receivers for a multicast group, even though the mroute was created on the FHR:

```
spine01# show ip mroute
Source          Group          Proto  Input        Output       TTL
Uptime
spine01#
```

```
exit01# show ip mroute
Source          Group          Proto  Input        Output       TTL
Uptime
172.16.5.105   239.2.2.9    none   br0         none        0      --:
--:--
```

This is expected behavior. The active source can be seen on the RP with `show ip pim upstream`:

```
spine01# show ip pim upstream
Iif      Source          Group          State        Uptime
JoinTimer RSTimer      KATimer      RefCnt
swp30    172.16.5.105   239.2.2.9   Prune       00:08:03  --:--
-- 00:02:20      1
!
spine01# show ip mroute
Source          Group          Proto  Input        Output       TTL
Uptime
spine01#
```

No mroute Entry Present in Hardware

Please verify that the hardware IP multicast entry is the maximum value already, using the `cl-resource-query` command:

```
cumulus@switch:~$ cl-resource-query | grep Mcast
Total Mcast Routes:      450,  0% of maximum value  450
```

For Mellanox chipsets, please refer to [TCAM Resource Profiles for Mellanox Switches](#) (see page 481).



Verify MSDP Session State

Run the following commands to verify the state of MSDP sessions:

```
switch# show ip msdp mesh-group
Mesh group : pod1
  Source : 100.1.1.1
    Member           State
    100.1.1.2      established
    100.1.1.3      established
spine-1# show ip msdp peer
Peer             Local          State       Uptime   SaCnt
100.1.1.2        100.1.1.1    established  00:07:21   0
100.1.1.3        100.1.1.1    established  00:07:21   0
spine-1#
```

View the Active Sources

Review the active sources learned locally (via PIM registers) and from MSDP peers:

```
switch# show ip msdp sa
Source                 Group          RP     Local   SPT
Uptime
44.1.11.2            239.1.1.1    100.1.1.1    n     n  00:00:4
0
44.1.11.2            239.1.1.2    100.1.1.1    n     n  00:00:2
5
spine-2#
```

Caveats and Errata

- Cumulus Linux 3.2.0 only supports PIM Sparse Mode - Any-source Multicast (PIM-SM ASM). Dense Mode, Bidirectional Multicast, and Source-specific Multicast are not supported.
- ECMP with multicast traffic is not implemented; all multicast traffic will utilize a single link.
- ECMP RPF is not supported; only a single path towards the source is supported. Traffic may be dropped for RPF failure if ECMP paths towards the source exist.
- SPT switchover is not supported. All traffic will flow over the shared tree.
- S,G mroutes are not build on routers that are not the Rendezvous Point (RP) or the First-hop Router (FHR). S,G PIM Joins will be sent, but only *,G mroutes are built. As a result, all traffic will flow over the *,G tree, similar to PIM Bidirectional Multicast.



- Non-native forwarding (register decapsulation) is not supported. Initial packet loss is expected while the PIM \ast, G tree is built from the Rendezvous Point (RP) to the First-hop Router (FHR) to trigger native forwarding.
- Cumulus Linux does not currently build an S, G mroute when forwarding over an \ast, G tree.



Monitoring and Troubleshooting

This chapter introduces monitoring and troubleshooting Cumulus Linux.

Contents

This chapter covers ...

- Using the Serial Console (see page 624)
 - Configuring the Serial Console on ARM Switches (see page 624)
 - Configuring the Serial Console on x86 Switches (see page 625)
- Getting General System Information (see page 626)
- Diagnostics Using cl-support (see page 626)
- Sending Log Files to a syslog Server (see page 627)
 - Local Logging (see page 627)
 - Enabling Remote syslog (see page 628)
 - Writing to syslog with Management VRF Enabled (see page 629)
 - Advanced Logging (see page 629)
 - Rate-limiting syslog Messages (see page 630)
 - Harmless syslog Error: Failed to reset devices.list (see page 631)
- Next Steps (see page 631)

Using the Serial Console

The serial console can be a useful tool for debugging issues, especially when you find yourself rebooting the switch often or if you don't have a reliable network connection.

The default serial console baud rate is 115200, which is the baud rate ONIE uses.

Configuring the Serial Console on ARM Switches

On ARM switches, the U-Boot environment variable `baudrate` identifies the baud rate of the serial console. To change the `baudrate` variable, use the `fw_setenv` command:

```
cumulus@switch:~$ sudo fw_setenv baudrate 9600
Updating environment variable: `baudrate'
Proceed with update [N/y]? y
```

You must reboot the switch for the `baudrate` change to take effect.

The valid values for `baudrate` are:

- 300



- 600
- 1200
- 2400
- 4800
- 9600
- 19200
- 38400
- 115200

Configuring the Serial Console on x86 Switches

On x86 switches, you configure serial console baud rate by editing `grub`.



Incorrect configuration settings in `grub` can cause the switch to be inaccessible via the console. Grub changes should be carefully reviewed before implementation.

The valid values for the baud rate are:

- 300
- 600
- 1200
- 2400
- 4800
- 9600
- 19200
- 38400
- 115200

To change the serial console baud rate:

1. Edit `/etc/default/grub`. The two relevant lines in `/etc/default/grub` are as follows; replace the 115200 value with a valid value specified above in the `--speed` variable in the first line and in the `console` variable in the second line:

```
GRUB_SERIAL_COMMAND="serial --port=0x2f8 --speed=115200 --word=8  
--parity=no --stop=1"  
GRUB_CMDLINE_LINUX="console=ttyS1,115200n8  
cl_platform=accton_as5712_54x"
```

2. After you save your changes to the grub configuration, type the following at the command prompt:

```
cumulus@switch:~$ update-grub
```



3. If you plan on accessing your switch's BIOS over the serial console, you need to update the baud rate in the switch BIOS. For more information, see [this knowledge base article](#).
4. Reboot the switch.

Getting General System Information

Two commands are helpful for getting general information about the switch and the version of Cumulus Linux you are running. These are helpful with system diagnostics and if you need to submit a support request to Cumulus Networks.

For information about the version of Cumulus Linux running on the switch, run `net show version`, which displays the contents of `/etc/lsb-release`:

```
cumulus@switch:~$ net show version
NCLU_VERSION=1.0
DISTRIB_ID="Cumulus Linux"
DISTRIB_RELEASE=3.2.1~1484951197.337c36a
DISTRIB_DESCRIPTION="Cumulus Linux 3.2.1~1484951197.337c36a"
```

For general information about the switch, run `net show system`, which gathers information about the switch from a number of files in the system:

```
cumulus@switch:~$ net show system
Cumulus 3.2.1~1484951197.337c36a
Build: Cumulus Linux 3.2.1~1484951197.337c36a
Uptime: 34 days, 4:31:46
```

Diagnostics Using cl-support

You can use `cl-support` to generate a single export file that contains various details and the configuration from a switch. This is useful for remote debugging and troubleshooting. For more information about `cl-support`, read [Understanding the cl-support Output File \(see page 643\)](#).

You should run `cl-support` before you submit a support request to Cumulus Networks as this file helps in the investigation of issues.

```
cumulus@switch:~$ sudo cl-support -h
Usage: cl-support [-h] [-s] [-t] [-v] [reason]...

Args:
[reason]: Optional reason to give for invoking cl-support.
          Saved into tarball's cmdline.args file.

Options:
-h: Print this usage statement
-s: Security sensitive collection
-t: User filename tag
```



```
-v: Verbose  
-e MODULES: Enable modules. Comma separated module list (run with -e  
help for module names)  
-d MODULES: Disable modules. Comma separated module list (run with -d  
help for module names)
```

Sending Log Files to a syslog Server

Logging on Cumulus Linux is done with `rsyslog`. `rsyslog` provides both local logging to the `syslog` file as well as the ability to export logs to an external `syslog` server. High precision timestamps are enabled for all `rsyslog` log files; here's an example:

```
2015-08-14T18:21:43.337804+00:00 cumulus switchd[ 3629 ]: switchd.c:  
1409 switchd version 1.0-c12.5+5
```

There are applications in Cumulus Linux that write directly to a log file without going through `rsyslog`. These files are typically located in `/var/log/`.



All Cumulus Linux rules are stored in separate files in `/etc/rsyslog.d/`, which are called at the end of the `GLOBAL DIRECTIVES` section of `/etc/rsyslog.conf`. As a result, the `RULES` section at the end of `rsyslog.conf` is ignored because the messages have to be processed by the rules in `/etc/rsyslog.d` and then dropped by the last line in `/etc/rsyslog.d/99-syslog.conf`.

Local Logging

Most logs within Cumulus Linux are sent through `rsyslog`, which then writes them to files in the `/var/log` directory. There are default rules in the `/etc/rsyslog.d/` directory that define where the logs are written:

Rule	Purpose
10-rules.conf	Sets defaults for log messages, include log format and log rate limits.
15-crit.conf	Logs crit, alert or emerg log messages to <code>/var/log/crit.log</code> to ensure they are not rotated away rapidly.
20-clagd.conf	Logs clagd messages to <code>/var/log/clagd.log</code> for MLAG (see page 300).
25-switchd.conf	Logs switchd messages to <code>/var/log/switchd.log</code> .



Rule	Purpose
30-ptmd.conf	Logs <code>ptmd</code> messages to <code>/var/log/ptmd.log</code> for Prescription Topology Manager (see page 257).
35-rdnbrd.conf	Logs <code>rdnbrd</code> messages to <code>/var/log/rdnbrd.log</code> for redistribute neighbor (see page 564).
40-netd.conf	Logs <code>netd</code> messages to <code>/var/log/netd.log</code> for NCLU (see page 80).
99-syslog.conf	All remaining processes that use <code>rsyslog</code> are sent to <code>/var/log/syslog</code> .

Log files that are rotated are compressed into an archive. Processes that do not use `rsyslog` write to their own log files within the `/var/log` directory. For more information on specific log files, see [Troubleshooting Log Files](#) (see [page 644](#)).

Enabling Remote syslog

If you need to send other log files — such as `switchd` logs — to a `syslog` server, do the following:

1. Create a file in `/etc/rsyslog.d/`. Make sure it starts with a number lower than 99 so that it executes before log messages are dropped in, such as `20-clagd.conf` or `25-switchd.conf`. Our example file is called `/etc/rsyslog.d/11-remotesyslog.conf`. Add content similar to the following:

```
## Logging switchd messages to remote syslog server

if $programname == 'switchd' then @192.168.1.2:514
```

This configuration sends log messages to a remote `syslog` server for the following processes: `clagd`, `switchd`, `ptmd`, `rdnbrd`, `netd` and `syslog`. The `if $programname` line sends the log files to the `syslog` server. It follows the same syntax as the `/var/log/syslog` file, where `@` indicates UDP, `192.168.1.2` is the IP address of the `syslog` server, and `514` is the UDP port.



For TCP-based syslog, use two `@@` before the IP address: `@@192.168.1.2:514`.

Running `syslog` over TCP places a burden on the switch to queue packets in the `syslog` buffer. This may cause detrimental effects if the remote `syslog` server becomes unavailable.



The numbering of the files in `/etc/rsyslog.d/` dictates how the rules are installed into `rsyslog.d`. If you want to remotely log only the messages in `/var/syslog`, and not those in `/var/log/clagd.log` or `/var/log/switchd.log`, for instance, then name the file `98-remotesyslog.conf`, since it's lower than the `/var/syslog` file `99-syslog.conf` only.

2. Restart `rsyslog`.

```
cumulus@switch:~$ sudo systemctl restart rsyslog.service
```

Writing to syslog with Management VRF Enabled

You can write to syslog with [management VRF](#) (see page 593) enabled by applying the following configuration; this configuration is commented out in the `/etc/rsyslog.d/99-syslog.conf` file:

```
## Copy all messages to the remote syslog server at 192.168.1.2 port
514
action(type="omfwd" Target="192.168.1.2" Device="mgmt" Port="514"
Protocol="udp")
```

Advanced Logging

Some applications bypass `rsyslog` and log to a file directly. For example, Quagga logs to `syslog` via the `log syslog` command, but can also log to a file directly. This section demonstrates how you can configure Cumulus Linux to read in from a flat log file and pass it through the `rsyslog` filter engine by using the `imfile` module.

! Never use the `imfile` module to read files that are written by `rsyslog`, as it can cause loops when reading and writing to the same log file.

Configuring advanced logging ...

Create a file and add content similar to the following:

```
module(load="imfile")
input(type="imfile"
      stateFile="quagga-state"
      File="/var/log/quagga/Quagga.log"
      Severity="Warning"
      Tag="quagga-log:"
      Facility="local7")
if $syslogtag contains "quagga-log" then action(type="omfwd" Target="192.168.1.2" Port="514" Protocol="udp")
```



You can find more information in the [rsyslog documentation](#).

Then restart `syslog`:

```
cumulus@switch:~$ sudo systemctl restart rsyslog.service
```

In the above configuration, each setting is defined as follows:

Setting	Description
load	Loads the <code>rsyslog</code> module to watch file contents.
File	The file to be sent through the <code>rsyslog</code> rules engine. In the above example, any changes made are sent to <code>/var/log/switchd.log</code> to the <code>syslog</code> server.
stateFile	<code>rsyslog</code> uses this to track the state of the file being monitored. This must be unique for each file being monitored.
Tag	Defines the <code>syslog</code> tag that precedes the <code>syslog</code> messages. In this example, all logs are prefaced with <code>quagga-log</code> .
Severity	Defines the logging severity level sent to the <code>syslog</code> server.
Facility	Defines the logging format. <code>local7</code> is common.

Rate-limiting syslog Messages

If you want to limit the number of `syslog` messages that can be written to the `syslog` file from individual processes, add the following configuration to `/etc/rsyslog.conf`. Adjust the interval and burst values to rate-limit messages to the appropriate levels required by your environment. For more information, read the [rsyslog documentation](#).

```
module(load="imuxsock"
       SysSock.RateLimit.Interval="2" SysSock.RateLimit.Burst="50")
```

The following test script shows an example of rate-limit output in Cumulus Linux ...

```
root@leaf1:mgmt-vrf:/home/cumulus# cat ./syslog.py
#!/usr/bin/python
import syslog
message_count=100
print "Sending %s Messages..."%(message_count)
for i in range(0,message_count):
    syslog.syslog("Message Number:%s"%(i))
```



```
print "DONE."
root@leaf1:mgmt-vrf:/home/cumulus# ./syslog.py
Sending 100 Messages...
DONE.
root@leaf1:mgmt-vrf:/home/cumulus# tail -n 60 /var/log/syslog
2017-02-22T19:59:50.043342+00:00 leaf1 syslog.py[22830]: Message
Number:0
2017-02-22T19:59:50.043723+00:00 leaf1 syslog.py[22830]: Message
Number:1
2017-02-22T19:59:50.043941+00:00 leaf1 syslog.py[22830]: Message
Number:2
2017-02-22T19:59:50.044565+00:00 leaf1 syslog.py[22830]: Message
Number:3
2017-02-22T19:59:50.044830+00:00 leaf1 syslog.py[22830]: Message
Number:4
2017-02-22T19:59:50.045680+00:00 leaf1 syslog.py[22830]: Message
Number:5
<...snip...
2017-02-22T19:59:50.056727+00:00 leaf1 syslog.py[22830]: Message
Number:45
2017-02-22T19:59:50.057599+00:00 leaf1 syslog.py[22830]: Message
Number:46
2017-02-22T19:59:50.057741+00:00 leaf1 syslog.py[22830]: Message
Number:47
2017-02-22T19:59:50.057936+00:00 leaf1 syslog.py[22830]: Message
Number:48
2017-02-22T19:59:50.058125+00:00 leaf1 syslog.py[22830]: Message
Number:49
2017-02-22T19:59:50.058324+00:00 leaf1 rsyslogd-2177: imuxsock[pid
22830]: begin to drop messages due to rate-limiting
```

Harmless syslog Error: Failed to reset devices.list

The following message gets logged to `/var/log/syslog` when you run `systemctl daemon-reload` and during system boot:

```
systemd[1]: Failed to reset devices.list on /system.slice: Invalid
argument
```

This message is harmless, and can be ignored. It is logged when `systemd` attempts to change cgroup attributes that are read only. The upstream version of `systemd` has been modified to not log this message by default.

The `systemctl daemon-reload` command is often issued when Debian packages are installed, so the message may be seen multiple times when upgrading packages.

Next Steps

The links below discuss more specific monitoring topics.



Single User Mode - Boot Recovery

Use single user mode to assist in troubleshooting system boot issues or for password recovery. To enter single user mode, follow steps below.

1. Boot the switch, as soon as you see the GRUB menu.

```
GNU GRUB  version 2.02~beta2-22+deb8u1

+-----+
-----+
| *Cumulus Linux GNU
/Linux
| Advanced options for Cumulus Linux GNU
/Linux
|
ONIE
|
|
+-----+
-----+
```

2. Use the ^ and v arrow keys to select **Advanced options for Cumulus Linux GNU/Linux**. A menu similar to the following should appear:

```
GNU GRUB  version 2.02~beta2-22+deb8u1

+-----+
-----+
| Cumulus Linux GNU/Linux, with Linux 4.1.0-cl-1-
amd64
| Cumulus Linux GNU/Linux, with Linux 4.1.0-cl-1-amd64
(sysvinit)
| *Cumulus Linux GNU/Linux, with Linux 4.1.0-cl-1-amd64
(recovery mode)
|
|
+-----+
-----+
```

3. Select **Cumulus Linux GNU/Linux, with Linux 4.1.0-cl-1-amd64 (recovery mode)**.

4. Press **ctrl-x** to reboot.
5. After the system reboots, set a new password.

```
cumulus@switch:~$ sudo passwd
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
```

6. Sync the `/etc` directory using `btrfs`, then reboot the system:

```
cumulus@switch:~$ sudo btrfs filesystem sync /etc
cumulus@switch:~$ sudo reboot -f
Restarting the system.
```

Resource Diagnostics Using `cl-resource-query`

You can use `cl-resource-query` to retrieve information about host entries, MAC entries, L2 and L3 routes, and ECMPs (equal-cost multi-path routes, see [Load Balancing \(see page 488\)](#)) that are in use. This is especially useful because Cumulus Linux syncs routes between the kernel and the switching silicon. If the required resource pools in hardware fill up, new kernel routes can cause existing routes to move from being fully allocated to being partially allocated.

In order to avoid this, routes in the hardware should be monitored and kept below the ASIC limits. For example, on systems with a Broadcom Trident II chipset, the limits are as follows:

```
routes: 8092 <<< if all routes are IPv6, or 16384 if all routes are
IPv4
long mask routes 2048 <<< these are routes with a mask longer than
the route mask limit
route mask limit 64
host_routes: 8192
ecmp_nhs: 16346
ecmp_nhs_per_route: 52
```

This translates to about 314 routes with ECMP next hops, if every route has the maximum ECMP NHs.

You can monitor this in Cumulus Linux with the `cl-resource-query` command. Results vary between switches running on different chipsets.

`cl-resource-query` results for a Mellanox Spectrum switch:

```
cumulus@switch:~$ sudo cl-resource-query
Host entries: 2, 0% of maximum value 5120
IPv4 neighbors: 2
IPv6 neighbors: 0
IPv4 entries: 33, 0% of maximum value 39936
```

```

IPv6 entries:          13,  0% of maximum value 15360
IPv4 Routes:           33
IPv6 Routes:           13
Total Routes:          46,  0% of maximum value 32768
ECMP nexthops:         0,   0% of maximum value 209664
MAC entries:           25,  0% of maximum value 409600

```

cl-resource-query results for a Broadcom Tomahawk switch:

```

cumulus@switch:~$ sudo cl-resource-query
Host entries:          1,   0% of maximum value 20480 <<< 2
IPv4 neighbors can use one entry
IPv4 neighbors:         1
IPv6 neighbors:         0
IPv4 entries:           5,   0% of maximum value 32668 <<< switch
h overrides the SDK max limits
IPv6 entries:           4,   0% of maximum value 16384 <<<
IPv4 Routes:            5
IPv6 Routes:            4
Total Routes:           9,   0% of maximum value 32768
ECMP nexthops:          0,   0% of maximum value 16350
MAC entries:            2,   0% of maximum value 40960

```

cl-resource-query results for a Broadcom Trident II switch:

```

cumulus@switch:~$ sudo cl-resource-query
Host entries:          1,   0% of maximum value 8192 <<< this
is the default software-imposed limit, 50% of the hardware limit
IPv4 neighbors:         1           <<< these are counts of the
number of valid entries in the table
IPv6 neighbors:         0
IPv4 entries:           13,  0% of maximum value 32668
IPv6 entries:           18,  0% of maximum value 16384
IPv4 Routes:            13
IPv6 Routes:            18
Total Routes:           31,  0% of maximum value 32768
ECMP nexthops:          0,   0% of maximum value 16346
MAC entries:            12,  0% of maximum value 32768

```

Monitoring System Hardware

You monitor system hardware in these ways, using:

- decode-syseeprom
- sensors
- smond



- Net-SNMP (see page 696)
- watchdog

Contents

This chapter covers ...

- Monitoring Hardware Using decode-syseeprom (see page 635)
 - Command Options (see page 636)
 - Related Commands (see page 636)
- Monitoring Hardware Using sensors (see page 636)
 - Command Options (see page 637)
- Monitoring Switch Hardware Using SNMP (see page 638)
- Monitoring System Units Using smond (see page 638)
 - Command Options (see page 638)
- Keeping the Switch Alive Using the Hardware Watchdog (see page 639)
- Related Information (see page 639)

Monitoring Hardware Using decode-syseeprom

The `decode-syseeprom` command enables you to retrieve information about the switch's EEPROM. If the EEPROM is writable, you can set values on the EEPROM.

For example:

```
cumulus@switch:~$ decode-syseeprom
TlvInfo Header:
  Id String:      TlvInfo
  Version:        1
  Total Length:   114
TLV Name           Code Len Value
-----
Product Name       0x21  4 4804
Part Number        0x22  14 R0596-F0009-00
Device Version     0x26  1 2
Serial Number      0x23  19 D1012023918PE000012
Manufacture Date   0x25  19 10/09/2013 20:39:02
Base MAC Address   0x24  6 00:E0:EC:25:7B:D0
MAC Addresses      0x2A  2 53
Vendor Name        0x2D  17 Penguin Computing
Label Revision     0x27  4 4804
Manufacture Country 0x2C  2 CN
CRC-32             0xFE  4 0x96543BC5
  (checksum valid)
```



Command Options

Usage: /usr/cumulus/bin/decode-syseeprom [-a][-r][-s [args]][-t]

Option	Description
-h, --help	Displays the help message and exits.
-a	Prints the base MAC address for switch interfaces.
-r	Prints the number of MACs allocated for switch interfaces.
-s	Sets the EEPROM content if the EEPROM is writable. <code>args</code> can be supplied in command line in a comma separated list of the form ' <code><field>=<value></code> , ...'. '.', '=' are illegal characters in field names and values. Fields that are not specified will default to their current values. If <code>args</code> are supplied in the command line, they will be written without confirmation. If <code>args</code> is empty, the values will be prompted interactively.
-t TARGET	Selects the target EEPROM (<code>board</code> , <code>psu2</code> , <code>psu1</code>) for the read or write operation; default is <code>board</code> .
-e, --serial	Prints the device serial number.

Related Commands

You can also use the `dmidecode` command to retrieve hardware configuration information that's been populated in the BIOS.

You can use `apt-get` to install the `lshw` program on the switch, which also retrieves hardware configuration information.

Monitoring Hardware Using sensors

The `sensors` command provides a method for monitoring the health of your switch hardware, such as power, temperature and fan speeds. This command executes [lm-sensors](#).

For example:

```
cumulus@switch:~$ sensors
tmp75-i2c-6-48
Adapter: i2c-1-mux (chan_id 0)
temp1:          +39.0  C  (high = +75.0  C, hyst = +25.0  C)

tmp75-i2c-6-49
Adapter: i2c-1-mux (chan_id 0)
```

```

temp1:          +35.5 C  (high = +75.0 C, hyst = +25.0 C)

ltc4215-i2c-7-40
Adapter: i2c-1-mux (chan_id 1)
in1:          +11.87 V
in2:          +11.98 V
power1:        12.98 W
curr1:         +1.09 A

max6651-i2c-8-48
Adapter: i2c-1-mux (chan_id 2)
fan1:          13320 RPM (div = 1)
fan2:          13560 RPM

```



Output from the `sensors` command varies depending upon the switch hardware you use, as each platform ships with a different type and number of sensors.

Command Options

Usage: `sensors [OPTION]... [CHIP]...`

Option	Description
<code>-c, --config-file</code>	Specify a config file; use <code>-</code> after <code>-c</code> to read the config file from <code>stdin</code> ; by default, <code>sensors</code> references the configuration file in <code>/etc/sensors.d/</code> .
<code>-s, --set</code>	Executes set statements in the config file (root only); <code>sensors -s</code> is run once at boot time and applies all the settings to the boot drivers.
<code>-f, --fahrenheit</code>	Show temperatures in degrees Fahrenheit.
<code>-A, --no-adapter</code>	Do not show the adapter for each chip.
<code>--bus-list</code>	Generate bus statements for <code>sensors.conf</code> .

If `[CHIP]` is not specified in the command, all chip info will be printed. Example chip names include:

- `lm78-i2c-0-2d *-i2c-0-2d`
- `lm78-i2c-0-* *-i2c-0-*`
- `lm78-i2c-*_2d *-i2c-*_2d`
- `lm78-i2c-*-* *-i2c-*-*`
- `lm78-isa-0290 *-isa-0290`
- `lm78-isa-* *-isa-*`



- lm78-*

Monitoring Switch Hardware Using SNMP

The Net-SNMP documentation is discussed here (see page 696).

Monitoring System Units Using smond

The `smond` daemon monitors system units like power supply and fan, updates their corresponding LEDs, and logs the change in the state. Changes in system unit state are detected via the `cp1d` registers. `smond` utilizes these registers to read all sources, which impacts the health of the system unit, determines the unit's health, and updates the system LEDs.

Use `smonctl` to display sensor information for the various system units:

```
cumulus@switch:~$ smonctl
Board : OK
Fan   : OK
PSU1  : OK
PSU2  : BAD
Temp1  (Networking ASIC Die Temp Sensor ) : OK
Temp10 (Right side of the board      ) : OK
Temp2  (Near the CPU (Right)        ) : OK
Temp3  (Top right corner          ) : OK
Temp4  (Right side of Networking ASIC ) : OK
Temp5  (Middle of the board         ) : OK
Temp6  (P2020 CPU die sensor       ) : OK
Temp7  (Left side of the board      ) : OK
Temp8  (Left side of the board      ) : OK
Temp9  (Right side of the board     ) : OK
```

Command Options

Usage: `smonctl [OPTION]... [CHIP]...`

Option	Description
<code>-s SENSOR, --sensor SENSOR</code>	Displays data for the specified sensor.
<code>-v, --verbose</code>	Displays detailed hardware sensors data.

For more information, read `man smond` and `man smonctl`.



Keeping the Switch Alive Using the Hardware Watchdog

Cumulus Linux includes a simplified version of the `wd_keepalive(8)` daemon from the standard `watchdog` Debian package. `wd_keepalive` writes to a file called `/dev/watchdog` periodically to keep the switch from resetting, at least once per minute. Each write delays the reboot time by another minute. After one minute of inactivity where `wd_keepalive` doesn't write to `/dev/watchdog`, the switch resets itself. The watchdog is enabled by default on all supported switches, and starts when you boot the switch, before `switchd` starts.

To enable the hardware watchdog, edit the `/etc/watchdog.d/<your_platform>` file and set `run_watchdog` to 1:

```
run_watchdog=1
```

To disable the watchdog, edit the `/etc/watchdog.d/<your_platform>` file and set `run_watchdog` to 0 :

```
run_watchdog=0
```

Then stop the daemon:

```
cumulus@switch:~$ sudo systemctl stop wd_keepalive.service
```

You can modify the settings for the watchdog — like the timeout setting and scheduler priority — in its configuration file, `/etc/watchdog.conf`.

Related Information

- packages.debian.org/search?keywords=lshw
- lm-sensors.org
- [Net-SNMP tutorials](#)

Monitoring Virtual Device Counters

Cumulus Linux gathers statistics for VXLANS and VLANs using virtual device counters. These counters are supported on Tomahawk, Trident II+ and Trident II-based platforms only; see the [Cumulus Networks HCL](#) for a list of supported platforms.

You can retrieve the data from these counters using tools like `ip -s link show`, `ifconfig`, `/proc/net/dev`, or `netstat -i`.

Contents

This chapter covers ...

- Sample VXLAN Statistics (see page 640)



- Sample VLAN Statistics (see page 641)
 - For VLANs Using the non-VLAN-aware Bridge Driver (see page 641)
 - For VLANs Using the VLAN-aware Bridge Driver (see page 641)
- Configuring the Counters in switchd (see page 642)
 - Configuring the Poll Interval (see page 642)
 - Configuring Internal VLAN Statistics (see page 642)
 - Clearing Statistics (see page 643)
- Caveats and Errata (see page 643)

Sample VXLAN Statistics

VXLAN statistics are available as follows:

- Aggregate statistics are available per VNI; this includes access and network statistics.
- Network statistics are available for each VNI and displayed against the VXLAN device. This is independent of the VTEP used, so this is a summary of the VNI statistics across all tunnels.
- Access statistics are available per VLAN subinterface.

First, get interface information regarding the VXLAN bridge:

```
cumulus@switch:~$ brctl show br-vxln16757104
bridge name          bridge id      STP enabled    interfaces
-vxln16757104        8000.443839006988    no           swp2s0.6
                                         swp2s1.6
                                         swp2s2.6
                                         swp2s3.6
                                         vxln16757104
```

To get VNI statistics, run:

```
cumulus@switch:~$ ip -s link show br-vxln16757104
62: br-vxln16757104: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP mode DEFAULT
    link/ether 44:38:39:00:69:88 brd ff:ff:ff:ff:ff:ff
    RX: bytes   packets   errors   dropped overrun mcast
        10848       158       0        0        0        0
    TX: bytes   packets   errors   dropped carrier collsns
        27816       541       0        0        0        0
```

To get access statistics, run:

```
cumulus@switch:~$ ip -s link show swp2s0.6
63: swp2s0.6@swp2s0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue master br-vxln16757104 state UP mode DEFAULT
    link/ether 44:38:39:00:69:88 brd ff:ff:ff:ff:ff:ff
    RX: bytes   packets   errors   dropped overrun mcast
```



2680	39	0	0	0	0
TX: bytes	packets	errors	dropped	carrier	collsns
7558	140	0	0	0	0

To get network statistics, run:

```
cumulus@switch:~$ ip -s link show vxln16757104
61: vxln16757104: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue
    master br-vxln16757104 state UNKNOWN mode DEFAULT
        link/ether e2:37:47:db:f1:94 brd ff:ff:ff:ff:ff:ff
        RX: bytes packets errors dropped overrun mcast
            0         0       0       0       0       0
        TX: bytes packets errors dropped carrier collsns
            0         0       0       9       0       0
```

Sample VLAN Statistics

For VLANs Using the non-VLAN-aware Bridge Driver

In this case, each bridge is a single L2 broadcast domain and is associated with an internal VLAN. This internal VLAN's counters are displayed as bridge netdev stats.

```
cumulus@switch:~$ brctl show br0
bridge name     bridge id          STP enabled    interfaces
br0             8000.443839006989   yes           bond0.100
                                         swp2s2.100
cumulus@switch:~$ ip -s link show br0
42: br0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue
    state UP mode DEFAULT
        link/ether 44:38:39:00:69:89 brd ff:ff:ff:ff:ff:ff
        RX: bytes packets errors dropped overrun mcast
            23201498  227514    0       0       0       0
        TX: bytes packets errors dropped carrier collsns
            18198262  178443    0       0       0       0
```

For VLANs Using the VLAN-aware Bridge Driver

For a bridge using the VLAN-aware driver (see page 277), the bridge is a just a container and each VLAN (VID /PVID) in the bridge is an independent L2 broadcast domain. As there is no netdev available to display these VLAN statistics, the `switchd` nodes are used instead:

```
cumulus@switch:~$ ifquery bridge
auto bridge
iface bridge inet static
    bridge-vlan-aware yes
```

```

bridge-ports swp2s0 swp2s1
bridge-stp on
bridge-vids 2000-2002 4094
cumulus@switch:~$ ls /cumulus/switchd/run/stats/vlan/
2 2000 2001 2002 all
cumulus@switch:~$ cat /cumulus/switchd/run/stats/vlan/2000/aggregate
Vlan id                      : 2000
L3 Routed In Octets           : -
L3 Routed In Packets          : -
L3 Routed Out Octets          : -
L3 Routed Out Packets         : -
Total In Octets               : 375
Total In Packets              : 3
Total Out Octets              : 387
Total Out Packets             : 3

```

Configuring the Counters in switchd

These counters are enabled by default. To configure them, use `cl-cfg` and configure them as you would any other `switchd` parameter (see page 171). The `switchd` parameters are as follows:

- `stats.vlan.aggregate`, which controls the statistics available for each VLAN. Its value defaults to *BRIEF*.
- `stats.vxlan.aggregate`, which controls the statistics available for each VNI (access and network). Its value defaults to *DETAIL*.
- `stats.vxlan.member`, which controls the statistics available for each local/access port in a VXLAN bridge. Its value defaults to *BRIEF*.

The values for each parameter can be one of the following:

- NONE: This disables the counter.
- BRIEF: This provides tx/rx packet/byte counters for the associated parameter.
- DETAIL: This provides additional feature-specific counters. In the case of `stats.vxlan.aggregate`, DETAIL provides access vs. network statistics. For the other types, DETAIL has the same effect as BRIEF.



If you change one of these settings on the fly, the new configuration applies only to those VNIs or VLANs set up after the configuration changed; previously allocated counters remain as is.

Configuring the Poll Interval

The virtual device counters are polled periodically. This can be CPU intensive, so the interval is configurable in `switchd`, with a default of 2 seconds.

```

# Virtual devices hw-stat poll interval (in seconds)
#stats.vdev_hw_poll_interval = 2

```



Configuring Internal VLAN Statistics

For debugging purposes, you may need to access packet statistics associated with internal VLAN IDs. These statistics are hidden by default, but can be configured in `switchd`:

```
#stats.vlan.show_internal_vlans = FALSE
```

Clearing Statistics

Since `ethtool` is not supported for virtual devices, you cannot clear the statistics cache maintained by the kernel. You can clear the hardware statistics via `switchd`:

```
cumulus@switch:~$ sudo echo 1 > /cumulus/switchd/clear/stats/vlan
cumulus@switch:~$ sudo echo 1 > /cumulus/switchd/clear/stats/vxlan
cumulus@switch:~$
```

Caveats and Errata

- Currently the CPU port is internally added as a member of all VLANs. Because of this, packets sent to the CPU are counted against the corresponding VLAN's tx packets/bytes. There is no workaround.
- When checking the virtual counters for the bridge, the TX count is the number of packets destined to the CPU before any hardware policers take effect. For example, if 500 broadcast packets are sent into the bridge, the CPU is also sent 500 packets. These 500 packets are policed by the default ACLs in Cumulus Linux, so the CPU might receive fewer than the 500 packets if the incoming packet rate is too high. The TX counter for the bridge should be equal to $500 * (\text{number of ports in the bridge} - \text{incoming port} + \text{CPU port})$ or just $500 * \text{number of ports in the bridge}$.
- You cannot use `ethtool -s` for virtual devices. This is because the counters available via `netdev` are sufficient to display the vlan/vxlan counters currently supported in the hardware (only rx/tx packets/bytes are supported currently).

Understanding the `cl-support` Output File

The `cl-support` command generates an archive of useful information for troubleshooting that can be auto-generated or manually created. To manually create it, run the `cl-support` command. The `cl-support` file is automatically generated when:

- There is a **core file dump** of any application (not specific to Cumulus Linux, but something all Linux distributions support), located in `/var/support/core`
- After the first failure of one of the following monitored services since the switch was rebooted or power cycled:
 - clagd
 - openvswitch-vtep
 - portwd
 - ptmd
 - quagga



- rdnbrd
- switchd
- vxrd
- vxsnd

The Cumulus Networks support team may request you submit the output from `cl-support` to help with the investigation of issues you might experience with Cumulus Linux.

```
cumulus@switch:~$ sudo cl-support -h
Usage: cl-support [-h] [-s] [-t] [-v] [reason]...

Args:
[reason]: Optional reason to give for invoking cl-support.
          Saved into tarball's cmdline.args file.

Options:
-h: Print this usage statement
-s: Security sensitive collection
-t: User filename tag
-v: Verbose
-e MODULES: Enable modules. Comma separated module list (run with -e
help for module names)
-d MODULES: Disable modules. Comma separated module list (run with -d
help for module names)
```

You can find information on the directories included in the `cl-support` file:

- [Troubleshooting the etc Directory \(see page 647\)](#) — In terms of sheer numbers of files, `/etc` contains the largest number of files to send to Cumulus Networks by far. However, log files could be significantly larger in file size.
- [Troubleshooting Log Files \(see page 644\)](#) — This guide highlights the most important log files to look at. Keep in mind, `cl-support` includes all of the log files.

Troubleshooting Log Files

The only real unique entity for logging on Cumulus Linux compared to any other Linux distribution is `switchd.log`, which logs the HAL (hardware abstraction layer) from hardware like the Broadcom or Mellanox ASIC.

This guide on [NixCraft](#) is amazing for understanding how `/var/log` works. The green highlighted rows below are the most important logs and usually looked at first when debugging.

Log	Description	Why is this important?
<code>/var/log/alternatives.log</code>	Information from the update-alternatives are logged into this log file.	
<code>/var/log/apt</code>		



Log	Description	Why is this important?
	Information the <code>apt</code> utility can send logs here; for example, from <code>apt-get install</code> and <code>apt-get remove</code> .	
<code>/var/log/audit/*</code>	Contains log information stored by the Linux audit daemon, <code>auditd</code> .	
<code>/var/log/auth.log</code>	<p>Authentication logs. Note that Cumulus Linux does not write to this log file; but because it's a standard file, Cumulus Linux creates it as a zero length file.</p>	
<code>/var/log/autoprovision</code>	Logs output generated by running the zero touch provisioning (see page 68) script.	
<code>/var/log/boot.log</code>	Contains information that is logged when the system boots.	
<code>/var/log/btmp</code>	<p>This file contains information about failed login attempts. Use the <code>last</code> command to view the <code>btmp</code> file. For example:</p> <pre data-bbox="432 1072 1160 1142">cumulus@switch:~\$ last -f /var/log/btmp more</pre>	
<code>/var/log/clagd.log</code>	Logs status of the <code>clagd</code> service (see page 300).	
<code>/var/log/dmesg</code>	<p>Contains kernel ring buffer information. When the system boots up, it prints number of messages on the screen that display information about the hardware devices that the kernel detects during boot process. These messages are available in the kernel ring buffer and whenever a new message arrives, the old message gets overwritten. You can also view the content of this file using the <code>dmesg</code> command. Note that Cumulus Linux does not write to this log file; but because it's a standard file, Cumulus Linux creates it as a zero length file.</p>	
<code>/var/log/dpkg.log</code>	Contains information that is logged when a package is installed or removed using the <code>dpkg</code> command.	
<code>/var/log/faillog</code>	<p>Contains failed user login attempts. Use the <code>faillog</code> command to display the contents of this file. Note that Cumulus Linux does not write to this log file; but because it's a standard file, Cumulus Linux creates it as a zero length file.</p>	



Log	Description	Why is this important?
/var/log/fsck/*	<p>The <code>fsck</code> utility is used to check and optionally repair one or more Linux filesystems.</p> <p>Note that Cumulus Linux does not write to this log file; but because it's a standard file, Cumulus Linux creates it as a zero length file.</p>	
/var/log/installer/*	Directory containing files related to the installation of Cumulus Linux.	
/var/log/lastlog	Formats and prints the contents of the last login log file.	
/var/log/netd.log	Log file for NCLU (see page 80).	
/var/log/news/*	<p>The <code>news</code> command keeps you informed of news concerning the system.</p> <p>Note that Cumulus Linux does not write to this log file; but because it's a standard file, Cumulus Linux creates it as a zero length file.</p>	
/var/log/ntpstats	Logs for network configuration protocol.	
/var/log/openvswitch/*	ovsdb-server logs.	
/var/log/quagga/*	Where Quagga logs to once enabled.	This is how Cumulus Networks troubleshoots routing. For example an md5 or mtu mismatch with OSPF.
/var/log/rdnbrd.log	Logs for redistribute neighbor (see page 564).	
/var/log/snapper.log	Log file for snapshots (see page 55).	These logs are valuable for the snapshots you take on your switch.
	The HAL log for Cumulus Linux.	



Log	Description	Why is this important?
/var/log/switchd.log		This is specific to Cumulus Linux. Any switchd crashes are logged here.
/var/log/syslog	The main system log, which logs everything except auth-related messages.	The primary log; it's easiest to grep this file to see what occurred during a problem.
/var/log/wtmp	Login records file.	

Troubleshooting the etc Directory

The `c1-support` (see page 643) script replicates the `/etc` directory.

Files that `c1-support` deliberately excludes are:

File	Description
<code>/etc/nologin</code>	<code>nologin</code> prevents unprivileged users from logging into the system.
<code>/etc/alternatives</code>	<code>update-alternatives</code> creates, removes, maintains and displays information about the symbolic links comprising the Debian alternatives system.

This is the alphabetical of the output from running `ls -l` on the `/etc` directory structure created by `c1-support`. The green highlighted rows are the ones Cumulus Networks finds most important when troubleshooting problems.

File	Description	Why is this important?
<code>adduser.conf</code>	The file <code>/etc/adduser.conf</code> contains defaults for the programs <code>adduser</code> , <code>addgroup</code> , <code>deluser</code> , and <code>delgroup</code> .	
<code>adjtime</code>	Corrects the time to synchronize the <code>system clock</code> .	
<code>apt</code>	<code>apt</code> (Advanced Package Tool) is the command-line tool for handling packages . This folder contains all the configurations.	<code>apt</code> interactions or unsupported apps can affect machine performance.

audisp	The directory that contains <code>audisp-remote.conf</code> , which is the file that controls the configuration of the audit remote logging subsystem .	
audit	The directory that contains the <code>/etc/audit/auditd.conf</code> , which contains configuration information specific to the audit daemon .	
bash.bashrc	Bash is an sh-compatible command language interpreter that executes commands read from standard input or from a file.	
bash_completion	This points to <code>/usr/share/bash-completion/bash_completion</code> .	
bash_completion.d	This folder contains app-specific code for Bash completion on Cumulus Linux, such as <code>mstptcl</code> .	
bcm.d	Broadcom-specific ASIC file structure (hardware interaction). If there are questions, contact the Cumulus Networks Support team . This is unique to Cumulus Linux.	
mlx	Mellanox-specific ASIC file structure (hardware interaction). If there are questions, contact the Cumulus Networks Support team . This is unique to Cumulus Linux.	
bindresvport.blacklist	This file contains a list of port numbers between 600 and 1024, which should not be used by <code>bindresvport</code> .	
ca-certificates	The folder for <code>ca-certificates</code> . It is empty by default on Cumulus Linux; see below for more information.	
ca-certificates.conf	Each lines list the pathname of activated CA certificates under <code>/usr/share/ca-certificates</code> .	
calendar	The system-wide default calendar file .	
chef	This is an example of something that is not included by default. In this instance, <code>c1-support</code> included the <code>chef</code> folder for some reason.	

File	Description	Why is this important?
		This is not installed by default, but this tool could have been installed or configured incorrectly, which is why it's included in the <code>cl-support</code> output.
<code>cron.d</code>	<code>cron</code> is a daemon that executes scheduled commands .	
<code>cron.daily</code>	See above.	
<code>cron.hourly</code>	See above.	
<code>cron.monthly</code>	See above.	
<code>cron.weekly</code>	See above.	
<code>crontab</code>	See above.	
<code>cumulus</code>	<p>This directory contains the following:</p> <ul style="list-style-type: none"> • ACL information, stored in the <code>acl</code> directory. • <code>switchd</code> configuration file, <code>switchd.conf</code>. • <code>qos</code>, which is under the <code>datapath</code> directory. • The routing protocol process priority, <code>nice.conf</code>. • The breakout cable configuration, under <code>ports.conf</code>. 	This folder is specific to Cumulus Linux and does not exist on other Linux platforms. For example, while you can configure <code>iptables</code> , to hardware accelerate rules into the hardware you need to use <code>cl-acltool</code> and have the rules under the <code>/etc/cumulus/acl/policy.d/<filename.rules</code>)
<code>debconf.conf</code>	Debconf is a configuration system for Debian packages .	
<code>debian_version</code>	The complete Debian version string .	
<code>debsums-ignore</code>	<code>debsums</code> verifies installed package files against their MD5 checksums. This file identifies the packages to ignore.	
<code>default</code>	This folder contains files with configurable flags for many different applications (most installed by default or added manually). For example, <code>/etc/default/networking</code> has a flag for <code>EXCLUDE_INTERFACES=</code> , which is set to nothing by default, but a user could change it to something like <code>swp3</code> .	

deluser.conf	The file <code>/etc/deluser.conf</code> contains defaults for the programs <code>deluser</code> and <code>delgroup</code> .	
dhcp	This directory contains DHCP-specific information .	
dpkg	The package manager for Debian.	
e2fsck.conf	The configuration file for e2fsck . It controls the default behavior of <code>e2fsck</code> while it checks ext2, ext3 or ext4 filesystems.	
environment	Utilized by pam_env for setting and unsetting environment variables.	
ethertypes	This file can be used to show readable characters instead of hexadecimal numbers for the protocols. For example, 0x0800 will be represented by IPv4.	
fstab	Static information about the filesystems .	
fstab.d	The directory that can contain additional <code>fstab</code> information; it is empty by default.	
fw_env.config	Configuration file utilized by U-Boot .	
gai.conf	Configuration file for sorting the return information from getaddrinfo .	
groff	The directory containing information for <code>groffer</code> , an application used for displaying Unix man pages .	
group	The <code>/etc/group</code> file is a text file that defines the groups on the system.	
group-	Backup for the <code>/etc/group</code> file.	
gshadow	<code>/etc/gshadow</code> contains the shadowed information for group accounts .	
gshadow-	Backup for the <code>/etc/gshadow</code> file.	
host.conf	Resolver configuration file , which contains options like <code>multi</code> that determines whether <code>/etc/hosts</code> will respond with multiple entries for DNS names.	



File	Description	Why is this important?
hostname	The system host name , such as leaf1, spine1, sw1.	
hosts	The static table lookup for hostnames.	
hosts.allow	The part of the host_access program for controlling a simple access control language. <code>hosts.allow=Access</code> is granted when a daemon/client pair matches an entry.	
hosts.deny	See hosts.allow above, except that access is denied when a daemon/client pair matches an entry.	
init	Default location of the system job configuration files .	
init.d	In order for a service to start when the switch boots, you should add the necessary script to the director here. The differences between <code>init</code> and <code>init.d</code> are explained well here .	
inittab	The format of the inittab file used by the sysv-compatible <code>init</code> process.	
inputrc	The initialization file utilized by <code>readline</code> .	
insserv	This application enables installed system init scripts ; this directory is empty by default.	
insserv.conf	Configuration file for insserv .	
insserv.conf.d	Additional directory for insserv configurations .	
iproute2	Directory containing values for the Linux command line tool <code>ip</code> .	
issue	<code>/etc/issue</code> is a text file that contains a message or system identification to be printed before the login prompt.	
issue.net	Identification file for telnet sessions .	
ld.so.cache	Contains a compiled list of candidate libraries previously found in the augmented library path.	
ld.so.conf	Used by the <code>ldconfig</code> tool, which configures dynamic linker run-time bindings .	

ld.so.conf.d	The directory that contains additional <code>ld.so.conf</code> configuration (see above).	
ldap	The directory containing the ldap.conf configuration file used to set the system-wide default to be applied when running LDAP clients.	
libaudit.conf	Configuration file utilized by get_auditfail_action .	
libnl-3	Directory for the configuration relating to the libnl library , which is the core library for implementing the fundamentals required to use the netlink protocol such as socket handling, message construction and parsing, and sending and receiving of data.	
lldpd.d	Directory containing configuration files whose commands are executed by <code>lldpccli</code> at startup.	
localtime	Copy of the original data file for <code>/etc/timezone</code> .	
logcheck	Directory containing <code>logcheck.conf</code> and logfiles utilized by the <code>log_check</code> program, which scans system logs for interesting lines.	
login.defs	Shadow password suite configuration .	
logrotate.conf	Rotates, compresses and mails system logs .	
logrotate.d	Directory containing additional log rotate configurations.	
lsb-release	Shows the current version of Linux on the system. Run <code>cat /etc/lsb-release</code> for output.	This shows you the version of the operating system you are running; also compare this to the output of <code>onie-select</code> .
magic	Used by the file command to determine file type. <code>magic</code> tests check for files with data in particular fixed formats.	
magic.mime	The magic MIME type causes the <code>file</code> command to output MIME type strings rather than the more traditional human readable ones.	



File	Description	Why is this important?
mailcap	The <code>mailcap</code> file is read by the metamail program to determine how to display non-text at the local site.	
mailcap.order	The order of entries in the <code>/etc/mailcap</code> file can be altered by editing the <code>/etc/mailcap.order</code> file.	
manpath.config	The manpath configuration file is used by the manual page utilities to assess users' manpaths at run time, to indicate which manual page hierarchies (manpaths) are to be treated as system hierarchies and to assign them directories to be used for storing cat files.	
mime.types	MIME type description file for cups .	
mke2fs.conf	Configuration file for <code>mke2fs</code> , which is a program that creates an ext, ext3 or ext4 filesystem .	
mlx	Mellanox-specific ASIC file structure (hardware interaction). If there are questions, contact the Cumulus Networks Support team . This is unique to Cumulus Linux.	
modprobe.d	Configuration directory for <code>modprobe</code> , which is a utility that can add and remove modules from the Linux kernel .	
modules	The kernel modules to load at boot time .	
motd	The contents of <code>/etc/motd</code> ("message of the day") are displayed by <code>pam_motd</code> after a successful login but just before it executes the login shell.	
mtab	The programs <code>mount</code> and <code>umount</code> maintain a list of currently mounted filesystems in the <code>/etc/mtab</code> file. If no arguments are given to <code>mount</code> , this list is printed.	
nanorc	The GNU nano <code>rcfile</code> .	
network	Contains the network interface configuration for <code>ifup</code> and <code>ifdown</code> .	The main configuration file is under <code>/etc/network/interfaces</code> . This is where you configure L2 and L3 information for all of your front panel ports (<code>swp</code>

File	Description	Why is this important?
		interfaces). Settings like MTU, link speed, IP address information, VLANs are all done here.
networks	Network name information.	
nsswitch.conf	System databases and name service switch configuration file.	
ntp.conf	NTP (network time protocol) server configuration file.	
openvswitch	The directory containing the <code>conf.db</code> file, which is used by <code>ovsdb-server</code> .	
openvswitch-vtep	Configuration files used for the VTEP daemon and <code>ovsdb-server</code> .	
opt	Host-specific configuration files for <code>add-on applications</code> installed in <code>/opt</code> .	
os-release	Operating system identification.	
pam.conf	The PAM (pluggable authentication module) configuration file. When a PAM-aware privilege granting application is started, it activates its attachment to the PAM-API. This activation performs a number of tasks, the most important being the reading of the configuration file(s).	
pam.d	Alternate directory to configure PAM (see above).	
passwd	User account information.	
passwd-	Backup file for <code>/etc/passwd</code> .	
perl	<code>Perl</code> is an available scripting language. <code>/etc/perl</code> contains configuration files specific to Perl.	
profile	<code>/etc/profile</code> is utilized by <code>sysprofile</code> , a modular centralized shell configuration.	
profile.d	The directory version of the above, which contains configuration files.	



File	Description	Why is this important?
protocols	The protocols definition file , a plain ASCII file that describes the various DARPA net protocols that are available from the TCP/IP subsystem.	
ptm.d	The directory containing scripts that are run if PTM (see page 257) passes or fails.	Cumulus Linux-specific folder for PTM (prescriptive topology manager).
python	Python is an available scripting language.	
python2.6	The 2.6 version of python .	
python2.7	The 2.7 version of python .	
quagga	Contains the configuration files for the Quagga routing suite (see page 489), the preferred Cumulus Linux routing engine.	
rc.local	The <code>/etc/rc.local</code> script is used by the system administrator to execute after all the normal system services are started , at the end of the process of switching to a multiuser runlevel. You can use it to start a custom service, for example, a server that's installed in <code>/usr/local</code> . Most installations don't need <code>/etc/rc.local</code> ; it's provided for the minority of cases where it's needed .	
rc0.d	Like <code>rc.local</code> , these scripts are booted by default, but the number of the folder represents the Linux runlevel . This folder 0 represents runlevel 0 (halt the system).	
rc1.d	This is run level 1, which is single-user/minimal mode.	
rc2.d	Runlevels 2 through 5 are multiuser modes. Debian systems (such as Cumulus Linux) come with <code>id=2</code> , which indicates that the default runlevel will be 2 when the multi-user state is entered , and the scripts in <code>/etc/rc2.d/</code> will be run.	
rc3.d	See above.	
rc4.d	See above.	
rc5.d	See above.	

File	Description	Why is this important?
rc6.d	Runlevel 6 is reboot the system.	
rcS.d	S stands for <i>single</i> and is equivalent to rc1.	
resolv.conf	Resolver configuration file , which is where DNS is set (domain, nameserver and search).	You need DNS to reach the Cumulus Linux repository.
rmt	This is not a mistake. The shell script /etc/rmt is provided for compatibility with other Unix-like systems, some of which have utilities that expect to find (and execute) rmt in the /etc directory on remote systems.	
rpc	The rpc file contains human-readable names that can be used in place of RPC program numbers.	
rsyslog.conf	The rsyslog.conf file is the main configuration file for rsyslogd , which logs system messages on *nix systems.	
rsyslog.d	The directory containing additional configuration for rsyslog.conf (see above).	
securetty	This file lists terminals into which the root user can log in .	
security	The /etc/security directory contains security-related configurations files . Whereas PAM concerns itself with the methods used to authenticate any given user, the files under /etc/security are concerned with just what a user can or cannot do. For example, the /etc/security/access.conf file contains a list of which users are allowed to log in and from what host (for example, using telnet). The /etc/security/limits.conf file contains various system limits, such as maximum number of processes.	
selinux	NSA Security-Enhanced Linux .	
sensors.d	The directory from which the sensors program loads its configuration; this is unique for each hardware platform. See also Monitoring System Hardware (see page 634).	
sensors3.conf		



File	Description	Why is this important?
	The sensors.conf file describes how <code>libsensors</code> , and thus all programs using it, should translate the raw readings from the kernel modules to real-world values.	
services	<code>services</code> is a plain ASCII file providing a mapping between human-readable textual names for internet services and their underlying assigned port numbers and protocol types.	
shadow	<code>shadow</code> is a file that contains the password information for the system's accounts and optional aging information.	
shadow-	The backup for the <code>/etc/shadow</code> file.	
shells	The pathnames of valid login shells .	
skel	The skeleton directory (usually <code>/etc/skel</code>) is used to copy default files and also sets a umask for the creation used by <code>pam_mkhomedir</code> .	
snmp	Interface functions to the SNMP (simple network management protocol) toolkit.	
ssh	The ssh configuration .	
ssl	The OpenSSL ss1 library implements the Secure Sockets Layer (SSL v2/v3) and Transport Layer Security (TLS v1) protocols. This directory holds certificates and configuration.	
staff-group-for-usr-local	Use <code>cat</code> or <code>more</code> on this file to learn more information, see http://bugs.debian.org/299007 .	
sudoers	The <code>sudoers</code> policy plugin determines a user's sudo privileges .	
sudoers.d	The directory file containing additional <code>sudoers</code> configuration (see above).	
sysctl.conf	Configures kernel parameters at boot .	
sysctl.d	The directory file containing additional configuration (see above).	



File	Description	Why is this important?
systemd	systemd system and service manager.	
terminfo	Terminal capability database.	
timezone	If this file exists, it is read and its contents are used as the time zone name.	
ucf.conf	The update configuration file preserves user changes in configuration files.	
udev	Dynamic device management.	
ufw	Provides both a command line interface and a framework for managing a netfilter firewall.	
vim	Configuration file for command line tool vim.	
wgetrc	Configuration file for command line tool wget.	

Troubleshooting Network Interfaces

The following sections describe various ways you can troubleshoot ifupdown2.

Contents

This chapter covers ...

- Enabling Logging for Networking (see page 658)
- Using ifquery to Validate and Debug Interface Configurations (see page 659)
- Debugging Mako Template Errors (see page 660)
- ifdown Cannot Find an Interface that Exists (see page 661)
- Removing All References to a Child Interface (see page 661)
- MTU Set on a Logical Interface Fails with Error: "Numerical result out of range" (see page 662)
- Interpreting iproute2 batch Command Failures (see page 662)
- Understanding the "RTNETLINK answers: Invalid argument" Error when Adding a Port to a Bridge (see page 663)
- MLAG Peerlink Interface Drops Many Packets (see page 663)

Enabling Logging for Networking

The /etc/default/networking file contains two settings for logging:

- To get ifupdown2 logs when the switch boots (stored in syslog)



- To enable logging when you run `systemctl [start|stop|reload] networking.service`

This file also contains an option for excluding interfaces when you boot the switch or run `systemctl start|stop|reload networking.service`. You can exclude any interface specified in `/etc/network/interfaces`. These interfaces do not come up when you boot the switch or start/stop/reload the networking service.

```
cumulus@switch:~$ cat /etc/default/networking
#
#
# Parameters for the /etc/init.d/networking script
#
#
# Change the below to yes if you want verbose logging to be enabled
VERBOSE="no"

# Change the below to yes if you want debug logging to be enabled
DEBUG="no"

# Change the below to yes if you want logging to go to syslog
SYSLOG="no"

# Exclude interfaces
EXCLUDE_INTERFACES=
```

Using ifquery to Validate and Debug Interface Configurations

You use `ifquery` to print parsed `interfaces` file entries.

To use `ifquery` to pretty print `iface` entries from the `interfaces` file, run:

```
cumulus@switch:~$ sudo ifquery bond0
auto bond0
iface bond0
    address 14.0.0.9/30
    address 2001:ded:beef:2::1/64
    bond-slaves swp25 swp26
```

Use `ifquery --check` to check the current running state of an interface within the `interfaces` file. It will return exit code 0 or 1 if the configuration does not match. The line `bond-xmit-hash-policy layer3+7` below fails because it should read `bond-xmit-hash-policy layer3+4`.

```
cumulus@switch:~$ sudo ifquery --check bond0
iface bond0
    bond-xmit-hash-policy layer3+7  [fail]
    bond-slaves swp25 swp26        [pass]
    address 14.0.0.9/30           [pass]
```



```
address 2001:ded:beef:2::1/64 [pass]
```

 ifquery --check is an experimental feature.

Use ifquery --running to print the running state of interfaces in the `interfaces` file format:

```
cumulus@switch:~$ sudo ifquery --running bond0
auto bond0
iface bond0
    bond-slaves swp25 swp26
    address 14.0.0.9/30
    address 2001:ded:beef:2::1/64
```

ifquery --syntax-help provides help on all possible attributes supported in the `interfaces` file. For complete syntax on the `interfaces` file, see `man interfaces` and `man ifupdown-addons-interfaces`.

You can use ifquery --print-savedstate to check the `ifupdown2` state database. ifdown works only on interfaces present in this state database.

```
cumulus@leaf1$ sudo ifquery --print-savedstate eth0
auto eth0
iface eth0 inet dhcp
```

Debugging Mako Template Errors

An easy way to debug and get details about template errors is to use the `mako-render` command on your `interfaces` template file or on `/etc/network/interfaces` itself.

```
cumulus@switch:~$ sudo mako-render /etc/network/interfaces
# This file describes the network interfaces available on your system
# and how to activate them. For more information, see interfaces(5).

# The loopback network interface
auto lo
iface lo inet loopback

# The primary network interface
auto eth0
iface eth0 inet dhcp
#auto eth1
#iface eth1 inet dhcp

# Include any platform-specific interface configuration
source /etc/network/interfaces.d/*.if
```

```
# ssim2 added
auto swp45
iface swp45

auto swp46
iface swp46

cumulus@switch:~$ sudo mako-render /etc/network/interfaces.d
/<interfaces_stub_file>
```

ifdown Cannot Find an Interface that Exists

If you are trying to bring down an interface that you know exists, use `ifdown` with the `--use-current-config` option to force `ifdown` to check the current `/etc/network/interfaces` file to find the interface. This can solve issues where the `ifup` command issues for that interface was interrupted before it updated the state database. For example:

```
cumulus@switch:~$ sudo ifdown br0
error: cannot find interfaces: br0 (interface was probably never up ?)

cumulus@switch:~$ sudo brctl show
bridge name      bridge id      STP enabled     interfaces
br0              8000.44383900279f    yes          downlink
                           peerlink

cumulus@switch:~$ sudo ifdown br0 --use-current-config
```

Removing All References to a Child Interface

If you have a configuration with a child interface, whether it's a VLAN, bond or another physical interface, and you remove that interface from a running configuration, you must remove every reference to it in the configuration. Otherwise, the interface continues to be used by the parent interface.

For example, consider the following configuration:

```
auto lo
iface lo inet loopback

auto eth0
iface eth0 inet dhcp

auto bond1
iface bond1
    bond-slaves swp2 swp1

auto bond3
iface bond3
```



```
bond-slaves swp8 swp6 swp7

auto br0
iface br0
    bridge-ports swp3 swp5 bond1 swp4 bond3
    bridge-pathcosts swp3=4 swp5=4 swp4=4
    address 11.0.0.10/24
    address 2001::10/64
```

Notice that bond1 is a member of br0. If bond1 is removed, you must remove the reference to it from the br0 configuration. Otherwise, if you reload the configuration with `ifreload -a`, bond1 is still part of br0.

MTU Set on a Logical Interface Fails with Error: "Numerical result out of range"

This error occurs when the [MTU \(see page 210\)](#) you are trying to set on an interface is higher than the MTU of the lower interface or dependent interface. Linux expects the upper interface to have an MTU less than or equal to the MTU on the lower interface.

In the example below, the swp1.100 VLAN interface is an upper interface to physical interface swp1. If you want to change the MTU to 9000 on the VLAN interface, you must include the new MTU on the lower interface swp1 as well.

```
auto swp1.100
iface swp1.100
    mtu 9000

auto swp1
iface swp1
    mtu 9000
```

Interpreting iproute2 batch Command Failures

`ifupdown2` batches `iproute2` commands for performance reasons. A batch command contains `ip -force -batch -` in the error message. The command number that failed is at the end of this line: `Command failed -:1.`

Below is a sample error for the command `link set dev host2 master bridge`. There was an error adding the bond `host2` to the bridge named `bridge` because `host2` did not have a valid address.

```
error: failed to execute cmd 'ip -force -batch - [link set dev host2
master bridge
addr flush dev host2
link set dev host1 master bridge
addr flush dev host1
]' (RTNETLINK answers: Invalid argument
Command failed -:1)
warning: bridge configuration failed (missing ports)
```



Understanding the "RTNETLINK answers: Invalid argument" Error when Adding a Port to a Bridge

This error can occur when the bridge port does not have a valid hardware address.

This can typically occur when the interface being added to the bridge is an incomplete bond; a bond without slaves is incomplete and does not have a valid hardware address.

MLAG Peerlink Interface Drops Many Packets

Losing a large number of packets across an MLAG peerlink interface may not be a problem. Instead this could be occurring in order to prevent looping of BUM (broadcast, unknown unicast and multicast) packets. For more information, and how to detect these drops, read the [MLAG chapter \(see page 332\)](#).

Monitoring Interfaces and Transceivers Using ethtool

The `ethtool` command enables you to query or control the network driver and hardware settings. It takes the device name (like `swp1`) as an argument. When the device name is the only argument to `ethtool`, it prints the current settings of the network device. See `man ethtool(8)` for details. Not all options are currently supported on switch port interfaces.

Contents

This chapter covers ...

- Monitoring Interface Status Using ethtool (see page 663)
 - Viewing and Clearing Interface Counters (see page 664)
- Monitoring Switch Port SFP/QSFP Hardware Information Using ethtool (see page 665)

Monitoring Interface Status Using ethtool

To check the status of an interface using `ethtool`:

```
cumulus@switch:~$ ethtool swp1
Settings for swp1:
  Supported ports: [ FIBRE ]
  Supported link modes:  1000baseT/Full
                        10000baseT/Full
  Supported pause frame use: No
  Supports auto-negotiation: No
  Advertised link modes:  1000baseT/Full
  Advertised pause frame use: No
  Advertised auto-negotiation: No
  Speed: 10000Mb/s
  Duplex: Full
  Port: FIBRE
  PHYAD: 0
  Transceiver: external
  Auto-negotiation: off
  Current message level: 0x00000000 (0)
```



```
Link detected: yes
```

To query interface statistics:

```
cumulus@switch:~$ sudo ethtool -S swp1
NIC statistics:
    HwIfInOctets: 1435339
    HwIfInUcastPkts: 11795
    HwIfInBcastPkts: 3
    HwIfInMcastPkts: 4578
    HwIfOutOctets: 14866246
    HwIfOutUcastPkts: 11791
    HwIfOutMcastPkts: 136493
    HwIfOutBcastPkts: 0
    HwIfInDiscards: 0
    HwIfInL3Drops: 0
    HwIfInBufferDrops: 0
    HwIfInAclDrops: 28
    HwIfInDot3LengthErrors: 0
    HwIfInErrors: 0
    SoftInErrors: 0
    SoftInDrops: 0
    SoftInFrameErrors: 0
    HwIfOutDiscards: 0
    HwIfOutErrors: 0
    HwIfOutQDrops: 0
    HwIfOutNonQDrops: 0
    SoftOutErrors: 0
    SoftOutDrops: 0
    SoftOutTxFifoFull: 0
    HwIfOutQLen: 0
```

Viewing and Clearing Interface Counters

Interface counters contain information about an interface. You can view this information when you run `cl-netstat`, `ifconfig`, or `cat /proc/net/dev`. You can also use `cl-netstat` to save or clear this information:

```
cumulus@switch:~$ sudo cl-netstat
Kernel Interface table
Iface      MTU Met            RX_OK RX_ERR RX_DRP RX_OVR          TX_OK TX_ERR
TX_DRP TX_OVR   Flg
-----
eth0      1500 0              611     0       0       0           487       0
0          0   BMRU
```

```
lo      16436   0          0      0      0      0          0      0
0       0        LRU
swp1    1500    0          0      0      0      0          0      0
0       0        BMU
```

```
cumulus@switch:~$ sudo cl-netstat -c
Cleared counters
```

Option	Description
-c	Copies and clears statistics. It does not clear counters in the kernel or hardware.
-d	Deletes saved statistics, either the uid or the specified tag.
-D	Deletes all saved statistics.
-l	Lists saved tags.
-r	Displays raw statistics (unmodified output of cl-netstat).
-t <tag name>	Saves statistics with <tag name>.
-v	Prints cl-netstat version and exits.

Monitoring Switch Port SFP/QSFP Hardware Information Using ethtool

To see hardware capabilities and measurement information on the SFP or QSFP module installed in a particular port, use the `ethtool -m` command. If the SFP/QSFP supports Digital Optical Monitoring (that is, the `Optical diagnostics support` field in the output below is set to Yes), the optical power levels and thresholds are also printed below the standard hardware details.

In the sample output below, you can see that this module is a 1000BASE-SX short-range optical module, manufactured by JDSU, part number PLRXPL-VI-S24-22. The second half of the output displays the current readings of the Tx power levels (`Laser output power`) and Rx power (`Receiver signal average optical power`), temperature, voltage and alarm threshold settings.

```
cumulus@switch$ sudo ethtool -m swp3
Identifier                      : 0x03 (SFP)
Extended identifier              : 0x04 (GBIC/SFP
defined by 2-wire interface ID)
Connector                        : 0x07 (LC)
Transceiver codes                : 0x00 0x00 0x00 0x00
1 0x20 0x40 0x0c 0x05
Transceiver type                 : Ethernet:
1000BASE-SX
Transceiver type                 : FC: intermediate
distance (I)
```

	Transceiver type	: FC: Shortwave
laser w/o OFC (SN)	Transceiver type	: FC: Multimode, 62.
5um (M6)	Transceiver type	: FC: Multimode,
50um (M5)	Transceiver type	: FC: 200 MBytes/sec
	Transceiver type	: FC: 100 MBytes/sec
	Encoding	: 0x01 (8B/10B)
	BR, Nominal	: 2100MBd
	Rate identifier	: 0x00 (unspecified)
	Length (SMF,km)	: 0km
	Length (SMF)	: 0m
	Length (50um)	: 300m
	Length (62.5um)	: 150m
	Length (Copper)	: 0m
	Length (OM3)	: 0m
	Laser wavelength	: 850nm
	Vendor name	: JDSU
	Vendor OUI	: 00:01:9c
	Vendor PN	: PLRXPL-VI-S24-22
	Vendor rev	: 1
	Optical diagnostics support	: Yes
	Laser bias current	: 21.348 mA
dBm	Laser output power	: 0.3186 mW / -4.97
	Receiver signal average optical power	: 0.3195 mW / -4.96
dBm	Module temperature	: 41.70 degrees C /
107.05	degrees F	
	Module voltage	: 3.2947 V
	Alarm/warning flags implemented	: Yes
	Laser bias current high alarm	: Off
	Laser bias current low alarm	: Off
	Laser bias current high warning	: Off
	Laser bias current low warning	: Off
	Laser output power high alarm	: Off
	Laser output power low alarm	: Off
	Laser output power high warning	: Off
	Laser output power low warning	: Off
	Module temperature high alarm	: Off
	Module temperature low alarm	: Off
	Module temperature high warning	: Off
	Module temperature low warning	: Off
	Module voltage high alarm	: Off
	Module voltage low alarm	: Off
	Module voltage high warning	: Off
	Module voltage low warning	: Off
	Laser rx power high alarm	: Off
	Laser rx power low alarm	: Off
	Laser rx power high warning	: Off
	Laser rx power low warning	: Off



	Laser bias current high alarm threshold	:	10.000 mA
	Laser bias current low alarm threshold	:	1.000 mA
	Laser bias current high warning threshold	:	9.000 mA
	Laser bias current low warning threshold	:	2.000 mA
	Laser output power high alarm threshold	:	0.8000 mW / -0.97
dBm	Laser output power low alarm threshold	:	0.1000 mW / -10.00
dBm	Laser output power high warning threshold	:	0.6000 mW / -2.22
dBm	Laser output power low warning threshold	:	0.2000 mW / -6.99
dBm	Module temperature high alarm threshold	:	90.00 degrees C / 194.00 degrees F
	Module temperature low alarm threshold	:	-40.00 degrees C / -40.00 degrees F
	Module temperature high warning threshold	:	85.00 degrees C / 185.00 degrees F
	Module temperature low warning threshold	:	-40.00 degrees C / -40.00 degrees F
	Module voltage high alarm threshold	:	4.0000 V
	Module voltage low alarm threshold	:	0.0000 V
	Module voltage high warning threshold	:	3.6450 V
	Module voltage low warning threshold	:	2.9550 V
	Laser rx power high alarm threshold	:	1.6000 mW / 2.04
dBm	Laser rx power low alarm threshold	:	0.0100 mW / -20.00
dBm	Laser rx power high warning threshold	:	1.0000 mW / 0.00
dBm	Laser rx power low warning threshold	:	0.0200 mW / -16.99

Network Troubleshooting

Cumulus Linux contains a number of command line and analytical tools to help you troubleshoot issues with your network.

Contents

This chapter covers ...

- Checking Reachability Using ping (see page 668)
- Printing Route Trace Using traceroute (see page 668)
- Manipulating the System ARP Cache (see page 669)
- Generating Traffic Using mz (see page 669)
- Creating Counter ACL Rules (see page 670)
- Configuring SPAN and ERSPAN (see page 671)



- Configuring SPAN for Switch Ports (see page 672)
- Configuring SPAN for Bonds (see page 675)
- Configuring ERSPAN (see page 675)
- Selective Spanning (see page 676)
- Removing SPAN Rules (see page 678)
- Monitoring Control Plane Traffic with tcpdump (see page 679)
- Caveats and Errata (see page 679)
- Related Information (see page 680)

Checking Reachability Using ping

`ping` is used to check reachability of a host. `ping` also calculates the time it takes for packets to travel the round trip. See `man ping` for details.

To test the connection to an IPv4 host:

```
cumulus@switch:~$ ping 192.0.2.45
PING 192.0.2.45 (192.0.2.45) 56(84) bytes of data.
64 bytes from 192.0.2.45: icmp_req=1 ttl=53 time=40.4 ms
64 bytes from 192.0.2.45: icmp_req=2 ttl=53 time=39.6 ms
...
...
```

To test the connection to an IPv6 host:

```
cumulus@switch:~$ ping6 -I swp1 2001::db8:ff:fe00:2
PING 2001::db8:ff:fe00:2(2001::db8:ff:fe00:2) from 2001::db8:ff:fe00:1
    swp1: 56 data bytes
64 bytes from 2001::db8:ff:fe00:2: icmp_seq=1 ttl=64 time=1.43 ms
64 bytes from 2001::db8:ff:fe00:2: icmp_seq=2 ttl=64 time=0.927 ms
```

Printing Route Trace Using traceroute

`traceroute` tracks the route that packets take from an IP network on their way to a given host. See `man traceroute` for details.

To track the route to an IPv4 host:

```
cumulus@switch:~$ traceroute www.google.com
traceroute to www.google.com (74.125.239.49), 30 hops max, 60 byte
packets
1  cumulusnetworks.com (192.168.1.1)  0.614 ms  0.863 ms  0.932 ms
...
5  core2-1-1-0.pao.net.google.com (198.32.176.31)  22.347 ms  22.584
ms  24.328 ms
6  216.239.49.250 (216.239.49.250)  24.371 ms  25.757 ms  25.987 ms
7  72.14.232.35 (72.14.232.35)  27.505 ms  22.925 ms  22.323 ms
```



```
8 nuq04s19-in-f17.1e100.net (74.125.239.49) 23.544 ms 21.851 ms 22  
.604 ms
```

Manipulating the System ARP Cache

`arp` manipulates or displays the kernel's IPv4 network neighbor cache. See `man arp` for details.

To display the ARP cache:

```
cumulus@switch:~$ arp -a  
? (11.0.2.2) at 00:02:00:00:00:10 [ether] on swp3  
? (11.0.3.2) at 00:02:00:00:00:01 [ether] on swp4  
? (11.0.0.2) at 44:38:39:00:01:c1 [ether] on swp1
```

To delete an ARP cache entry:

```
cumulus@switch:~$ arp -d 11.0.2.2  
cumulus@switch:~$ arp -a  
? (11.0.2.2) at <incomplete> on swp3  
? (11.0.3.2) at 00:02:00:00:00:01 [ether] on swp4  
? (11.0.0.2) at 44:38:39:00:01:c1 [ether] on swp1
```

To add a static ARP cache entry:

```
cumulus@switch:~$ arp -s 11.0.2.2 00:02:00:00:00:10  
cumulus@switch:~$ arp -a  
? (11.0.2.2) at 00:02:00:00:00:10 [ether] PERM on swp3  
? (11.0.3.2) at 00:02:00:00:00:01 [ether] on swp4  
? (11.0.0.2) at 44:38:39:00:01:c1 [ether] on swp1
```

Generating Traffic Using mz

`mz` is a fast traffic generator. It can generate a large variety of packet types at high speed. See `man mz` for details.

For example, to send two sets of packets to TCP port 23 and 24, with source IP 11.0.0.1 and destination 11.0.0.2, do the following:

```
cumulus@switch:~$ sudo mz swp1 -A 11.0.0.1 -B 11.0.0.2 -c 2 -v -t tcp  
"dp=23-24"  
  
Mausezahn 0.40 - (C) 2007-2010 by Herbert Haas - http://www.perihel.at  
/sec/mz/  
Use at your own risk and responsibility!  
-- Verbose mode --
```

```
This system supports a high resolution clock.
The clock resolution is 4000250 nanoseconds.
Mausezahn will send 4 frames...
IP: ver=4, len=40, tos=0, id=0, frag=0, ttl=255, proto=6, sum=0, SA=
11.0.0.1, DA=11.0.0.2,
payload=[see next layer]
TCP: sp=0, dp=23, S=42, A=42, flags=0, win=10000, len=20, sum=0,
payload=

IP: ver=4, len=40, tos=0, id=0, frag=0, ttl=255, proto=6, sum=0, SA=
11.0.0.1, DA=11.0.0.2,
payload=[see next layer]
TCP: sp=0, dp=24, S=42, A=42, flags=0, win=10000, len=20, sum=0,
payload=

IP: ver=4, len=40, tos=0, id=0, frag=0, ttl=255, proto=6, sum=0, SA=
11.0.0.1, DA=11.0.0.2,
payload=[see next layer]
TCP: sp=0, dp=23, S=42, A=42, flags=0, win=10000, len=20, sum=0,
payload=

IP: ver=4, len=40, tos=0, id=0, frag=0, ttl=255, proto=6, sum=0, SA=
11.0.0.1, DA=11.0.0.2,
payload=[see next layer]
TCP: sp=0, dp=24, S=42, A=42, flags=0, win=10000, len=20, sum=0,
payload=
```

Creating Counter ACL Rules

In Linux, all ACL rules are always counted. To create an ACL rule for counting purposes only, set the rule action to ACCEPT. See the [Netfilter \(see page 121\)](#) chapter for details on how to use cl-acltool to set up iptables-/ip6tables-/ebtables-based ACLs.



Always place your rules files under /etc/cumulus/acl/policy.d/.

To count all packets going to a Web server:

```
cumulus@switch:~$ cat sample_count.rules

[iptables]
-A FORWARD -p tcp --dport 80 -j ACCEPT

cumulus@switch:~$ sudo cl-acltool -i -p sample_count.rules
Using user provided rule file sample_count.rules
Reading rule file sample_count.rules ...
Processing rules in file sample_count.rules ...
Installing acl policy... done.
```

```
cumulus@switch:~$ sudo iptables -L -v
Chain INPUT (policy ACCEPT 16 packets, 2224 bytes)
pkts bytes target     prot opt in      out      source
destination

Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
pkts bytes target     prot opt in      out      source
destination
    2  156 ACCEPT      tcp  --  any     any     anywhere
anywhere          tcp dpt:http

Chain OUTPUT (policy ACCEPT 44 packets, 8624 bytes)
pkts bytes target     prot opt in      out      source
destination
```



The `-p` option clears out all other rules, and the `-i` option is used to reinstall all the rules.

Configuring SPAN and ERSPAN

SPAN (Switched Port Analyzer) provides for the mirroring of all packets coming in from or going out of an interface (the *SPAN source*), and being copied and transmitted out of a local port (the *SPAN destination*) for monitoring. The SPAN destination port is also referred to as a mirror-to-port (MTP). The original packet is still switched, while a mirrored copy of the packet is sent out of the MTP.

ERSPAN (Encapsulated Remote SPAN) enables the mirrored packets to be sent to a monitoring node located anywhere across the routed network. The switch finds the outgoing port of the mirrored packets by doing a lookup of the destination IP address in its routing table. The original L2 packet is encapsulated with GRE for IP delivery. The encapsulated packets have the following format:

```
-----| MAC_HEADER | IP_HEADER | GRE_HEADER | L2_Mirrored_Packet |
```



Mirrored traffic is not guaranteed. If the MTP is congested, mirrored packets may be discarded.

SPAN and ERSPAN are configured via `c1-acltool`, the [same utility for security ACL configuration \(see page 121\)](#). The match criteria for SPAN and ERSPAN is usually an interface; for more granular match terms, use [selective spanning \(see page 676\)](#). The SPAN source interface can be a port, a subinterface or a bond interface. Both ingress and egress traffic on interfaces can be matched.

Cumulus Linux supports a maximum of 2 SPAN destinations. Multiple rules (SPAN sources) can point to the same SPAN destination, although a given SPAN source cannot specify 2 SPAN destinations. The SPAN destination (MTP) interface can be a physical port, a subinterface, or a bond interface. The SPAN/ERSPAN action is independent of security ACL actions. If packets match both a security ACL rule and a SPAN rule, both actions will be carried out.





Always place your rules files under /etc/cumulus/acl/policy.d/.

Configuring SPAN for Switch Ports

This section describes how to set up, install, verify and uninstall SPAN rules. In the examples that follow, you will span (mirror) switch port swp4 input traffic and swp4 output traffic to destination switch port swp19.

First, create a rules file in /etc/cumulus/acl/policy.d/:

```
cumulus@switch:~$ sudo bash -c 'cat <<EOF > /etc/cumulus/acl/policy.d/span.rules
[iptables]
-A FORWARD --in-interface swp4 -j SPAN --dport swp19
-A FORWARD --out-interface swp4 -j SPAN --dport swp19
EOF'
```



Using cl-acltool with the --out-interface rule applies to transit traffic only; it does not apply to traffic sourced from the switch.

Next, verify all the rules that are currently installed:

```
cumulus@switch:~$ sudo iptables -L -v
Chain INPUT (policy ACCEPT 0 packets, 0 bytes)
  pkts bytes target     prot opt in     out    source
destination
      0     0 DROP       all   --  swp+   any    240.0.0.0/5
anywhere
      0     0 DROP       all   --  swp+   any    loopback/8
anywhere
      0     0 DROP       all   --  swp+   any    base-address.mcast.net/
8 anywhere
      0     0 DROP       all   --  swp+   any    255.255.255.255
anywhere
      0     0 SETCLASS   ospf  --  swp+   any    anywhere
anywhere          SETCLASS  class:7
      0     0 POLICE    ospf  --  any    any    anywhere
anywhere          POLICE   mode:pkt rate:2000 burst:2000
      0     0 SETCLASS   tcp   --  swp+   any    anywhere
anywhere          tcp dpt:bgp SETCLASS  class:7
      0     0 POLICE    tcp   --  any    any    anywhere
anywhere          tcp dpt:bgp POLICE  mode:pkt rate:2000 burst:2000
      0     0 SETCLASS   tcp   --  swp+   any    anywhere
anywhere          tcp spt:bgp SETCLASS  class:7
      0     0 POLICE    tcp   --  any    any    anywhere
anywhere          tcp spt:bgp POLICE  mode:pkt rate:2000 burst:2000
```



```
0      0  SETCLASS    tcp  --  swp+  any      anywhere
anywhere                      tcp dpt:5342  SETCLASS  class:7
0      0  POLICE     tcp  --  any   any      anywhere
anywhere                      tcp dpt:5342  POLICE   mode:pkt rate:2000 burst:200
0
0      0  SETCLASS    tcp  --  swp+  any      anywhere
anywhere                      tcp spt:5342  SETCLASS  class:7
0      0  POLICE     tcp  --  any   any      anywhere
anywhere                      tcp spt:5342  POLICE   mode:pkt rate:2000 burst:200
0
0      0  SETCLASS    icmp --  swp+  any      anywhere
anywhere                      SETCLASS  class:2
0      0  POLICE     icmp --  any   any      anywhere
anywhere                      POLICE   mode:pkt rate:100  burst:40
15    5205  SETCLASS  udp   --  swp+  any      anywhere
anywhere                      udp dpts:bootps:bootpc  SETCLASS  class:2
11    3865  POLICE   udp   --  any   any      anywhere
anywhere                      udp dpt:bootps  POLICE   mode:pkt rate:100  burst:10
0
0      0  POLICE     udp   --  any   any      anywhere
anywhere                      udp dpt:bootpc  POLICE   mode:pkt rate:100  burst:10
0
0      0  SETCLASS    tcp  --  swp+  any      anywhere
anywhere                      tcp dpts:bootps:bootpc  SETCLASS  class:2
0      0  POLICE     tcp  --  any   any      anywhere
anywhere                      tcp dpt:bootps  POLICE   mode:pkt rate:100  burst:10
0
0      0  POLICE     tcp  --  any   any      anywhere
anywhere                      tcp dpt:bootpc  POLICE   mode:pkt rate:100  burst:10
0
17    1088  SETCLASS  igmp --  swp+  any      anywhere
anywhere                      SETCLASS  class:6
17    1156  POLICE   igmp --  any   any      anywhere
anywhere                      POLICE   mode:pkt rate:300  burst:100
394   41060  POLICE  all   --  swp+  any      anywhere
anywhere                      ADDRTYPE match dst-type LOCAL POLICE   mode:pkt
rate:1000 burst:1000  class:0
0      0  POLICE     all   --  swp+  any      anywhere
anywhere                      ADDRTYPE match dst-type IPROUTER POLICE   mode:
pkt rate:400 burst:100  class:0
988   279K  SETCLASS  all   --  swp+  any      anywhere
anywhere                      SETCLASS  class:0

Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out      source
destination
0      0  DROP       all   --  swp+  any      240.0.0.0/5
anywhere
0      0  DROP       all   --  swp+  any      loopback/8
anywhere
0      0  DROP       all   --  swp+  any      base-address.mcast.net/
8      anywhere
```

```

      0      0  DROP      all  --  swp+    any      255.255.255.255
anywhere
26864 4672K SPAN      all  --  swp4    any      anywhere
anywhere      dport:swp19  <---- input packets on swp4

40722   47M SPAN      all  --  any     swp4    anywhere
anywhere      dport:swp19  <---- output packets on swp4

Chain OUTPUT (policy ACCEPT 67398 packets, 5757K bytes)
 pkts bytes target      prot opt in      out      source
destination

```

Install the rules:

```

cumulus@switch:~$ sudo cl-acltool -i
[sudo] password for cumulus:
Reading rule file /etc/cumulus/acl/policy.d/00control_plane.rules ...
Processing rules in file /etc/cumulus/acl/policy.d/00control_plane.
rules ...
Reading rule file /etc/cumulus/acl/policy.d/99control_plane_catch_all.
rules ...
Processing rules in file /etc/cumulus/acl/policy.d/99control_plane_catch_all.rules ...
Reading rule file /etc/cumulus/acl/policy.d/span.rules ...
Processing rules in file /etc/cumulus/acl/policy.d/span.rules ...
Installing acl policy
done.

```



Running the following command is incorrect and will remove **all** existing control-plane rules or other installed rules and only install the rules defined in `span.rules`:

```

cumulus@switch:~$ sudo cl-acltool -i -P /etc/cumulus/acl/policy.d
/span.rules

```

Verify that the SPAN rules were installed:

```

cumulus@switch:~$ sudo cl-acltool -L all | grep SPAN
38025 7034K SPAN      all  --  swp4    any      anywhere
anywhere      dport:swp19
50832   55M SPAN      all  --  any     swp4    anywhere
anywhere      dport:swp19

```



Configuring SPAN for Bonds

This section describes how to configure SPAN for all packets going out of `bond0` locally to `bond1`.

First, create a rules file in `/etc/cumulus/acl/policy.d/`:

```
cumulus@switch:~$ sudo bash -c 'cat <<EOF > /etc/cumulus/acl/policy.d
/span_bond.rules
[iptables]
-A FORWARD --out-interface bond0 -j SPAN --dport bond1
EOF'
```



Using `cl-acltool` with the `--out-interface` rule applies to transit traffic only; it does not apply to traffic sourced from the switch.

Install the rules:

```
cumulus@switch:~$ sudo cl-acltool -i
[sudo] password for cumulus:
Reading rule file /etc/cumulus/acl/policy.d/00control_plane.rules ...
Processing rules in file /etc/cumulus/acl/policy.d/00control_plane.
rules ...
Reading rule file /etc/cumulus/acl/policy.d/99control_plane_catch_all.
rules ...
Processing rules in file /etc/cumulus/acl/policy.d/99control_plane_catch_all.rules ...
Reading rule file /etc/cumulus/acl/policy.d/span_bond.rules ...
Processing rules in file /etc/cumulus/acl/policy.d/span_bond.rules ...
Installing acl policy
done.
```

Verify that the SPAN rules were installed:

```
cumulus@switch:~$ sudo iptables -L -v | grep SPAN
 19  1938  SPAN      all  --  any    bond0    anywhere
 anywhere          dport:bond1
```

Configuring ERSPAN

This section describes how to configure ERSPAN for all packets coming in from `swp1` to 12.0.0.2:

First, create a rules file in `/etc/cumulus/acl/policy.d/`:

```
cumulus@switch:~$ sudo bash -c 'cat <<EOF > /etc/cumulus/acl/policy.d
/erspan.rules
EOF'
```

```
[iptables]
-A FORWARD --in-interface swp1 -j ERSPAN --src-ip 12.0.0.1 --dst-ip 12
.0.0.2 --ttl 64
EOF'
```

Install the rules:

```
cumulus@switch:~$ sudo cl-acltool -i
Reading rule file /etc/cumulus/acl/policy.d/00control_plane.rules ...
Processing rules in file /etc/cumulus/acl/policy.d/00control_plane.
rules ...
Reading rule file /etc/cumulus/acl/policy.d/99control_plane_catch_all.
rules ...
Processing rules in file /etc/cumulus/acl/policy.d/99control_plane_catch_all.rules ...
Reading rule file /etc/cumulus/acl/policy.d/erspan.rules ...
Processing rules in file /etc/cumulus/acl/policy.d/erspan.rules ...
Installing acl policy
done.
```

Verify that the ERSPAN rules were installed:

```
cumulus@switch:~$ sudo iptables -L -v | grep SPAN
 69 6804 ERSPAN      all -- swp1    any     anywhere
anywhere          ERSPAN src-ip:12.0.0.1 dst-ip:12.0.0.2
```

The `src-ip` option can be any IP address, whether it exists in the routing table or not. The `dst-ip` option must be an IP address reachable via the routing table. The destination IP address must be reachable from a front-panel port, and not the management port. Use `ping` or `ip route get <ip>` to verify that the destination IP address is reachable. Setting the `--ttl` option is recommended.



When using [Wireshark](#) to review the ERSPAN output, Wireshark may report the message "Unknown version, please report or test to use fake ERSPAN preference", and the trace is unreadable. To resolve this, go into the General preferences for Wireshark, then go to **Protocols** > **ERSPAN** and check the **Force to decode fake ERSPAN frame** option.

Selective Spanning

SPAN/ERSPAN traffic rules can be configured to limit the traffic that is spanned, to reduce the volume of copied data.



Cumulus Linux 3.2 supports selective spanning for `iptables` only. `ip6tables` and `eptables` are not supported.

The following matching fields are supported:

- IPv4 SIP/DIP
- IP protocol
- L4 (TCP/UDP) src/dst port
- TCP flags
- An ingress port/wildcard (swp+) can be specified in addition



With ERSPAN, a maximum of two --src-ip --dst-ip pairs are supported. Exceeding this limit produces an error when you install the rules with `c1-acltool`.

SPAN Examples

- To mirror forwarded packets from all ports matching SIP 20.0.1.0 and DIP 20.0.1.2 to port swp1s1:

```
-A FORWARD --in-interface swp+ -s 20.0.0.2 -d 20.0.1.2 -j SPAN --
dport swp1s2
```

- To mirror icmp packets from all ports to swp1s2:

```
-A FORWARD --in-interface swp+ -s 20.0.0.2 -p icmp -j SPAN --
dport swp1s2
```

- To mirror forwarded UDP packets received from port swp1s0, towards DIP 20.0.1.2 and destination port 53:

```
-A FORWARD --in-interface swp1s0 -d 20.0.1.2 -p udp --dport 53 -
j SPAN --dport swp1s2
```

- To mirror all forwarded TCP packets with only SYN set:

```
-A FORWARD --in-interface swp+ -p tcp --tcp-flags ALL SYN -j
SPAN --dport swp1s2
```

- To mirror all forwarded TCP packets with only FIN set:

```
-A FORWARD --in-interface swp+ -p tcp --tcp-flags ALL FIN -j
SPAN --dport swp1s2
```

ERSPAN Examples

- To mirror forwarded packets from all ports matching SIP 20.0.1.0 and DIP 20.0.1.2:

```
-A FORWARD --in-interface swp+ -s 20.0.0.2 -d 20.0.1.2 -j ERSPAN
--src-ip 90.0.0.1 --dst-ip 20.0.2.2
```

- To mirror ICMP packets from all ports:

```
-A FORWARD --in-interface swp+ -s 20.0.0.2 -p icmp -j ERSPAN --
src-ip 90.0.0.1 --dst-ip 20.0.2.2
```

- To mirror forwarded UDP packets received from port swp1s0, towards DIP 20.0.1.2 and destination port 53:

```
-A FORWARD --in-interface swp1s0 -d 20.0.1.2 -p udp --dport 53 -
-j ERSPAN --src-ip 90.0.0.1 --dst-ip 20.0.2.2
```

- To mirror all forwarded TCP packets with only SYN set:

```
-A FORWARD --in-interface swp+ -p tcp --tcp-flags ALL SYN -j
ERSPAN --src-ip 90.0.0.1 --dst-ip 20.0.2.2
```

- To mirror all forwarded TCP packets with only FIN set:

```
-A FORWARD --in-interface swp+ -p tcp --tcp-flags ALL FIN -j
ERSPAN --src-ip 90.0.0.1 --dst-ip 20.0.2.2
```

Removing SPAN Rules

To remove your SPAN rules, run:

```
#Remove rules file:
cumulus@switch:~$ sudo rm /etc/cumulus/acl/policy.d/span.rules
#Reload the default rules
cumulus@switch:~$ sudo cl-acltool -i
cumulus@switch:~$
```

To verify that the SPAN rules were removed:

```
cumulus@switch:~$ sudo cl-acltool -L all | grep SPAN
cumulus@switch:~$
```



Monitoring Control Plane Traffic with tcpdump

You can use `tcpdump` to monitor control plane traffic — traffic sent to and coming from the switch CPUs. `tcpdump` does **not** monitor data plane traffic; use `cl-acltool` instead (see above).

For more information on `tcpdump`, read the [tcpdump documentation](#) and the [tcpdump man page](#).

The following example incorporates a few `tcpdump` options:

- `-i bond0`, which captures packets from `bond0` to the CPU and from the CPU to `bond0`
- `host 169.254.0.2`, which filters for this IP address
- `-c 10`, which captures 10 packets then stops

```
cumulus@switch:~$ sudo tcpdump -i bond0 host 169.254.0.2 -c 10
tcpdump: WARNING: bond0: no IPv4 address assigned
tcpdump: verbose output suppressed, use -v or -vv for full protocol
decode
listening on bond0, link-type EN10MB (Ethernet), capture size 65535
bytes
16:24:42.532473 IP 169.254.0.2 > 169.254.0.1: ICMP echo request, id 27
785, seq 6, length 64
16:24:42.532534 IP 169.254.0.1 > 169.254.0.2: ICMP echo reply, id 2778
5, seq 6, length 64
16:24:42.804155 IP 169.254.0.2.40210 > 169.254.0.1.5342: Flags [.],
seq 266275591:266277039, ack 3813627681, win 58, options [nop,nop,TS
val 590400681 ecr 530346691], length 1448
16:24:42.804228 IP 169.254.0.1.5342 > 169.254.0.2.40210: Flags [.],
ack 1448, win 166, options [nop,nop,TS val 530348721 ecr 590400681],
length 0
16:24:42.804267 IP 169.254.0.2.40210 > 169.254.0.1.5342: Flags [P.],
seq 1448:1836, ack 1, win 58, options [nop,nop,TS val 590400681 ecr 53
0346691], length 388
16:24:42.804293 IP 169.254.0.1.5342 > 169.254.0.2.40210: Flags [.],
ack 1836, win 165, options [nop,nop,TS val 530348721 ecr 590400681],
length 0
16:24:43.532389 IP 169.254.0.2 > 169.254.0.1: ICMP echo request, id 27
785, seq 7, length 64
16:24:43.532447 IP 169.254.0.1 > 169.254.0.2: ICMP echo reply, id 2778
5, seq 7, length 64
16:24:43.838652 IP 169.254.0.1.59951 > 169.254.0.2.5342: Flags [.],
seq 2555144343:2555145791, ack 2067274882, win 58, options [nop,nop,
TS val 530349755 ecr 590399688], length 1448
16:24:43.838692 IP 169.254.0.1.59951 > 169.254.0.2.5342: Flags [P.],
seq 1448:1838, ack 1, win 58, options [nop,nop,TS val 530349755 ecr 59
0399688], length 390
10 packets captured
12 packets received by filter
0 packets dropped by kernel
```



Caveats and Errata

- SPAN rules cannot match outgoing subinterfaces.
- ERSPAN rules must include `ttl` for versions 1.5.1 and earlier.

Related Information

- www.perihel.at/sec/mz/mzguide.html
- en.wikipedia.org/wiki/Ping
- www.tcpdump.org
- en.wikipedia.org/wiki/Traceroute

Using NCLU to Troubleshoot Your Network Configuration

The `network` command line utility (see page 80) (NCLU) can quickly return a lot of information about your network configuration.

Contents

This chapter covers ...

- Using `net show` Commands (see page 680)
- Showing Interfaces (see page 681)
- Other Useful Features (see page 682)
- Installing `netshow` on a Linux Server (see page 682)

Using `net show` Commands

Running `net show` and pressing TAB displays all available command line arguments usable by `net`. The output looks like this:

```
cumulus@switch:~$ net show <TAB>
  bgp          : Border Gateway Protocol
  bridge       : A layer2 bridge
  clag         : Multi-Chassis Link Aggregation
  commit       : apply the commit buffer to the system
  configuration : Settings, configuration state, etc
  counters     : show netstat counters
  hostname     : System hostname
  igmp         : Internet Group Management Protocol
  interface    : An interface such as swp1, swp2, etc
  ip           : Internet Protocol version 4
  ipv6         : Internet Protocol version 6
  lldp         : Link Layer Discovery Protocol
  lnv          : Lightweight Network Virtualization
  mroute       : Configure static unicast route into MRIB for
  multicast RPF lookup
  msdp         : Multicast Source Discovery Protocol
  ospf          : Open Shortest Path First (OSPFv2)
```



ospf6	:	Open Shortest Path First (OSPFv3)
pim	:	Protocol Independent Multicast
rollback	:	revert to a previous configuration state
route	:	Static routes
route-map	:	Route-map
system	:	System information
version	:	Version number

Showing Interfaces

To show all available interfaces that are physically UP, run `net show interface`:

```
cumulus@switch:~$ net show interface

      Name      Speed      MTU      Mode           Summary
--  -----  -----  -----
----- 
UP    lo      N/A      65536   Loopback      IP: 10.0.0.11/32, 127.0.0.1
/8 , ::1/128
UP    eth0     1G      1500    Mgmt          IP: 192.168.0.11/24(DHCP)
UP    swp1     1G      1500    Access/L2     Untagged: br0
UP    swp2     1G      1500    NotConfigured
UP    swp51    1G      1500    NotConfigured
UP    swp52    1G      1500    NotConfigured
UP    blue     N/A      65536   NotConfigured
UP    br0      N/A      1500    Bridge/L3    IP: 172.16.1.1/24
                                         Untagged Members: swp1
                                         802.1q Tag: Untagged
                                         STP: RootSwitch(32768)
UP    red      N/A      65536   NotConfigured
```

Whereas `net show interface all` displays every interface regardless of state:

```
cumulus@switch:~$ net show interface all

      Name      Speed      MTU      Mode           Summary
--  -----  -----  -----
----- 
UP    lo      N/A      65536   Loopback      IP: 10.0.0.11/32, 127.0
.0.1/8 , ::1/128
UP    eth0     1G      1500    Mgmt          IP: 192.168.0.11/24(DHC
P)
UP    swp1     1G      1500    Access/L2     Untagged: br0
UP    swp2     1G      1500    NotConfigured
ADMDN  swp45    0M      1500    NotConfigured
ADMDN  swp46    0M      1500    NotConfigured
ADMDN  swp47    0M      1500    NotConfigured
ADMDN  swp48    0M      1500    NotConfigured
ADMDN  swp49    0M      1500    NotConfigured
```

ADMDN	swp50	0M	1500	NotConfigured	
UP	swp51	1G	1500	NotConfigured	
UP	swp52	1G	1500	NotConfigured	
UP	blue	N/A	65536	NotConfigured	
UP	br0	N/A	1500	Bridge/L3	IP: 172.16.1.1/24 Untagged Members: swp1 802.1q Tag: Untagged STP: RootSwitch(32768)
UP	red	N/A	65536	NotConfigured	
ADMDN	vagrant	0M	1500	NotConfigured	

You can get information about the switch itself by running `net show system`:

```
cumulus@switch:~$ net show system
Arctica 3200C
Cumulus 3.2.0
Build: Cumulus Linux 3.2.0
Uptime: 8 days, 0:45:29
```

Other Useful Features

NCLU uses the `python network-docopt` package. This is inspired by `docopt` and provides the ability to specify partial commands, without tab completion and running the complete option. For example:

```
net show int runs netshow interface
net show sys runs netshow system
```

Installing netshow on a Linux Server

`netshow` is a tool developed by Cumulus Networks for troubleshooting networks. In Cumulus Linux, it's been replaced by NCLU. However, NCLU is not available on Linux hosts at this time, so Cumulus Networks recommends you use `netshow` to help troubleshoot servers. To install `netshow` on a Linux server, run:

```
root@host:~# pip install netshow-linux-lib
```



Debian and Red Hat packages will be available in the near future.

Monitoring System Statistics and Network Traffic with sFlow

sFlow is a monitoring protocol that samples network packets, application operations, and system counters. sFlow collects both interface counters and sampled 5-tuple packet information, enabling you to monitor your network traffic as well as your switch state and performance metrics. An outside server, known as an *sFlow collector*, is required to collect and analyze this data.



`hsflowd` is the daemon that samples and sends sFlow data to configured collectors. `hsflowd` is not included in the base Cumulus Linux installation. After installation, `hsflowd` will automatically start when the switch boots up.

Contents

This chapter covers ...

- [Installing hsflowd \(see page 683\)](#)
- [Configuring sFlow \(see page 683\)](#)
 - [Configuring sFlow via DNS-SD \(see page 683\)](#)
 - [Manually Configuring /etc/hsflowd.conf \(see page 684\)](#)
- [Configuring sFlow Visualization Tools \(see page 685\)](#)
- [Related Information \(see page 685\)](#)

Installing hsflowd

To download and install the `hsflowd` package, use `apt-get`:

```
cumulus@switch:~$ sudo apt-get update
cumulus@switch:~$ sudo apt-get install -y hsflowd
```

Configuring sFlow

You can configure `hsflowd` to send to the designated collectors via two methods:

- DNS service discovery (DNS-SD)
- Manually configuring `/etc/hsflowd.conf`

Configuring sFlow via DNS-SD

With this method, you need to configure your DNS zone to advertise the collectors and polling information to all interested clients. Add the following content to the zone file on your DNS server:

```
_sflow._udp SRV 0 0 6343 collector1
_sflow._udp SRV 0 0 6344 collector2
_sflow._udp TXT (
  "txtvers=1"
  "sampling.1G=2048"
  "sampling.10G=4096"
  "sampling.40G=8192"
  "polling=20"
)
```

The above snippet instructs `hsflowd` to send sFlow data to `collector1` on port 6343 and to `collector2` on port 6344. `hsflowd` will poll counters every 20 seconds and sample 1 out of every 2048 packets.



From Cumulus Linux 2.5.3 onwards, the maximum samples/second delivered from the hardware is limited to 16K. This can be configured by editing the `/etc/cumulus/datapath/traffic.conf` file, as shown below:

```
# Set sflow/sample ingress cpu packet rate and burst in packets
/sec
# Values: {0..16384}
#sflow.rate = 16384
#sflow.burst = 16384
```

After the initial configuration is ready, bring up the sFlow daemon by running:

```
cumulus@switch:~$ sudo systemctl start hsflowd.service
```

No additional configuration is required in `/etc/hsflowd.conf`.

Manually Configuring /etc/hsflowd.conf

With this method you will set up the collectors and variables on each switch.

Edit `/etc/hsflowd.conf` and change `DNSSD = on` to `DNSSD = off`:

```
DNSSD = off
```

Then set up your collectors and sampling rates in `/etc/hsflowd.conf`:

```
# Manual Configuration (requires DNSSD=off above)
#####
#
# Typical configuration is to send every 30 seconds
polling = 20

sampling.1G=2048
sampling.10G=4096
sampling.40G=8192

collector {
    ip = 192.0.2.100
    udpport = 6343
}

collector {
    ip = 192.0.2.200
    udpport = 6344
```

{}

This configuration polls the counters every 20 seconds, samples 1 of every 2048 packets and sends this information to a collector at 192.0.2.100 on port 6343 and to another collector at 192.0.2.200 on port 6344.



Some collectors require each source to transmit on a different port, others may listen on only one port. Please refer to the documentation for your collector for more information.

Configuring sFlow Visualization Tools

For information on configuring various sFlow visualization tools, read this [Help Center article](#).

Related Information

- [sFlow Collectors](#)
- [sFlow Wikipedia page](#)

Using netq to Troubleshoot the Network

ⓘ Early Access Feature

`netq` is an [early access feature](#) in Cumulus Linux 3.2.1. Before you can install `netq`, you must enable the Early Access repository. For more information about the Cumulus Linux repository, read this [knowledge base article](#).

`netq` is a tool for troubleshooting the whole network fabric. Instead of using other tools to troubleshoot node by node, `netq` aggregates data from across all the nodes in a network, so you can query and diagnose issues affecting the whole network, analyze outages or discover why two or more switches cannot communicate. `netq` can return a wealth of data about your network for both layer 2 and the layer 3 IP fabric, including:

- Interface history
- MLAG checks
- Anycast IP validation
- Interface address history
- IP neighbor history
- MTU validation of traceroutes
- Route history
- Route origin validation
- Support for VRF analysis and commands

`netq` provides the ability to see the output of commands on other switches, even if a switch is currently unavailable. You can even see the command history so you can go back in time before the issue arose to debug it.

Because `netq` is a Linux application, it's easy to automate with tools like Ansible, Puppet or Chef.



Contents

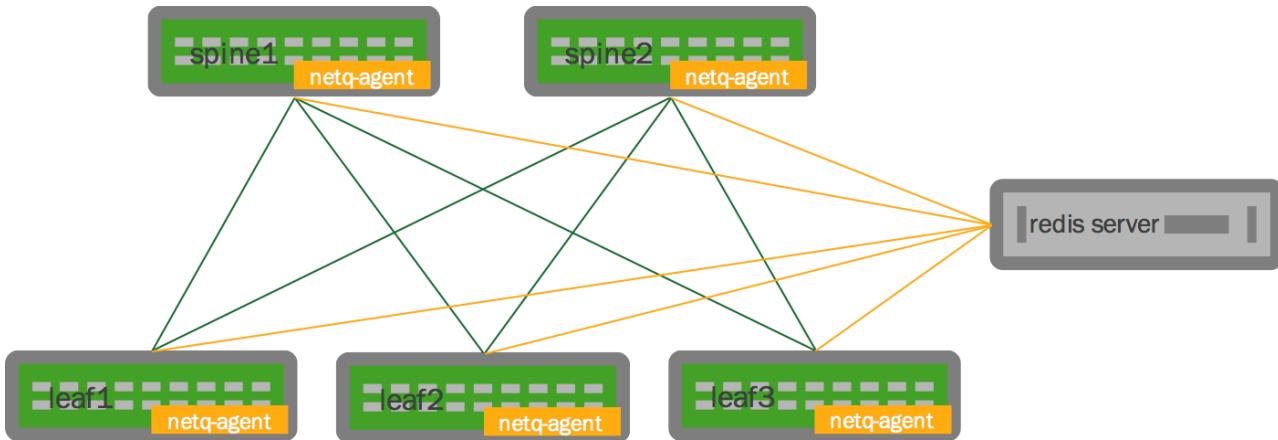
This chapter covers ...

- Components (see page 686)
- Feature Limitations (see page 687)
- Installing netq (see page 687)
 - Installing redis Server (see page 688)
- Configuring the redis Server (see page 688)
- Configuring netq (see page 689)
 - Configuring NetQ in Management VRF (see page 689)
- Using netq (see page 690)
 - Checking the Health of the Network (see page 690)
 - Using netq show (see page 691)
 - Using netq view (see page 695)
- Monitoring the redis Server (see page 696)
 - Specifying a Different redis Server (see page 696)
- Troubleshooting netq (see page 696)

Components

netq has three primary components:

- **netq-agent**: The back end Python agent installed on every Cumulus Linux switch in the network; the agent pushes out data to a central server (a `redis` server, see below) periodically and when specific `netlink` events occur. The `redis` server processes the queries and sends back a response to the switch. The agent listens for these events:
 - address (IPv4 and IPv6)
 - route (IPv4 and IPv6)
 - link
 - bridge fdb
 - IP neighbor
- **netq**: The command line interface to the `netq-agent`. You can use the `netq` CLI on every Cumulus Linux switch as well as the `redis` server.
- **redis** server: The database/key-value store where all network information sent from `netq-agents` running on Cumulus Linux switches is collected and aggregated. The server runs `redis` version 2.8.17-1.



Feature Limitations

`netq` is an [early access feature](#). As such, the following features are limited or unavailable at this time, and should be available in a future release of Cumulus Linux:

- You can check BGP health only at this time; reporting OSPF health should follow in the near future.
- You cannot determine the origin of static routes.
- Clustering the `redis` server for high availability has not been tested.
- `netq` has been tested with up to 20 nodes in a fabric, with 8k routes and MAC addresses per node.

If you are interested in trying out this or any other early access feature, contact your Cumulus Networks account representative to let us know you are testing it.

Installing `netq`

To install the `netq` package — `cumulus-netq` — on a switch, follow the instructions below. The `cumulus-netq` package contains `netq` and the `netq-agent`. Cumulus Networks recommends you install the `netq-agent` on every Cumulus Linux switch in the network; you can also install it on the `redis` server.

1. Open the `/etc/apt/sources.list` file in a text editor.
2. Uncomment the early access repository lines and save the file:

```
deb http://repo3.cumulusnetworks.com/repo CumulusLinux-3-early-access cumulus
deb-src http://repo3.cumulusnetworks.com/repo CumulusLinux-3-early-access cumulus
```

3. Run the following commands in a terminal to install the `cumulus-netq` package:

```
cumulus@switch:~$ sudo apt-get update
cumulus@switch:~$ sudo apt-get install cumulus-netq
```



Installing redis Server

Cumulus Networks recommends you install the `redis` server on its own server or VM. Ideally, you should run the `redis` server on a separate, powerful server for maximum usability and performance — Cumulus Networks recommends a system with a quad core CPU, 16GB of RAM (with 8GB for `redis` itself) and 512GB of storage.

You need to download and install two packages:

- The [redis-server package](#), version 2.8.17-1.
- The [redis-tools package](#), version 2.8.17-1.

Use `apt-get` to install the packages:

```
root@redis-server:~# apt-get update
root@redis-server:~# apt-get install redis-server redis-tools
```

After you install these packages, connect the server over the management network to ensure network connectivity even if the in-band network is unavailable. Note its IP address, as you need to specify it when you configure `netq`.

⚠ If you want to run the `netq` CLI on the `redis` server, you need to install the `cumulus-netq` package on the server. You'll need to update your `sources.list` file to include the [Cumulus Linux repository](#). The `cumulus-netq` package contains the `netq` client, which contains the CLI.

⚠ You cannot specify a port number for the `redis` server at this time.

Once you install the `redis` server, you must configure it before you can configure `netq` on the switch.

Configuring the redis Server

Depending upon the operating system of the `redis` server host, you may need to modify its configuration before it can start monitoring the network. Check the `/etc/redis/redis.conf` file and verify that the server is listening to external-facing ports, and not the localhost.

1. Edit `/etc/redis/redis.conf`:

```
root@redis-server:~# vi /etc/redis/redis.conf
```

2. If the bind line links to localhost (127.0.0.1), change it to the IP address of one or more external ports, such as `eth0`:

```
bind 192.0.2.240
```



3. Restart the `redis-server` service. For example, on a Debian host, run:

```
root@redis-server:~# systemctl restart redis-server
```

Configuring netq

Once you install the `netq` packages and configure the redis server, you need to configure `netq` to monitor your network.

1. To ensure useful output, ensure that [NTP \(see page 87\)](#) is running.
2. Specify the IP address of the `redis` server. For example:

```
cumulus@switch:~$ sudo netq add server 198.51.100.1
```

3. Start the `netq` agent.

```
cumulus@switch:~$ sudo netq agent start
```



If you see the following error, it means you haven't added the redis server or the server wasn't configured:

```
cumulus@switch:~$ sudo netq agent start
Error: Please specify IP address of DB server
```

The `netq` configuration is stored in the following files:

- `/etc/netq/netq-agent.conf`: Contains basic agent configuration, including the `redis` server IP address.
- `/etc/netq/netq-agent-commands.json`: Contains key-value pairs of commands whose outputs are pushed along with the key to be associated, periodicity of push and so forth.
- `/etc/netq/netq-agent-running.json`: Log of the actual commands that are being pushed to the agent, as determined when the agent starts, as well as the `redis` server IP address and more.

Configuring NetQ in Management VRF

To configure NetQ in the management VRF, after you configure the management VRF, start the `netq` service as followed:

```
cumulus@switch:mgmt-vrf:~$ sudo systemctl start netq-agent@mgmt.service
```



Only one instance of `netq-agent` can run at a time. If the `netq-agent@mgmt.service` has not been started, check if the default `netq-agent.service` is running. If it is, stop the default service.

```
cumulus@switch:mgmt-vrf:~$ sudo systemctl stop netq-agent
```

Using `netq`

`netq` has a number of options to use with the command to return various kinds of data about your network — press the *Tab* key at any time to reveal the options available to a given part of the command. Running `netq` on its own reveals all the options, with a brief explanation for each one:

```
cumulus@switch:~$ netq <TAB>
add      : Update configuration
agent    : Netq agent
check    : check health of services or correctness of parameter
help     : Show usage info
resolve  : Annotate input with names and interesting info
server   : IP address of DB server
show     : Show fabric-wide info
trace    : Control plane trace path across fabric
view    : Show output of pre-defined commands on specific node
```

Checking the Health of the Network

It's best to start with `netq check agents` to see the status of every network node, based on the whether the agent missed receiving any heartbeats sent from the node. A node's status can be one of the following:

- **Fresh:** The agent is running fine, no heartbeats were missed.
- **Stale:** The agent missed one heartbeat, which is not unusual.
- **Rotten:** The agent missed five consecutive heartbeats.

cumulus@switch:~\$ sudo netq check agents				
Node Name	Connect Time	Last Connect	Status	
leaf-1	2017-02-09 21:44:34	7s ago	Fresh	
leaf-2	2017-02-09 21:44:39	just now	Fresh	
leafc-11	2017-02-09 21:44:12	29s ago	Fresh	
leafc-12	2017-02-09 21:44:18	23s ago	Fresh	
leafc-21	2017-02-09 21:44:23	18s ago	Fresh	
leafc-22	2017-02-09 21:44:29	11s ago	Fresh	
noc-pr	2017-02-09 21:43:50	19s ago	Fresh	



noc-se	2017-02-09 21:43:56	15s ago	Fresh
spine-1	2017-02-09 21:44:01	9s ago	Fresh
spine-2	2017-02-09 21:44:07	4s ago	Fresh

You can also check the health of BGP and MLAG in the network:

```
cumulus@leaf-1:~$ netq check bgp
Total Sessions: 28, Failed Sessions: 0

cumulus@leaf-1$ netq check clag
Checked Nodes: 5, Failed Nodes: 1
Node           Reason
-----
leafc-11      Peer Connectivity failed
leafc-11      Singly Attached Bonds: hostbond4, hostbond5
```

Using `netq show`

The `netq show` command can return information regarding the network fabric overall, including:

- Interface IP and MAC addresses across all nodes in the fabric.
- Interfaces across all nodes in the fabric.
- IPv4 and IPv6-related information.
- LLDP-based neighbor information.
- MAC address information, with VLANs if present, across the fabric.

The `netq show` command takes the following options:

```
cumulus@switch:~$ netq show <TAB>
bgp          : Check the status of all the BGP sessions across the
fabric
changes      : Show changes in interface/mac/route/neighbor state
between specified times
clag         : Check the status of all the CLAG daemons across the
fabric
interfaces   : Interfaces across all nodes in fabric
inventory    : add help text
ip           : IPv4 related info
ipv6         : IPv6 related info
lldp         : LLDP based neighbor info
macs         : Show mac address info, with optional VLAN, across the
fabric
services     : Show various system services
stp          : Spanning Tree
top_talkers   : Top interface talkers in fabric
```

To see the hardware information for all the nodes across the fabric, run:

```
cumulus@switch:~$ netq show inventory
Node          Switch        OS           CPU      ASIC       Ports
-----  -----
dell-z9100-05 Z9100-ON    Cumulus Linux x86_64   Tomahawk   32 x
100G-QSFP28
mlx-2100-05   SN2100      Cumulus Linux x86_64   Spectrum   16 x
100G-QSFP28
mlx-2410-03   SN2410      Cumulus Linux x86_64   Spectrum   48 x
25G-SFP28 & 8 x 100G-QSFP28
mlx-2700-11   SN2700      Cumulus Linux x86_64   Spectrum   32 x
100G-QSFP28
st1-11        Arctica 4806XP Cumulus Linux x86_64   Trident2  48 x
10G-SFP+ & 6 x 40G-QSFP+
st1-12        Arctica 4806XP Cumulus Linux x86_64   Trident2  48 x
10G-SFP+ & 6 x 40G-QSFP+
st1-13        Arctica 4806XP Cumulus Linux x86_64   Trident2  48 x
10G-SFP+ & 6 x 40G-QSFP+
st1-s1        S6000-ON    Cumulus Linux x86_64   Trident2  32 x
40G-QSFP+
st1-s2        S6000-ON    Cumulus Linux x86_64   Trident2  32 x
40G-QSFP+
```

To see MAC address information for switch leaf-1, run:

```
cumulus@leaf-1:~$ netq show macs leaf-1
MAC          VLAN     Node Name      Egress Port
Origin Last Changed
-----  -----
00:02:00:00:00:16  100     leaf-1      VlanA-1
1      4m ago
00:02:00:00:00:16  101     leaf-1      VlanA-1
1      4m ago
00:02:00:00:00:16  102     leaf-1      VlanA-1
1      4m ago
00:02:00:00:00:16  103     leaf-1      VlanA-1
1      4m ago
00:02:00:00:00:16  104     leaf-1      VlanA-1
1      4m ago
00:02:00:00:00:16  105     leaf-1      VlanA-1
1      4m ago
00:02:00:00:00:16  106     leaf-1      VlanA-1
1      4m ago
00:02:00:00:00:16  Intf    leaf-1      hostbond3
1      4m ago
```



```
00:02:00:00:00:18      Intf      leaf-1      hostbond4
1          4m ago
02:02:00:00:00:09      100       leaf-1      hostbond3
1          4m ago
```

You can filter the output to a given interface on a switch, in this case hostbond4 on leaf-1:

```
cumulus@spine-1:~$ netq show macs leaf-1 nexthop hostbond4
MAC           VLAN      Node Name      Egress Port
Origin Last Changed
-----
-----
00:02:00:00:00:18      Intf      leaf-1      hostbond4
1          7m ago
02:02:00:00:00:0b      100       leaf-1      hostbond4
1          7m ago
02:02:00:00:00:0c      100       leaf-1      hostbond4
1          7m ago
```

To see the route information, run:

```
cumulus@switch:~$ netq show ip route 3.0.3.3
Route info about prefix 3.0.3.3 on host *
Origin Table          IP             Node
Nexthops              Last Changed
-----
-----
0      default          0.0.0.0/0      noc-pr
192.168.0.2: eth0     11m ago
0      default          0.0.0.0/0      noc-se
192.168.0.2: eth0     11m ago
0      default          3.0.3.0/26    leaf-1
169.254.0.1: uplink-1, 10m ago

169.254.0.1: uplink-2
0      default          3.0.3.0/26    leaf-2
169.254.0.1: uplink-1, 10m ago

169.254.0.1: uplink-2
0      default          3.0.3.0/26    leafc-11
169.254.0.1: uplink-1, 10m ago

169.254.0.1: uplink-2
0      default          3.0.3.0/26    leafc-12
169.254.0.1: uplink-1, 5m ago

169.254.0.1: uplink-2
0      default          3.0.3.0/26    spine-1
169.254.0.1: downlink-3, 10m ago
```

```

169.254.0.1: downlink-4
0      default          3.0.3.0/26      spine-2
169.254.0.1: downlink-3,   10m ago

169.254.0.1: downlink-4
1      default          3.0.3.0/26      leafc-21      VlanA-
1.105           10m ago
1      default          3.0.3.3/32      leafc-22      VlanA-
1.105           10m ago

```

To view the routes originated from a VRF on specific node, run:

```

cumulus@leaf-1$ netq show ip route leafc-11 vrf DataVrf1080 origin
Route info about prefix * on host leafc-11
Origin Table          IP                  Node
Nexthops              Last Changed
-----  -----
-----  -----
1      DataVrf1080    3.0.0.0/26      leafc-11      VlanA-1-100-
v0            10m ago
1      DataVrf1080    3.0.0.0/32      leafc-11      VlanA-1-100-
v0            10m ago
1      DataVrf1080    3.0.0.1/32      leafc-11      VlanA-1-100-
v0            10m ago
1      DataVrf1080    3.0.0.2/32      leafc-11      VlanA-
1.100           10m ago
1      DataVrf1080    3.0.0.63/32     leafc-11      VlanA-
1.100           10m ago
1      DataVrf1080    30.0.0.15/32     leafc-11
DataVrf1080          10m ago

```



The [VRR \(see page 337\)](#) IPv6 link local address is not displayed when you run `netq show ipv6 address`. `netq` only displays the SVI link local address.

```

cumulus@switch:~$ netq show ipv6 address qct-ly8-05 | egrep
"VlanA-1-101|VlanA-1.101"
2001:fee1:0:1::1/64 switch VlanA-1-101.VRR DataVrf1081 9h ago
2001:fee1:0:1::2/64 switch VlanA-1.101 DataVrf1081 9h ago
fe80::848:b1ff:fe22:7084/64 switch VlanA-1.101 DataVrf1081 9h
ago

```

Using netq view

The `netq view` command provides information about a specific node in the network. The available options are:

```
cumulus@leaf-1:~$ netq view leaf-2
config      : add help text
counters    : add help text
ipv4        : add help text
keys         : add help text
lldp        : LLDP based neighbor info
meminfo     : add help text
top          : add help text
uptime       : add help text
```

For example, to see the `top` output on leaf-2, run:

```
cumulus@leaf-1:~$ netq view leaf-2 top

Output retrieved from 48s ago
top - 21:57:08 up 13 min, 0 users, load average: 0.04, 0.07, 0.03
Tasks: 89 total, 1 running, 88 sleeping, 0 stopped, 0 zombie
%Cpu(s): 2.0 us, 1.6 sy, 0.0 ni, 96.3 id, 0.0 wa, 0.0 hi, 0.0
si, 0.0 st
KiB Mem: 1981988 total, 317784 used, 1664204 free, 4124
buffers
KiB Swap: 0 total, 0 used, 0 free. 92300
cached Mem
 PID USER      PR  NI    VIRT    RES    SHR S %CPU %MEM      TIME+
COMMAND
 1220 root      20   0 163100 121108  6572 S  0.0  6.1  0:02.88
python
 2808 root      20   0  66620  20972  6956 S  6.5  1.1  0:07.97
netq-agent
 1225 root      20   0 265292 14372  6000 S  0.0  0.7  0:06.67
arp_refresh
 1223 root      20   0  42240  12212  5984 S  0.0  0.6  0:00.22
python
 1051 root      20   0  32604   9384  4260 S  0.0  0.5  0:00.04
pwm
 1049 root      20   0  32628   9368  4236 S  0.0  0.5  0:00.06
smond
 1393 root      20   0 104464   7800  4532 S  0.0  0.4  0:02.02
arp_nhs
 1390 root      20   0 104212   7564  4564 S  0.0  0.4  0:00.67
python
     1 root      20   0 112492   6844  2980 S  0.0  0.3  0:04.17
systemd
```

```

1391 root      20   0    24632   6476   3436 S    0.0   0.3   0:00.11 ts
1837 quagga   15  -5    48540   5940   3180 S    0.0   0.3   0:00.77
bgpd

```

Monitoring the redis Server

You can use the `redis-cli info` command to determine how much memory is consumed by the `redis` server, how many connections there are, and so forth.

Two recommended commands to use are `redis-cli info` and `redis-cli ping`.



To use the `redis CLI`, you need to [install redis-tools](#).

Specifying a Different redis Server

If you need to change the IP address of the `redis` server, run `netq add server` again, specifying the IP address of the new server, then restart the `netq` agent. For example:

```

cumulus@switch:~$ sudo netq add 198.51.100.10
cumulus@switch:~$ sudo netq agent restart

```

Note that you need to specify this for every switch that you're monitoring with `netq`. Cumulus Networks recommends you use an automation tool like Ansible or Puppet to quickly update the server across all switches.

Troubleshooting netq

`netq` agent logs to `/var/log/netq-agent.log`. The logs are logrotated.

To ensure that the `netq` agent is running, run:

```

cumulus@switch:~$ netq agent status
Running...

```

SNMP Monitoring

Cumulus Linux utilizes the open source Net-SNMP agent `snmpd`, v5.7.3, which provides support for most of the common industry-wide MIBs, including interface counters and TCP/UDP IP stack data.



Cumulus Linux does not prevent customers from extending SNMP features. However, Cumulus Networks encourages the use of higher performance monitoring environments, rather than SNMP.



Contents

This chapter covers ...

- Introduction to SNMP (Simple Network Management Protocol) (see page 697)
- Configuring Ports for SNMP to Listen for Requests (see page 697)
- Starting the SNMP Daemon (see page 697)
- Configuring SNMP (see page 698)
 - Setting up the Custom Cumulus Networks MIBs (see page 698)
 - Enabling the .1.3.6.1.2.1 Range (see page 699)
 - Enabling Public Community (see page 700)
 - Configuring SNMPV3 (see page 700)
- snmpwalk a Switch from Another Linux Device (see page 703)
 - Troubleshooting Tips Table for snmpwalks (see page 704)
- SNMP Traps (see page 705)
 - snmptrapd.conf (see page 705)
 - Generating Event Notification Traps (see page 705)
- Supported MIBs (see page 710)

Introduction to SNMP (Simple Network Management Protocol)

SNMP is an IETF standards-based network management architecture and protocol that traces its roots back to Carnegie-Mellon University in 1982. Since then, it's been modified by programmers at the University of California. In 1995, this code was also made publicly available as the UCD project. After that, `ucd-snmp` was extended by work done at the University of Liverpool as well as later in Denmark. In late 2000, the project name changed to `net-snmp` and became a fully-fledged collaborative open source project. The version used by Cumulus Networks is base on the latest `net-snmp` 5.7.3 branch with added custom MIBs and pass through and pass persist scripts.

Configuring Ports for SNMP to Listen for Requests

For security reasons, the default port binding for `snmpd` is the loopback local address; consequently by default, the SNMP service does not listen for SNMP requests from outside the switch. In order to listen to requests from outside the switch, you need to change this binding to a specific IP address (or all interfaces) after configuring security access (community strings, users, and so forth). This is a change from older versions of Cumulus Linux (before version 3.0), which listened to incoming requests on all interfaces by default. The `snmpd` configuration file is `/etc/snmp/snmpd.conf` and should be modified before enabling and starting `snmpd`. The default configuration has no access community strings defined so `snmpd` will not respond to any SNMP requests until this is added.

Starting the SNMP Daemon

The following procedure is the recommended process to start `snmpd` and monitor it using `systemctl`.

To start the SNMP daemon:

1. Start the `snmpd` daemon:



```
cumulus@switch:~$ sudo systemctl start snmpd.service
```

2. Configure the `snmpd` daemon to start automatically after reboot:

```
cumulus@switch:~$ sudo systemctl enable snmpd.service
```

3. To enable `snmpd` to restart automatically after failure:

- a. Create a file called `/etc/systemd/system/snmpd.service.d/restart.conf`.
- b. Add the following lines:

```
[Service]
Restart=always
RestartSec=60
```

- c. Run `sudo systemctl daemon-reload`.

Once the service is started, SNMP can be used to manage various components on the Cumulus Linux switch.

Configuring SNMP

Cumulus Linux ships with a production usable default `snmpd.conf` file included. This section covers a few basic configuration options in `snmpd.conf`. For more information regarding further configuring this file, refer to the `snmpd.conf` man page.



The default `snmpd.conf` file does not include all supported MIBs or OIDs that can be exposed.



Customers must at least change the default community string for v1 or v2c environments or the `snmpd` daemon will not respond to any requests.

Setting up the Custom Cumulus Networks MIBs



No changes are required in the `/etc/snmp/snmpd.conf` file on the switch, in order to support the custom Cumulus Networks MIBs. The following lines are already included by default:

```
view systemonly included .1.3.6.1.4.1.40310.1
view systemonly included .1.3.6.1.4.1.40310.2
sysObjectID 1.3.6.1.4.1.40310
pass_persist .1.3.6.1.4.1.40310.1 /usr/share/snmp/resq_pp.py
pass_persist .1.3.6.1.4.1.40310.2 /usr/share/snmp
/cl_drop_cntrs_pp.py
```



However, several files need to be copied to the server, in order for the custom Cumulus MIB to be recognized on the destination NMS server.

- /usr/share/snmp/mibs/Cumulus-Snmp-MIB.txt
- /usr/share/snmp/mibs/Cumulus-Counters-MIB.txt
- /usr/share/snmp/mibs/Cumulus-Resource-Query-MIB.txt

Enabling the .1.3.6.1.2.1 Range

Some MIBs, including storage information, are not included by default in `snmpd.conf` in Cumulus Linux. This results in some default views on common network tools (like `librenms`) to return less than optimal data. More MIBs can be included, by enabling all the .1.3.6.1.2.1 range. This simplifies the configuration file, removing concern that any required MIBs will be missed by the monitoring system. Various MIBs were added to version 3.0 and include the following: ENTITY and ENTITY-SENSOR MIB and parts of the BRIDGE-MIB and Q-BRIDGE-MIBs. These are included in the default configuration.



The view of the BRIDGE-MIB and Q-BRIDGE-MIB are commented out.



This configuration grants access to a large number of MIBs, including all MIB2 MIBs, which could reveal more data than expected. In addition to being a security vulnerability, it could consume more CPU resources.

To enable the .1.3.6.1.2.1 range:

1. Open `/etc/snmp/snmpd.conf` in a text editor.
2. Make sure the following lines are included in the configuration:

```
#####
#
# ACCESS CONTROL
#
#
# system
view systemonly included .1.3.6.1.2.1
# quagga ospf6
view systemonly included .1.3.6.1.3.102
# lldpd (Note: lldpd must be restarted with the -x option
# configured in order to send info to snmpd via Agent X
view systemonly included .1.0.8802.1.1.2
# Cumulus specific
view systemonly included .1.3.6.1.4.1.40310.1
view systemonly included .1.3.6.1.4.1.40310.2
```



3. Restart `snmpd`:

```
cumulus@switch:~$ sudo systemctl restart snmpd.service
```

Enabling Public Community

The `snmpd` authentication for versions 1 and 2 is disabled by default in Cumulus Linux. This password (called a community string) can be enabled by setting **rocommunity** (for read-only access) or **rwcommunity** (for read-write access). To enable read-only querying by a client:

1. Open `/etc/snmp/snmpd.conf` in a text editor.
2. To allow read-only access using a password *public* from any client IP address (*default*) for the view you defined before with `systemonly`, add the following line to the end of the file, then save it:

```
rocommunity public default -V systemonly
```

Syntax	Meaning
rocommunity	Read-only community; (<i>rwcommunity</i> is for read-write access).
public	Plain text password.
default	"default" allows connections from any system. "localhost" allows requests only from the local host. A restricted source can either be a specific hostname (or address), or a subnet, represented as IP/MASK (like 10.10.10.0/255.255.255.0), or IP/BITS (like 10.10.10.0/24), or the IPv6 equivalents.
systemonly	The name of this particular SNMP view. This is a user-defined value.

3. Restart `snmpd`:

```
cumulus@switch:~$ sudo systemctl restart snmpd.service
```

Configuring SNMPv3

Since community strings in versions 1 and 2c are sent in the clear, SNMPv3 is often used to enable authentication and encryption. SNMPv3 was first released around 2000. A minimal example is shown here for `/etc/snmp/snmpd.conf` that defines three users, each with a different combination of authentication and encryption. Please change these usernames and passwords before using this in a network:



Make sure you change the usernames and passwords in the sample code below, as the ones used here are for explanatory purposes only.



```
# simple no auth user
#createUser user1

# user with MD5 authentication
#createUser user2 MD5 user2password

# user with MD5 for auth and DES for encryption
#createUser user3 MD5 user3password DES user3encryption

# user666 with SHA for authentication and AES for encryption
createUser user666 SHA user666password AES user666encryption

# user999 with MD5 for authentication and DES for encryption
createUser user999 MD5 user999password DES user999encryption

# restrict users to certain OIDs
# (Note: creating rouser or rwuser will give
# access regardless of the createUser command above. However,
# createUser without rouser or rwuser will not provide any access).
rouser user1 noauth 1.3.6.1.2.1.1
rouser user2 auth 1.3.6.1.2.1
rwuser user3 priv 1.3.6.1.2.1
rwuser user666
rwuser user999
```

Once you make this configuration and restart the `snmpd` daemon, the user access can be checked with a client — the Debian package called `snmp` contains `snmpget` and `snmpwalk`, as well as other programs that are useful for checking daemon functionality from the switch itself or from another workstation. The following commands check the access for each user defined above from the localhost (the switch itself):

```
# check user1 which has no authentication or encryption (NoauthNoPriv)
snmpget -v 3 -u user1 -l NoauthNoPriv localhost 1.3.6.1.2.1.1.0
snmpwalk -v 3 -u user1 -l NoauthNoPriv localhost 1.3.6.1.2.1.1

# check user2 which has authentication but no encryption (authNoPriv)
snmpget -v 3 -u user2 -l authNoPriv -a MD5 -A user2password localhost
1.3.6.1.2.1.1.0
snmpget -v 3 -u user2 -l authNoPriv -a MD5 -A user2password localhost
1.3.6.1.2.1.2.1.0
snmpwalk -v 3 -u user2 -l authNoPriv -a MD5 -A user2password
localhost 1.3.6.1.2.1

# check user3 which has both authentication and encryption (authPriv)
snmpget -v 3 -u user3 -l authPriv -a MD5 -A user3password -x DES -X
user3encryption localhost .1.3.6.1.2.1.1.0
snmpwalk -v 3 -u user3 -l authPriv -a MD5 -A user3password -x DES -X
user3encryption localhost .1.3.6.1.2.1
snmpwalk -v 3 -u user666 -l authPriv -a SHA -x AES -A user666password
-X user666encryption localhost 1.3.6.1.2.1.1
```



```
snmpwalk -v 3 -u user999 -l authPriv -a MD5 -x DES -A user999password  
-X user999encryption localhost 1.3.6.1.2.1.1
```

A slightly more secure method of configuring SNMPv3 users without creating cleartext passwords is the following:

1. Install the `net-snmp-config` script that is in `libsnmp-dev` package:

```
cumulus@switch:~$ sudo apt-get update  
cumulus@switch:~$ sudo apt-get install libsnmp-dev
```

2. Stop the daemon:

```
cumulus@switch:~$ sudo systemctl stop snmpd.service
```

3. Use the `net-snmp-config` command to create two users, one with MD5 and DES, and the next with SHA and AES.



The minimum password length is 8 characters and the arguments `-a` and `-x` to `net-snmp-config` have different meanings than they do for `snmpwalk`.

```
cumulus@switch:~$ sudo net-snmp-config --create-snmpv3-user -a  
md5authpass -x desprivpass -A MD5 -X DES userMD5withDES  
cumulus@switch:~$ sudo net-snmp-config --create-snmpv3-user -a  
shaauthpass -x aesprivpass -A SHA -X AES userSHAwithAES  
cumulus@switch:~$ sudo systemctl start snmpd.service
```

This adds a `createUser` command in `/var/lib/snmp/snmpd.conf`. Do **not** edit this file by hand, unless you are removing usernames. It also adds the `rwuser` in `/usr/share/snmp/snmpd.conf`. You may want to edit this file and restrict access to certain parts of the MIB by adding `noauth`, `auth` or `priv` to allow unauthenticated access, require authentication or to enforce use of encryption, respectively.

The `snmpd` daemon reads the information from the `/var/lib/snmp/snmpd.conf` file and then the line is removed (eliminating the storage of the master password for that user) and replaced with the key that is derived from it (using the EngineID). This key is a localized key, so that if it is stolen it cannot be used to access other agents. To remove the two users `userMD5withDES` and `userSHAwithAES`, you need simply stop the `snmpd` daemon and edit the files `/var/lib/snmp/snmpd.conf` and `/usr/share/snmp/snmpd.conf`. Simply remove the lines containing the username. Then restart the `snmpd` daemon as in step 3 above.

From a client, you would access the MIB with the correct credentials. (Again, note that the roles of `-x`, `-a` and `-x` and `-A` are reversed on the client side as compared with the `net-snmp-config` command used above.)



```
snmpwalk -v 3 -u userMD5withDES -l authPriv -a MD5 -x DES -A  
md5authpass -X desprivpass localhost 1.3.6.1.2.1.1.1  
snmpwalk -v 3 -u userSHAwithAES -l authPriv -a SHA -x AES -A  
shaauthpass -X aesprivpass localhost 1.3.6.1.2.1.1.1
```

snmpwalk a Switch from Another Linux Device

One of the most important ways to troubleshoot is to `snmpwalk` the switch from another Linux device that can reach the Cumulus Linux switch. For this demonstration, another switch running Cumulus Linux within the network is used.

1. Open `/etc/apt/sources.list` in an editor.
2. Add the following line, and save the file:

```
deb http://ftp.us.debian.org/debian/ jessie main non-free
```

3. Update the switch:

```
cumulus@switch:~$ sudo apt-get update
```

4. Install the `snmp` and `snmp-mibs-downloader` packages:

```
cumulus@switch:~$ sudo apt-get install snmp snmp-mibs-downloader
```

5. Verify that the "mibs :" line is commented out in `/etc/snmp/snmp.conf`:

```
#  
# As the snmp packages come without MIB files due to license  
# reasons, loading  
# of MIBs is disabled by default. If you added the MIBs you can  
# reenable  
# loading them by commenting out the following line.  
#mibs :
```

6. Perform an `snmpwalk` on the switch. The switch running `snmpd` in the demonstration is using IP address 192.168.0.111. It is possible to `snmpwalk` the switch from itself. Run the following command, which rules out an SNMP problem against a networking problem.

```
cumulus@switch:~$ snmpwalk -c public -v2c 192.168.0.111
```

Here is some sample output:

```

IF-MIB::ifPhysAddress.2 = STRING: 74:e6:e2:f5:a2:80
IF-MIB::ifPhysAddress.3 = STRING: 0:e0:ec:25:b8:54
IF-MIB::ifPhysAddress.4 = STRING: 74:e6:e2:f5:a2:81
IF-MIB::ifPhysAddress.5 = STRING: 74:e6:e2:f5:a2:82
IF-MIB::ifPhysAddress.6 = STRING: 74:e6:e2:f5:a2:83
IF-MIB::ifPhysAddress.7 = STRING: 74:e6:e2:f5:a2:84
IF-MIB::ifPhysAddress.8 = STRING: 74:e6:e2:f5:a2:85
IF-MIB::ifPhysAddress.9 = STRING: 74:e6:e2:f5:a2:86
IF-MIB::ifPhysAddress.10 = STRING: 74:e6:e2:f5:a2:87
IF-MIB::ifPhysAddress.11 = STRING: 74:e6:e2:f5:a2:88
IF-MIB::ifPhysAddress.12 = STRING: 74:e6:e2:f5:a2:89
IF-MIB::ifPhysAddress.13 = STRING: 74:e6:e2:f5:a2:8a
IF-MIB::ifPhysAddress.14 = STRING: 74:e6:e2:f5:a2:8b
IF-MIB::ifPhysAddress.15 = STRING: 74:e6:e2:f5:a2:8c
IF-MIB::ifPhysAddress.16 = STRING: 74:e6:e2:f5:a2:8d
IF-MIB::ifPhysAddress.17 = STRING: 74:e6:e2:f5:a2:8e
IF-MIB::ifPhysAddress.18 = STRING: 74:e6:e2:f5:a2:8f
IF-MIB::ifPhysAddress.19 = STRING: 74:e6:e2:f5:a2:90

```

Any information gathered here should verify that `snmpd` is running correctly on the Cumulus Linux side, reducing locations where a problem may reside.

Troubleshooting Tips Table for snmpwalks

Run <code>snmpwalk</code> from	If it works	If it does not work
switch (switch to monitor)	<code>snmpd</code> is serving information correctly. The problem resides somewhere else. For example, network connectivity, or Prism misconfiguration.	Is <code>snmpd</code> misconfigured or installed incorrectly?
switch2 (another Cumulus Linux switch in the network)	<code>snmpd</code> is serving information correctly and network reachability works between switch and switch2 . The problem resides somewhere else. For example, Prism cannot reach switch , or there is a Prism misconfiguration.	Network connectivity is not able to grab information? Is there an <code>iptables</code> rule blocking? Is the <code>snmpwalk</code> being run correctly?
Nutanix Prism CLI (see page 713) (SSH to the cluster IP address)	<code>snmpd</code> is serving information correctly and network reachability works between switch and the Nutanix Appliance . The problem resides somewhere else. For example, the GUI might be misconfigured.	Is the right community name being used in the GUI? Is <code>snmp v2c</code> being used?



SNMP Traps

snmptrapd.conf

The Net-SNMP trap daemon configuration file, `/etc/snmp/snmptrapd.conf`, is used to configure how incoming traps should be processed. For more information about specific configuration options within the file, run the following command:

```
cumulus@switch:~$ man 5 snmptrapd.conf

#####
#####

#
# EXAMPLE-trap.conf:
#   An example configuration file for configuring the Net-SNMP
snmptrapd agent.
#
#####
#####

#
# This file is intended to only be an example. If, however, you want
# to use it, it should be placed in /etc/snmp/snmptrapd.conf.
# When the snmptrapd agent starts up, this is where it will look for
it.
#
# All lines beginning with a '#' are comments and are intended for you
# to read. All other lines are configuration commands for the agent.

#
# PLEASE: read the snmptrapd.conf(5) manual page as well!
#
snmpTrapdAddr localhost
forward default {{global['snmp_server']}}}
```

Generating Event Notification Traps

The Net-SNMP agent provides a method to generate SNMP trap events, via the Distributed Management (DisMan) Event MIB, for various system events, including linkup/down, exceeding the temperature sensor threshold, CPU load, or memory threshold, or other SNMP MIBs.

Monitoring Fans, Power Supplies, or Transformers

SNMP can be configured to monitor the operational status of an Entity MIB or Entity-Sensor MIB. The operational status, given as a value of ok(1), unavailable(2), or nonoperational(3), can be determined by adding the following example configuration to `/etc/snmp/snmpd.conf`, and adjusting the values:

- Using the `entPhySensorOperStatus` integer:

```
# without installing extra MIBS we can check the check Fan1
status
# if the Fan1 index is 100011001
monitor -I -r 10 -o 1.3.6.1.2.1.47.1.1.1.7.100011001 "Fan1
Not OK" 1.3.6.1.2.1.99.1.1.1.5.100011001 > 1
# Any Entity Status non OK (greater than 1)
monitor -r 10 -o 1.3.6.1.2.1.47.1.1.1.7 "Sensor Status
Failure" 1.3.6.1.2.1.99.1.1.1.5 > 1
```



The `entPhySensorOperStatus` integer can be found by walking the `entPhysicalName` table.

To get all sensor information, run `snmpwalk` on the `entPhysicalName` table. For example:

```
cumulus@leaf01:~$ snmpwalk -v 2c -cpublic localhost .
1.3.6.1.2.1.47.1.1.1.1.7
iso.3.6.1.2.1.47.1.1.1.1.7.100000001 = STRING: "PSU1Temp1"
iso.3.6.1.2.1.47.1.1.1.1.7.100000002 = STRING: "PSU2Temp1"
iso.3.6.1.2.1.47.1.1.1.1.7.100000003 = STRING: "Temp1"
iso.3.6.1.2.1.47.1.1.1.1.7.100000004 = STRING: "Temp2"
iso.3.6.1.2.1.47.1.1.1.1.7.100000005 = STRING: "Temp3"
iso.3.6.1.2.1.47.1.1.1.1.7.100000006 = STRING: "Temp4"
iso.3.6.1.2.1.47.1.1.1.1.7.100000007 = STRING: "Temp5"
iso.3.6.1.2.1.47.1.1.1.1.7.100011001 = STRING: "Fan1"
iso.3.6.1.2.1.47.1.1.1.1.7.100011002 = STRING: "Fan2"
iso.3.6.1.2.1.47.1.1.1.1.7.100011003 = STRING: "Fan3"
iso.3.6.1.2.1.47.1.1.1.1.7.100011004 = STRING: "Fan4"
iso.3.6.1.2.1.47.1.1.1.1.7.100011005 = STRING: "Fan5"
iso.3.6.1.2.1.47.1.1.1.1.7.100011006 = STRING: "Fan6"
iso.3.6.1.2.1.47.1.1.1.1.7.100011007 = STRING: "PSU1Fan1"
iso.3.6.1.2.1.47.1.1.1.1.7.100011008 = STRING: "PSU2Fan1"
iso.3.6.1.2.1.47.1.1.1.1.7.110000001 = STRING: "PSU1"
iso.3.6.1.2.1.47.1.1.1.1.7.110000002 = STRING: "PSU2"
```

- Using the OID name:

```
# for a specific fan called Fan1 with an index 100011001
monitor -I -r 10 -o entPhysicalName.100011001 "Fan1 Not OK"
entPhySensorOperStatus.100011001 > 1
# for any Entity Status not OK ( greater than 1)
```



```
monitor -r 10 -o entPhysicalName "Sensor Status Failure"
entPhySensorOperStatus > 1
```



The OID name can be used if the `snmp-mibs-downloader` package is installed.

Enabling MIB to OID Translation

MIB names can be used instead of OIDs, by installing the `snmp-mibs-downloader`, to download SNMP MIBs to the switch prior to enabling traps. This greatly improves the readability of the `snmpd.conf` file.

1. Open `/etc/apt/sources.list` in a text editor.
2. Add the `non-free` repository, and save the file:

```
cumulus@switch:~$ sudo deb http://ftp.us.debian.org/debian/
jessie main non-free
```

3. Update the switch:

```
cumulus@switch:~$ sudo apt-get update
```

4. Install the `snmp-mibs-downloader`:

```
cumulus@switch:~$ sudo apt-get install snmp-mibs-downloader
```

5. Open the `/etc/snmp/snmp.conf` file to verify that the `mibs :` line is commented out:

```
# As the snmp packages come without MIB files due to license
# reasons, loading
# of MIBs is disabled by default. If you added the MIBs you can
# reenable
# loading them by commenting out the following line.
#mibs :
```

6. Open the `/etc/default/snmpd` file to verify that the `export MIBS=` line is commented out:

```
# This file controls the activity of snmpd and snmptrapd
# Don't load any MIBs by default.
# You might comment this lines once you have the MIBs Downloaded.
```



```
#export MIBS=
```

- Once the configuration has been confirmed, remove or comment out the `non-free` repository in `/etc/apt/sources.list`.

```
#deb http://ftp.us.debian.org/debian/ jessie main non-free
```

Configuring Trap Events

The following configurations should be made in `/etc/snmp/snmp.conf`, in order to enable specific types of traps. Once configured, restart the `snmpd` service to apply the changes.

```
cumulus@switch:~$ sudo systemctl restart snmpd.service
```

Defining Access Credentials

An SNMPv3 username is required to authorize the DisMan service. The example code below uses `cumulusUser` as the username.

```
createUser cumulusUser
iquerySecName cumulusUser
rouser cumulusUser
```

Defining Trap Receivers

The example code below creates a trap receiver that is capable of receiving SNMPv2 traps.

```
trap2sink 192.168.1.1 public
```



Although the traps are sent to an SNMPV2 receiver, the SNMPv3 user is still required.



It is possible to define multiple trap receivers, and to use the domain name instead of IP address in the `trap2sink` directive.

Configuring LinkUp/Down Notifications

The `linkUpDownNotifications` directive is used to configure linkup/down notifications when the operational status of the link changes.



```
linkUpDownNotifications yes
```



The default frequency for checking link up/down is 60 seconds. The default frequency can be changed using the `monitor` directive directly instead of the `linkUpDownNotifications` directive. See `man snmpd.conf` for details.

Configuring Temperature Notifications

Temperature sensor information for each available sensor is maintained in the the `ImSensors` MIB. Each platform may contain a different number of temperature sensors. The example below generates a trap event when any temperature sensors exceeds a threshold of 68 degrees (centigrade). It monitors each `1mTempSensorsValue`. When the threshold value is checked and exceeds the `1mTempSensorsValue`, a trap is generated. The `-o 1mTempSensorsDevice` option is used to instruct SNMP to also include the `ImTempSensorsDevice` MIB in the generated trap. The default frequency for the `monitor` directive is 600 seconds. The default frequency may be changed using the `-r` option.:

```
monitor lmTemSensor -o lmTempSensorsDevice lmTempSensorsValue > 68000
```

Alternatively, temperature sensors may be monitored individually. To monitor the sensors individually, first use the `sensors` command to determine which sensors are available to be monitored on the platform.

```
cumulus@switch:~$ sudo sensors  
  
CY8C3245-i2c-4-2e  
Adapter: i2c-0-mux (chan_id 2)  
fan5: 7006 RPM (min = 2500 RPM, max = 23000 RPM)  
fan6: 6955 RPM (min = 2500 RPM, max = 23000 RPM)  
fan7: 6799 RPM (min = 2500 RPM, max = 23000 RPM)  
fan8: 6750 RPM (min = 2500 RPM, max = 23000 RPM)  
temp1: +34.0 C (high = +68.0 C)  
temp2: +28.0 C (high = +68.0 C)  
temp3: +33.0 C (high = +68.0 C)  
temp4: +31.0 C (high = +68.0 C)  
temp5: +23.0 C (high = +68.0 C)
```

Configure a `monitor` command for the specific sensor using the `-I` option. The `-I` option indicates that the monitored expression is applied to a single instance. In this example, there are five temperature sensors available. The following monitor directive can be used to monitor only temperature sensor three at five minute intervals.

```
monitor -I -r 300 lmTemSensor3 -o lmTempSensorsDevice.3  
lmTempSensorsValue.3 > 68000
```



Configuring Free Memory Notifications

You can monitor free memory using the following directives. The example below generates a trap when free memory drops below 1,000,000KB. The free memory trap also includes the amount of total real memory:

```
monitor MemFreeTotal -o memTotalReal memTotalFree < 1000000
```

Configuring Processor Load Notifications

To monitor CPU load for 1, 5 or 15 minute intervals, use the `load` directive in conjunction with the `monitor` directive. The following example will generate a trap when the 1 minute interval reaches 12%, the 5 minute interval reaches 10% or the 15 minute interval reaches 5%.

```
load 12 10 5
monitor -r 60 -o laNames -o laErrMsg "laTable" laErrorFlag !=0
```

Configuring Disk Utilization Notifications

To monitor disk utilization for all disks, use the `includeAllDisks` directive in conjunction with the `monitor` directive. The example code below generates a trap when a disk is 99% full:

```
includeAllDisks 1%
monitor -r 60 -o dskPath -o DiskErrMsg "dskTable" diskErrorFlag !=0
```

Configuring Authentication Notifications

To generate authentication failure traps, use the `authtrapenable` directive:

```
authtrapenable 1
```

Supported MIBs

Below are the MIBs supported by Cumulus Linux, as well as suggested uses for them. The overall Cumulus Linux MIB is defined in `/usr/share/snmp/mibs/Cumulus-Snmp-MIB.txt`.

MIB Name	Suggested Uses
BRIDGE and Q-BRIDGE	The dot1dBasePortEntry and dot1dBasePortIfIndex tables in the BRIDGE-MIB and dot1qBase, dot1qFdbEntry, dot1qTpFdbEntry, dot1qTpFdbStatus, and the dot1qVlanStaticName tables in the Q-BRIDGE-MIB tables. You must uncomment the <code>bridge_pp.py pass_persist</code> script in <code>/etc/snmp/snmpd.conf</code> .



MIB Name	Suggested Uses
BGP4	<p>Implementation of the BGP4 MIB (RFC 4273) which includes bgpPeerTable and bgp4PathAttrTable. By default, the <code>bgp4_pp.py</code> script does not show the bgp4PathAttrTable since this can be very large and result in a periodic CPU spike. To enable this table, add <code>--include-paths</code> to the end of the line (as an option) after uncommenting it in the <code>/etc/snmp/snmpd.conf</code> configuration file.</p> <pre>#pass_persist .1.3.6.1.2.1.15 /usr/share/snmp/bgp4_pp.py</pre>
CUMULUS-COUNTERS-MIB	<p>Discard counters: Cumulus Linux also includes its own counters MIB, defined in <code>/usr/share/snmp/mibs/Cumulus-Counters-MIB.txt</code>. It has the OID <code>.1.3.6.1.4.1.40310.2</code></p>
CUMULUS-RESOURCE-QUERY-MIB	<p>Cumulus Linux includes its own resource utilization MIB, which is similar to using <code>cl-resource-query</code>. It monitors L3 entries by host, route, nexthops, ECMP groups and L2 MAC/BDPU entries. The MIB is defined in <code>/usr/share/snmp/mibs/Cumulus-Resource-Query-MIB.txt</code>, and has the OID <code>.1.3.6.1.4.1.40310.1</code>.</p>
CUMULUS-POE-MIB	<p>The Cumulus Networks custom Power over Ethernet (see page 174) PoE MIB defined in <code>/usr/share/snmp/mibs/Cumulus-POE-MIB.txt</code>. For devices that provide PoE, this provides users with the system wide power information in <code>poeSystemValues</code> as well as per interface <code>PoeObjectsEntry</code> values for the <code>poeObjectsTable</code>. Most of this information comes from the <code>poectl</code> command. This MIB is enabled by uncommenting the following line in <code>/etc/snmp/snmpd.conf</code>:</p> <pre>#pass_persist .1.3.6.1.4.1.40310.3 /usr/share/snmp/cl_poe_pp.py</pre>
DISMAN-EVENT	Trap monitoring
ENTITY	From RFC 4133, the temperature sensors, fan sensors, power sensors, and ports are covered.
ENTITY-SENSOR	Physical sensor information (temperature, fan, and power supply) from RFC 3433.
HOST-RESOURCES	Users, storage, interfaces, process info, run parameters
IEEE8023-LAG-MIB	Implementation of the IEEE 8023-LAG-MIB includes the <code>dot3adAggTable</code> and <code>dot3adAggPortListTable</code> tables. To enable this, edit <code>/etc/snmp/snmpd.conf</code> and uncomment or add the following lines:



MIB Name	Suggested Uses
	<pre>view systemonly included .1.2.840.10006.300.43 pass_persist .1.2.840.10006.300.43 /usr/share/snmp /ieee8023_lag_pp.py</pre>
IF-MIB	Interface description, type, MTU, speed, MAC, admin, operation status, counters
IP (includes ICMP)	IPv4, IPv4 addresses, counters, netmasks
IPv6	IPv6 counters
IP-FORWARD	IP routing table
LLDP	L2 neighbor info from <code>lldpd</code> (note, you need to enable the SNMP subagent (see page 257) in LLDP). <code>lldpd</code> needs to be started with the <code>-x</code> option to enable connectivity to <code>snmpd</code> (AgentX).
LM-SENSORS MIB	Fan speed, temperature sensor values, voltages. This is deprecated since the ENTITY-SENSOR MIB has been added.
NET-SNMP-AGENT	Agent timers, user, group config
NET-SNMP-EXTEND	Agent timers, user, group config
NET-SNMP-EXTEND-MIB	(See also this knowledge base article on extending NET-SNMP in Cumulus Linux to include data from power supplies, fans and temperature sensors.)
NET-SNMP-VACM	Agent timers, user, group config
NOTIFICATION-LOG	Local logging
SNMP-FRAMEWORK	Users, access
SNMP-MPD	Users, access
SNMP-TARGET	
	Users, access



MIB Name	Suggested Uses
SNMP-USER-BASED-SM	
SNMP-VIEW-BASED-ACM	Users, access
SNMPv2	SNMP counters (For information on exposing CPU and memory information via SNMP, see this knowledge base article .)
TCP	TCP related information
UCD-SNMP	System memory, load, CPU, disk IO
UDP	UDP related information



Due to licensing restrictions, SNMP support within Quagga has been disabled and not included in Cumulus Linux.

However, Cumulus Linux does support the BGP4 MIB (RFC 4273), which includes `bgpPeerTable` and `bgp4PathAttrTable`. By default, the `pass persist` script `bgp4_pp.py` does not show the `bgp4PathAttrTable` since it can be very large and result in a periodic CPU spike.

To enable this table, [see above](#) (see page 711).



The ENTITY MIB does not currently show the chassis information in Cumulus Linux.

Using Nutanix Prism as a Monitoring Tool

Nutanix Prism is a graphical user interface (GUI) for managing infrastructure and virtual environments. In order to use it, you need to take special steps within Cumulus Linux before you can configure Prism.

Contents

This chapter covers ...

- Configuring Cumulus Linux (see page 714)
- Configuring Nutanix (see page 715)
- Switch Information Displayed on Nutanix Prism (see page 718)
- Troubleshooting a Nutanix Node (see page 719)
- Enabling LLDP/CDP on VMware ESXi (Hypervisor on Nutanix) (see page 719)
 - Enabling LLDP/CDP on Nutanix Acropolis (Hypervisor on Nutanix Acropolis) (see page 721)
- Troubleshooting Connections without LLDP or CDP (see page 721)



Configuring Cumulus Linux

1. SSH to the Cumulus Linux switch that needs to be configured, replacing [switch] below as appropriate:

```
cumulus@switch:~$ ssh cumulus@[switch]
```

2. Confirm the switch is running Cumulus Linux 2.5.5 or newer:

```
cumulus@switch:~$ net show system
Cumulus 3.2.1~1484951197.337c36a
Build: Cumulus Linux 3.2.1~1484951197.337c36a
Uptime: 4 days, 1:30:04
```

3. Open the /etc/snmp/snmpd.conf file in an editor.
4. Uncomment the following 3 lines in the /etc/snmp/snmpd.conf file, and save the file:
 - bridge_pp.py

```
pass_persist .1.3.6.1.2.1.17 /usr/share/snmp/bridge_pp.py
```

- Community

```
rocommunity public default -V systemonly
```

- Line directly below the Q-BRIDGE-MIB (.1.3.6.1.2.1.17)

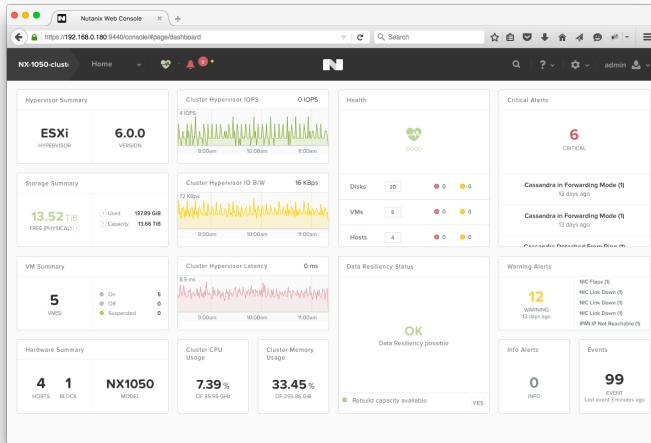
```
# BRIDGE-MIB and Q-BRIDGE-MIB tables
view systemonly included .1.3.6.1.2.1.17
```

5. Restart snmpd:

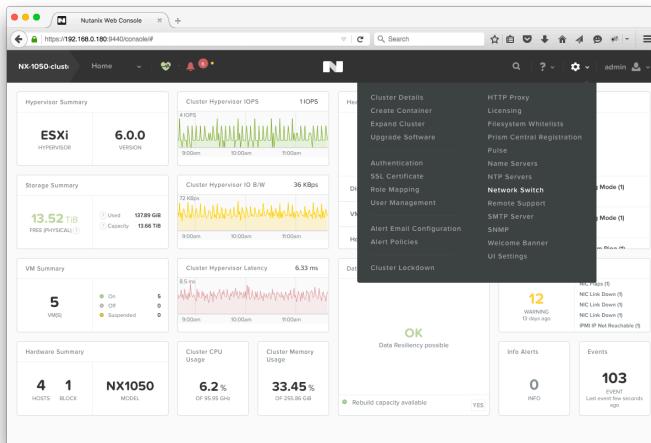
```
cumulus@switch:~$ sudo systemctl restart snmpd.service
Restarting network management services: snmpd.
```

Configuring Nutanix

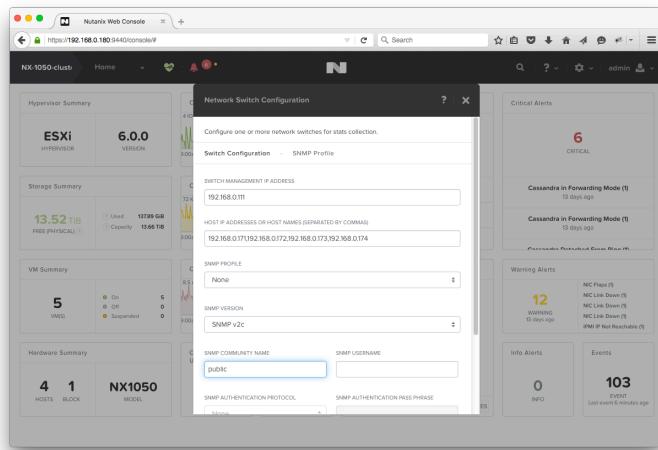
1. Log into the Nutanix Prism. Nutanix defaults to the Home menu, referred to as the Dashboard:



2. Click on the gear icon  in the top right corner of the dashboard, and select NetworkSwitch:



3. Click the **+Add Switch Configuration** button in the **Network Switch Configuration** pop up window.
4. Fill out the **Network Switch Configuration** for the Top of Rack (ToR) switch configured for snmpd in the previous section:



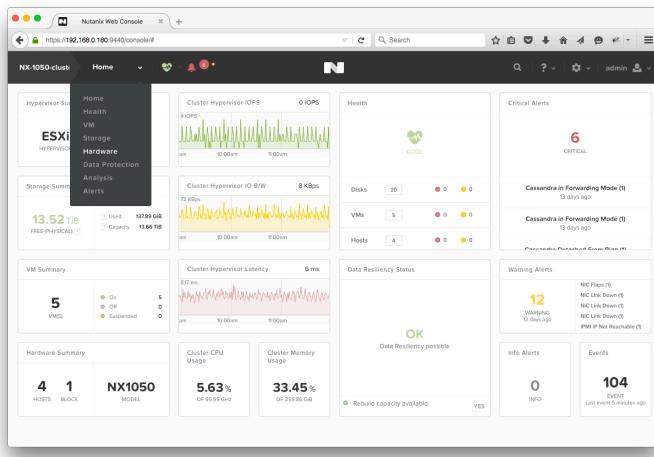
Configuration Parameter	Description	Value Used in Example
Switch Management IP Address	This can be any IP address on the box. In the screenshot above, the eth0 management IP is used.	192.168.0.111
Host IP Addresses or Host Names	IP addresses of Nutanix hosts connected to that particular ToR switch.	192.168.0.171,192.168.0.172,192.168.0.173,192.168.0.174
SNMP Profile	Saved profiles, for easy configuration when hooking up to multiple switches.	None
SNMP Version	SNMP v2c or SNMP v3. Cumulus Linux has only been tested with SNMP v2c for Nutanix integration.	SNMP v2c
SNMP Community Name		public

Configuration Parameter	Description	Value Used in Example
	SNMP v2c uses communities to share MIBs. The default community for snmpd is 'public'.	



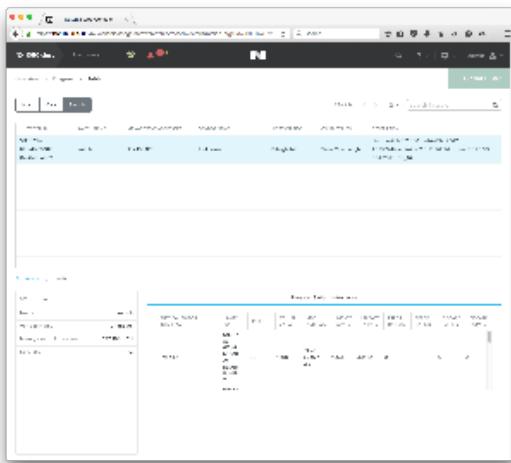
The rest of the values were not touched for this demonstration. They are usually used with SNMP v3.

5. Save the configuration. The switch will now be present in the **Network Switch Configuration** menu now.
6. Close the pop up window to return to the dashboard.
7. Open the **Hardware** option from the **Home** dropdown menu:



8. Click the **Table** button.

9. Click the **Switch** button. Configured switches are shown in the table, as indicated in the screenshot below, and can be selected in order to view interface statistics:



The switch has been added correctly, when interfaces hooked up to the Nutanix hosts are visible.

Switch Information Displayed on Nutanix Prism

- Physical Interface (e.g. swp1, swp2). This will only display swp interfaces connected to Nutanix hosts by default.
- Switch ID - Unique identifier that Nutanix keeps track of each port ID (see below)
- Index - interface index, in the above demonstration swp49 maps to Index 52 because there is a loopback and two ethernet interface before the swp starts.
- MTU of interface
- MAC Address of Interface
- Unicast RX Packets (Received)
- Unicast TX Packets (Transmitted)
- Error RX Packets (Received)
- Error TX Packets (Transmitted)
- Discard RX Packets (Received)
- Discard TX Packets (Transmitted)

The Nutanix appliance will use Switch IDs that can also be viewed on the Prism CLI (by SSHing to the box). To view information from the Nutanix CLI, login using the default username **nutanix**, and the password **nutanix/4u**.

```
nutanix@NTNX-14SM15270093-D-CVM:192.168.0.184:~$ ncli network list-switch
      Switch ID          : 00051a76-f711-89b6-0000-00000003bac:::
5f13678e-6ffd-4b33-912f-f1aa6e8da982
      Name          : switch
```

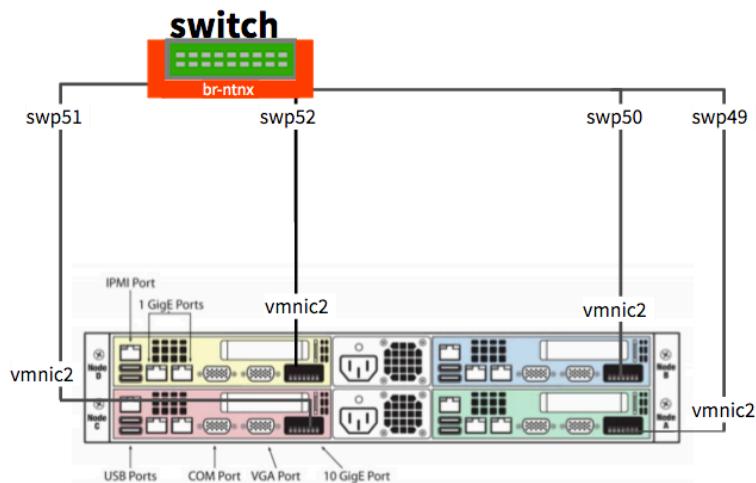
```

Switch Management Address : 192.168.0.111
Description : Linux switch 3.2.65-1+deb7u2+cl2.5+2 #
3.2.65-1+deb7u2+cl2.5+2 SMP Mon Jun 1 18:26:59 PDT 2015 x86_64
Object ID : enterprises.40310
Contact Information : Admin <admin@company.com>
Location Information : Raleigh, NC
Services : 72
Switch Vendor Name : Unknown
Port IDs : 00051a76-f711-89b6-0000-000000003bac::5f13678e-6ffd-4b33-912f-f1aa6e8da982:52, 00051a76-f711-89b6-0000-000000003bac::5f13678e-6ffd-4b33-912f-f1aa6e8da982:53, 00051a76-f711-89b6-0000-000000003bac::5f13678e-6ffd-4b33-912f-f1aa6e8da982:54, 00051a76-f711-89b6-0000-000000003bac::5f13678e-6ffd-4b33-912f-f1aa6e8da982:55

```

Troubleshooting a Nutanix Node

To help visualize the following diagram is provided:



Nutanix Node	Physical Port	Cumulus Linux Port
Node A (Green)	vmnic2	swp49
Node B (Blue)	vmnic2	swp50
Node C (Red)	vmnic2	swp51
Node D (Yellow)	vmnic2	swp52

Enabling LLDP/CDP on VMware ESXi (Hypervisor on Nutanix)

- Follow the directions on one of the following websites to enable CDP:

- [kb.vmware.com/selfservice/microsites/search.do?
language=en_US&cmd=displayKC&externalId=1003885](http://kb.vmware.com/selfservice/microsites/search.do?language=en_US&cmd=displayKC&externalId=1003885)



- wahlnetwork.com/2012/07/17/utilizing-cdp-and-lldp-with-vsphere-networking/

For example, switch CDP on:

```
root@NX-1050-A:~] esxcli network vswitch standard set -c  
both -v vSwitch0
```

Then confirm it is running:

```
root@NX-1050-A:~] esxcli network vswitch standard list -v  
vSwitch0  
vSwitch0  
    Name: vSwitch0  
    Class: etherswitch  
    Num Ports: 4082  
    Used Ports: 12  
    Configured Ports: 128  
    MTU: 1500  
    CDP Status: both  
    Beacon Enabled: false  
    Beacon Interval: 1  
    Beacon Threshold: 3  
    Beacon Required By:  
    Uplinks: vmnic3, vmnic2, vmnic1, vmnic0  
    Portgroups: VM Network, Management Network
```

The **both** means CDP is now running, and the lldp dameon on Cumulus Linux is capable of 'seeing' CDP devices.

2. After the next CDP interval, the Cumulus Linux box will pick up the interface via the lldp daemon:

```
cumulus@switch:~$ lldpctl show neighbor swp49  
-----  
-----  
LLDP neighbors:  
-----  
  
Interface:      swp49, via: CDPv2, RID: 6, Time: 0 day, 00:34:58  
Chassis:  
    ChassisID:      local NX-1050-A  
    SysName:       NX-1050-A  
    SysDescr:      Releasebuild-2494585 running on VMware ESX  
    MgmtIP:        0.0.0.0  
    Capability:    Bridge, on  
Port:  
    PortID:        ifname vmnic2  
    PortDescr:     vmnic2  
-----  
-----
```



3. Use `net show` to look at lldp information:

```
cumulus@switch:~$ net show lldp

Local Port      Speed     Mode          Remote Port
Remote Host    Summary
-----  -----  -----  -----
-----  -----
eth0           1G       Mgmt        ===  swp6
oob-mgmt-switch IP: 192.168.0.11/24(DHCP)
swp1           1G       Access/L2   ===  44:38:39:00:00:03
server01        Untagged: br0
swp51           1G       NotConfigured  ===  swp1
spine01
swp52           1G       NotConfigured  ===  swp1
spine02
```

Enabling LLDP/CDP on Nutanix Acropolis (Hypervisor on Nutanix Acropolis)

Nutanix Acropolis is an alternate hypervisor that Nutanix supports. Acropolis Hypervisor uses the yum packaging system and is capable of installing normal Linux lldp daemons to operating just like Cumulus Linux. LLDP should be enabled for each interface on the host. Refer to <https://community.mellanox.com/docs/DOC-1522> for setup instructions.

Troubleshooting Connections without LLDP or CDP

1. Find the MAC address information in the Prism GUI, located in: **Hardware > Table > Host > Host NICs**
2. Select a MAC address to troubleshoot (e.g. 0c:c4:7a:09:a2:43 represents vmnic0 which is tied to NX-1050-A).
3. List out all the MAC addresses associated to the bridge:

```
cumulus@switch:~$ brctl showmacs br-ntnx
port name mac addr      vlan  is local?  ageing timer
swp9   00:02:00:00:00:06  0     no        66.94
swp52  00:0c:29:3e:32:12  0     no        2.73
swp49  00:0c:29:5a:f4:7f  0     no        2.73
swp51  00:0c:29:6f:e1:e4  0     no        2.73
swp49  00:0c:29:74:0c:ee  0     no        2.73
swp50  00:0c:29:a9:36:91  0     no        2.73
swp9   08:9e:01:f8:8f:0c  0     no        13.56
swp9   08:9e:01:f8:8f:35  0     no        2.73
swp4   0c:c4:7a:09:9e:d4  0     no        24.05
swp1   0c:c4:7a:09:9f:8e  0     no        13.56
swp3   0c:c4:7a:09:9f:93  0     no        13.56
swp2   0c:c4:7a:09:9f:95  0     no        24.05
swp52  0c:c4:7a:09:a0:c1  0     no        2.73
```

swp51	0c:c4:7a:09:a2:35	0	no	2.73
swp49	0c:c4:7a:09:a2:43	0	no	2.73
swp9	44:38:39:00:82:04	0	no	2.73
swp9	74:e6:e2:f5:a2:80	0	no	2.73
swp1	74:e6:e2:f5:a2:81	0	yes	0.00
swp2	74:e6:e2:f5:a2:82	0	yes	0.00
swp3	74:e6:e2:f5:a2:83	0	yes	0.00
swp4	74:e6:e2:f5:a2:84	0	yes	0.00
swp5	74:e6:e2:f5:a2:85	0	yes	0.00
swp6	74:e6:e2:f5:a2:86	0	yes	0.00
swp7	74:e6:e2:f5:a2:87	0	yes	0.00
swp8	74:e6:e2:f5:a2:88	0	yes	0.00
swp9	74:e6:e2:f5:a2:89	0	yes	0.00
swp10	74:e6:e2:f5:a2:8a	0	yes	0.00
swp49	74:e6:e2:f5:a2:b1	0	yes	0.00
swp50	74:e6:e2:f5:a2:b2	0	yes	0.00
swp51	74:e6:e2:f5:a2:b3	0	yes	0.00
swp52	74:e6:e2:f5:a2:b4	0	yes	0.00
swp9	8e:0f:73:1b:f8:24	0	no	2.73
swp9	c8:1f:66:ba:60:cf	0	no	66.94

Alternatively, you can use grep:

```
cumulus@switch:~$ brctl showmacs br-ntnx | grep 0c:c4:7a:09:a2:43
swp49      0c:c4:7a:09:a2:43      0      no          4.58
```

vmnic1 is now hooked up to swp49. This matches what is seen in lldp:

```
cumulus@switch:~$ lldpctl show neighbor swp49
-----
-----
LLDP neighbors:
-----
Interface:      swp49, via: CDPv2, RID: 6, Time: 0 day, 01:11:12
Chassis:
    ChassisID:      local NX-1050-A
    SysName:        NX-1050-A
    SysDescr:       Releasebuild-2494585 running on VMware ESX
    MgmtIP:         0.0.0.0
    Capability:    Bridge, on
Port:
    PortID:        ifname vmnic2
    PortDescr:     vmnic2
-----
```



Monitoring Best Practices

The following monitoring processes are considered best practices for reviewing and troubleshooting potential issues with Cumulus Linux environments. In addition, several of the more common issues have been listed, with potential solutions included.

Contents

This chapter covers ...

- Overview (see page 723)
 - Trend Analysis via Metrics (see page 723)
 - Alerting via Triggered Logging (see page 724)
 - Log Formatting (see page 724)
- Hardware (see page 724)
- System Data (see page 726)
 - CPU Idle Time (see page 726)
 - Disk Usage (see page 728)
- Process Restart (see page 728)
- Layer 1 Protocols and Interfaces (see page 729)
- Layer 2 Protocols (see page 736)
- Layer 3 Protocols (see page 738)
 - BGP (see page 738)
 - OSPF (see page 738)
 - Route and Host Entries (see page 739)
 - Routing Logs (see page 740)
- Logging (see page 740)
- Protocols and Services (see page 742)
 - NTP (see page 742)
- Device Management (see page 742)
 - Device Access Logs (see page 742)
 - Device Super User Command Logs (see page 743)

Overview

This document aims to provide two sets of outputs:

- Metrics that can be polled from Cumulus Linux and used in trend analysis
- Critical log messages that can be monitored for triggered alerts

Trend Analysis via Metrics

A metric is a quantifiable measure that is used to track and assess the status of a specific infrastructure component. It is a check collected over time. Examples of metrics include bytes on an interface, CPU utilization and total number of routes.



Metrics are more valuable when used for trend analysis.

Alerting via Triggered Logging

Triggered issues are normally sent to `syslog`, but could go to another log file depending on the feature. On Cumulus Linux, `rsyslog` handles all logging including local and remote logging. Logs are the best method to use for generating alerts when the system transitions from a stable steady state.

Sending logs to a centralized collector, then creating an alerts based on critical logs is optimal solution for alerting.

Log Formatting

Most log files in Cumulus Linux use a standard presentation format. For example, consider this `syslog` entry:

```
2017-03-08T06:26:43.569681+00:00 leaf01 sysmonitor: Critically high  
CPU use: 99%
```

- 2017-03-08T06:26:43.569681+00:00 is the timestamp.
- *leaf01* is the hostname.
- *sysmonitor* is the process that is the source of the message.
- *Critically high CPU use: 99%* is the message.

For brevity and legibility, the timestamp and hostname have been omitted from the examples in this chapter.

Hardware

The `smonctl` process provides monitoring functionality for various switch hardware elements. Minimum /maximum values are output, depending on the flags applied to the basic command. The hardware elements and applicable commands/flags are listed in the table below:

Hardware Element	Monitoring Command/s	Interval Poll
Temperature	<pre>cumulus@switch:~\$ smonctl -j cumulus@switch:~\$ smonctl -j -s TEMP[X]</pre>	600 seconds
Fan	<pre>cumulus@switch:~\$ smonctl -j cumulus@switch:~\$ smonctl -j -s FAN[X]</pre>	600 seconds
PSU		600 seconds



Hardware Element	Monitoring Command/s	Interval Poll
	<pre>cumulus@switch:~\$ smonctl -j cumulus@switch:~\$ smonctl -j -s PSU[X]</pre>	
PSU Fan	<pre>cumulus@switch:~\$ smonctl -j cumulus@switch:~\$ smonctl -j -s PSU[X]Fan[X]</pre>	600 seconds
PSU Temperature	<pre>cumulus@switch:~\$ smonctl -j cumulus@switch:~\$ smonctl -j -s PSU[X]Temp [X]</pre>	600 seconds
Voltage	<pre>cumulus@switch:~\$ smonctl -j cumulus@switch:~\$ smonctl -j -s Volt[X]</pre>	600 seconds
Front Panel LED	<pre>cumulus@switch:~\$ ledmgrd -d cumulus@switch:~\$ ledmgrd -j</pre>	600 seconds

Hardware Logs	Log Location	Log Entries
High temperature	/var /log /syslog	<pre>/usr/sbin/smond : : Temp1(Board Sensor near CPU): state changed from UNKNOWN to OK /usr/sbin/smond : : Temp2(Board Sensor Near Virtual Switch): state changed from UNKNOWN to OK /usr/sbin/smond : : Temp3(Board Sensor at Front Left Corner): state changed from UNKNOWN to OK</pre>



Hardware Logs	Log Location	Log Entries
		<pre>/usr/sbin/smond : : Temp4(Board Sensor at Front Right Corner): state changed from UNKNOWN to OK /usr/sbin/smond : : Temp5(Board Sensor near Fan): state changed from UNKNOWN to OK</pre>
Fan speed issues	/var /log /syslog	<pre>/usr/sbin/smond : : Fan1(Fan Tray 1, Fan 1): state changed from UNKNOWN to OK /usr/sbin/smond : : Fan2(Fan Tray 1, Fan 2): state changed from UNKNOWN to OK /usr/sbin/smond : : Fan3(Fan Tray 2, Fan 1): state changed from UNKNOWN to OK /usr/sbin/smond : : Fan4(Fan Tray 2, Fan 2): state changed from UNKNOWN to OK /usr/sbin/smond : : Fan5(Fan Tray 3, Fan 1): state changed from UNKNOWN to OK /usr/sbin/smond : : Fan6(Fan Tray 3, Fan 2): state changed from UNKNOWN to OK</pre>
PSU failure	/var /log /syslog	<pre>/usr/sbin/smond : : PSU1Fan1(PSU1 Fan): state changed from UNKNOWN to OK /usr/sbin/smond : : PSU2Fan1(PSU2 Fan): state changed from UNKNOWN to BAD</pre>

System Data

Cumulus Linux includes a number of ways to monitor various aspects of system data. In addition, alerts are issued in high risk situations.

CPU Idle Time

When a CPU reports five high CPU alerts within a span of 5 minutes, an alert is logged.

❗ Short High CPU Bursts

Short bursts of high CPU can occur during switchd churn or routing protocol startup. Do not set alerts for these short bursts.



System Element	Monitoring Command/s	Interval Poll
CPU utilization	cumulus@switch:~\$ cat /proc/stat cumulus@switch:~\$ top -b -n 1	30 seconds

CPU Logs	Log Location	Log Entries
High CPU	/var/log /syslog	sysmonitor: Critically high CPU use: 99% systemd[1]: Starting Monitor system resources (cpu, memory, disk)... systemd[1]: Started Monitor system resources (cpu, memory, disk). sysmonitor: High CPU use: 89% systemd[1]: Starting Monitor system resources (cpu, memory, disk)... systemd[1]: Started Monitor system resources (cpu, memory, disk). sysmonitor: CPU use no longer high: 77%

Cumulus Linux 3.0 and later monitors CPU, memory and disk space via `sysmonitor`. The configurations for the thresholds are stored in `/etc/cumulus/sysmonitor.conf`. More information is available via `man sysmonitor`.

CPU measure	Thresholds
Use	Alert: 90% Crit: 95%
Process Load	Alarm: 95% Crit: 125%

Click here to see differences between Cumulus Linux 2.5 ESR and 3.0 and later...

CPU Logs	Log Location	Log Entries
High CPU		jdoe[2803]: 'localhost' cpu system usage of 41.1% matches resource limit [cpu system usage>30.0%]

CPU Logs	Log Location	Log Entries
	/var /log /syslog	jdoe[4727]: 'localhost' sysloadavg(15min) of 111.0 matches resource limit [sysloadavg(15min)>110.0]

In Cumulus Linux 2.5, CPU logs are created with each unique threshold:

CPU measure	< 2.5 Threshold
User	70%
System	30%
Wait	20%

Cumulus Linux 2.5, CPU and memory warnings are generated via jdoe. The configuration for the thresholds are stored in `/etc/jdoe/jdoorc.d/cl-utilities.rc`.

Disk Usage

When monitoring disk utilization, you can exclude `tmpfs` from monitoring.

System Element	Monitoring Command/s	Interval Poll
Disk utilization	cumulus@switch:~\$ /bin/df -x tmpfs	300 seconds

Process Restart

In Cumulus Linux 3.0 and later, `systemd` is responsible for monitoring and restarting processes.

Process Element	Monitoring Command/s
View processes monitored by systemd	cumulus@switch:~\$ systemctl status

**Process Element****Monitoring Command/s**

Click here to changes from Cumulus Linux 2.5 ESR to 3.0 and later...

Cumulus Linux 2.5.2 through 2.5 ESR uses a forked version of monit called jdoo to monitor processes. If the process ever fails, jdoo then invokes init.d to restart the process.

Process Element	Monitoring Command/s
View processes monitored by jdoo	<pre>cumulus@switch:~\$ jdoo summary</pre>
View process restarts	<pre>cumulus@switch:~\$ sudo cat /var/log/syslog</pre>
View current process state	<pre>cumulus@switch:~\$ ps -aux</pre>

Layer 1 Protocols and Interfaces

Link and port state interface transitions are logged to /var/log/syslog and /var/log/switchd.log.

Interface Element	Monitoring Command/s
Link state	<pre>cumulus@switch:~\$ cat /sys/class/net/[iface]/operstate cumulus@switch:~\$ net show interface all json</pre>
Link speed	<pre>cumulus@switch:~\$ cat /sys/class/net/[iface]/speed cumulus@switch:~\$ net show interface all json</pre>



Interface Element	Monitoring Command/s
Port state	<pre>cumulus@switch:~\$ ip link show cumulus@switch:~\$ net show interface all json</pre>
Bond state	<pre>cumulus@switch:~\$ cat /proc/net/bonding/[bond] cumulus@switch:~\$ net show interface all json</pre>

Interface counters are obtained from either querying the hardware or the Linux kernel. The two outputs should align, but the Linux kernel aggregates the output from the hardware.

Interface Counter Element	Monitoring Command/s	Interval Poll
Interface counters	<pre>cumulus@switch:~\$ cat /sys/class/net/[iface] /statistics/[stat_name] cumulus@switch:~\$ net show counters json cumulus@switch:~\$ cl-netstat -j cumulus@switch:~\$ ethtool -S [iface]</pre>	10 seconds

Layer 1 Logs	Log Location	Log Entries
Link failure/Link flap	/var/log/switchd.log	<pre>switchd[5692]: nic.c:213 nic_set_carrier: swp17: setting kernel carrier: down switchd[5692]: netlink.c:291 libnl: swp1, family 0, ifi 20, oper down switchd[5692]: nic.c:213 nic_set_carrier: swp1: setting kernel carrier: up switchd[5692]: netlink.c:291 libnl: swp17, family 0, ifi 20, oper up</pre>



Layer 1 Logs	Log Location	Log Entries
Unidirectional link	/var /log /swi tchd .log /var /log /ptm .log	<pre>ptmd[7146]: ptm_bfd.c:2471 Created new session 0x1 with peer 10.255.255.11 port swp1 ptmd[7146]: ptm_bfd.c:2471 Created new session 0x2 with peer fe80::4638:39ff:fe00:5b port swp1 ptmd[7146]: ptm_bfd.c:2471 Session 0x1 down to peer 10.255.255.11, Reason 8 ptmd[7146]: ptm_bfd.c:2471 Detect timeout on session 0x1 with peer 10.255.255.11, in state 1</pre>
Bond Negotiation • Working	/var /log /sys log	<pre>kernel: [85412.763193] bonding: bond0 is being created... kernel: [85412.770014] bond0: Enslaving swp2 as a backup interface with an up link kernel: [85412.775216] bond0: Enslaving swp1 as a backup interface with an up link kernel: [85412.797393] IPv6: ADDRCONF (NETDEV_UP): bond0: link is not ready kernel: [85412.799425] IPv6: ADDRCONF (NETDEV_CHANGE): bond0: link becomes ready</pre>
Bond Negotiation • Failing	/var /log /sys log	<pre>kernel: [85412.763193] bonding: bond0 is being created... kernel: [85412.770014] bond0: Enslaving swp2 as a backup interface with an up link kernel: [85412.775216] bond0: Enslaving swp1 as a backup interface with an up link kernel: [85412.797393] IPv6: ADDRCONF (NETDEV_UP): bond0: link is not ready</pre>
MLAG peerlink negotiation • Working	/var /log /sys log	<pre>lldpd[998]: error while receiving frame on swp50: Network is down lldpd[998]: error while receiving frame on swp49: Network is down</pre>

Layer 1 Logs	Log Location	Log Entries
		<pre>kernel: [76174.262893] peerlink: Setting ad_actor_system to 44:38:39:00:00:11 kernel: [76174.264205] 8021q: adding VLAN 0 to HW filter on device peerlink mstpd: one_clag_cmd: setting (1) peer link: peerlink mstpd: one_clag_cmd: setting (1) clag state: up mstpd: one_clag_cmd: setting system-mac 44:39:39:ff:40:94 mstpd: one_clag_cmd: setting clag-role secondary</pre>
	<pre>/var /log /clagd.log</pre>	<pre>clagd[14003]: Cleanup is executing. clagd[14003]: Cannot open file "/tmp/pre-clagd.q7Xio clagd[14003]: Cleanup is finished clagd[14003]: Beginning execution of clagd version 1 clagd[14003]: Invoked with: /usr/sbin/clagd --daemon clagd[14003]: Role is now secondary clagd[14003]: HealthCheck: role via backup is second clagd[14003]: HealthCheck: backup active clagd[14003]: Initial config loaded clagd[14003]: The peer switch is active. clagd[14003]: Initial data sync from peer done. clagd[14003]: Initial handshake done. clagd[14003]: Initial data sync to peer done.</pre>
MLAG peerlink negotiation <ul style="list-style-type: none"> • Failing 	<pre>/var /log /syslog</pre>	<pre>lldpd[998]: error while receiving frame on swp50: Network is down lldpd[998]: error while receiving frame on swp49: Network is down kernel: [76174.262893] peerlink: Setting ad_actor_system to 44:38:39:00:00:11 kernel: [76174.264205] 8021q: adding VLAN 0 to HW filter on device peerlink mstpd: one_clag_cmd: setting (1) peer link: peerlink</pre>



Layer 1 Logs	Log Location	Log Entries
		<pre>mstpd: one_clag_cmd: setting (1) clag state: down mstpd: one_clag_cmd: setting system-mac 44:39:39:ff:40:94 mstpd: one_clag_cmd: setting clag-role secondary</pre>
	/var/log/cla gd.log	<pre>clagd[26916]: Cleanup is executing. clagd[26916]: Cannot open file "/tmp/pre-clagd.6M527vvGX0/brbatch" for reading: No such file or directory clagd[26916]: Cleanup is finished clagd[26916]: Beginning execution of clagd version 1.3.0 clagd[26916]: Invoked with: /usr/sbin/clagd --daemon 169.254.1.2 peerlink.4094 44:38:39:FF:01:01 --priority 1000 --backupIp 10.0.0.2 clagd[26916]: Role is now secondary clagd[26916]: Initial config loaded</pre>
MLAG port negotiation <ul style="list-style-type: none"> • Working 	/var/log/sys log	<pre>kernel: [77419.112195] bonding: server01 is being created... lldpd[998]: error while receiving frame on swp1: Network is down kernel: [77419.122707] 8021q: adding VLAN 0 to HW filter on device swp1 kernel: [77419.126408] server01: Enslaving swp1 as a backup interface with a down link kernel: [77419.177175] server01: Setting ad_actor_system to 44:39:39:ff:40:94 kernel: [77419.190874] server01: Warning: No 802.3ad response from the link partner for any adapters in the bond kernel: [77419.191448] IPv6: ADDRCONF (NETDEV_UP): server01: link is not ready kernel: [77419.191452] 8021q: adding VLAN 0 to HW filter on device server01 kernel: [77419.192060] server01: link status definitely up for interface swp1, 1000 Mbps full duplex</pre>



Layer 1 Logs	Log Location	Log Entries
		<pre>kernel: [77419.192065] server01: now running without any active interface! kernel: [77421.491811] IPv6: ADDRCONF (NETDEV_CHANGE): server01: link becomes ready mstpd: one_clag_cmd: setting (1) mac 44:38:39:00:00:17 <server01, None></pre>
	/var/log/clangd.log	<pre>clagd[14003]: server01 is now dual connected.</pre>
MLAG port negotiation • Failing	/var/log/syslog	<pre>kernel: [79290.290999] bonding: server01 is being created... kernel: [79290.299645] 8021q: adding VLAN 0 to HW filter on device swp1 kernel: [79290.301790] server01: Enslaving swp1 as a backup interface with a down link kernel: [79290.358294] server01: Setting ad_actor_system to 44:39:39:ff:40:94 kernel: [79290.373590] server01: Warning: No 802.3ad response from the link partner for any adapters in the bond kernel: [79290.374024] IPv6: ADDRCONF (NETDEV_UP): server01: link is not ready kernel: [79290.374028] 8021q: adding VLAN 0 to HW filter on device server01 kernel: [79290.375033] server01: link status definitely up for interface swp1, 1000 Mbps full duplex kernel: [79290.375037] server01: now running without any active interface!</pre>
	/var/log/clangd.log	<pre>clagd[14291]: Conflict (server01): matching clag-id (1) not configured on peer ...</pre>



Layer 1 Logs	Log Location	Log Entries
	gd.log	clagd[14291]: Conflict cleared (server01): matching clag-id (1) detected on peer
MLAG port negotiation • Flapping	/var/log/syslog	mstpd: one_clag_cmd: setting (0) mac 00:00:00:00:00:00 <server01, None> mstpd: one_clag_cmd: setting (1) mac 44:38:39:00:00:03 <server01, None>
	/var/log/clagd.log	clagd[14291]: server01 is no longer dual connected clagd[14291]: server01 is now dual connected.

Prescriptive Topology Manager (PTM) uses LLDP information to compare against a topology.dot file that describes the network. It has built in alerting capabilities, so it is preferable to use PTM on box rather than polling LLDP information regularly. The PTM code is available on the Cumulus Networks [GitHub repository](#). Additional PTM, BFD and associated logs are documented in the code.



Peering information should be tracked through PTM. For more information, refer to the [Prescriptive Topology Manager documentation](#).

Neighbor Element	Monitoring Command/s	Interval Poll
LLDP Neighbor	cumulus@switch:~\$ lldpctl -f json	300 seconds
Prescriptive Topology Manager	cumulus@switch:~\$ ptmctl -j [-d]	Triggered



Layer 2 Protocols

Spanning tree is a protocol that prevents loops in a layer 2 infrastructure. In a stable state, the spanning tree protocol should stably converge. Monitoring the Topology Change Notifications (TCN) in STP helps identify when new BPDUs were received.

Interface Counter Element	Monitoring Command/s	Interval Poll
STP TCN Transitions	<pre>cumulus@switch:~\$ mstpcctl showbridge json cumulus@switch:~\$ mstpcctl showport json</pre>	60 seconds
MLAG peer state	<pre>cumulus@switch:~\$ clagctl status cumulus@switch:~\$ clagd -j cumulus@switch:~\$ cat /var/log/clagd.log</pre>	60 seconds
MLAG peer MACs	<pre>cumulus@switch:~\$ clagctl dumppeermacs cumulus@switch:~\$ clagctl dumpourmacs</pre>	300 seconds

Layer 2 Logs	Log Location	Log Entries
Spanning Tree Working	/var /log /syslog	<pre>kernel: [1653877.190724] device swp1 entered promiscuous mode kernel: [1653877.190796] device swp2 entered promiscuous mode mstpd: create_br: Add bridge bridge mstpd: clag_set_sys_mac_br: set bridge mac 00:00:00:00:00:00 mstpd: create_if: Add iface swp1 as port#2 to bridge bridge mstpd: set_if_up: Port swp1 : up</pre>



Layer 2 Logs	Log Location	Log Entries
		<pre>mstpd: create_if: Add iface swp2 as port#1 to bridge bridge mstpd: set_if_up: Port swp2 : up mstpd: set_br_up: Set bridge bridge up mstpd: MSTP_OUT_set_state: bridge:swp1:0 entering blocking state(Disabled) mstpd: MSTP_OUT_set_state: bridge:swp2:0 entering blocking state(Disabled) mstpd: MSTP_OUT_flush_all_fids: bridge:swp1:0 Flushing forwarding database mstpd: MSTP_OUT_flush_all_fids: bridge:swp2:0 Flushing forwarding database mstpd: MSTP_OUT_set_state: bridge:swp1:0 entering learning state(Designated) mstpd: MSTP_OUT_set_state: bridge:swp2:0 entering learning state(Designated) sudo: pam_unix(sudo:session): session closed for user root mstpd: MSTP_OUT_set_state: bridge:swp1:0 entering forwarding state(Designated) mstpd: MSTP_OUT_set_state: bridge:swp2:0 entering forwarding state(Designated) mstpd: MSTP_OUT_flush_all_fids: bridge:swp2:0 Flushing forwarding database mstpd: MSTP_OUT_flush_all_fids: bridge:swp1:0 Flushing forwarding database</pre>
Spanning Tree Blocking	/var /log /syslog	<pre>mstpd: MSTP_OUT_set_state: bridge:swp2:0 entering blocking state(Designated) mstpd: MSTP_OUT_set_state: bridge:swp2:0 entering learning state(Designated) mstpd: MSTP_OUT_set_state: bridge:swp2:0 entering forwarding state(Designated) mstpd: MSTP_OUT_flush_all_fids: bridge:swp2:0 Flushing forwarding database mstpd: MSTP_OUT_flush_all_fids: bridge:swp2:0 Flushing forwarding database mstpd: MSTP_OUT_set_state: bridge:swp2:0 entering blocking state(Alternate) mstpd: MSTP_OUT_flush_all_fids: bridge:swp2:0 Flushing forwarding database</pre>



Layer 3 Protocols

When Quagga boots up for the first time, there will be a different log file for each daemon that has been activated. If the log file is ever edited (for example, through `vtysh` or `Quagga.conf`), the integrated configuration sends all logs to the same file.

In order to send Quagga logs to syslog, apply the configuration `log syslog` in `vtysh`.

BGP

When monitoring BGP, check if BGP peers are operational. There is not much value in alerting on the current operational state of the peer as monitoring the transition is more valuable, and this is done by monitoring syslog.

Monitoring the routing table provides trending on the size of the infrastructure. This is especially useful when integrated with host based solutions (ie. RoH) when the routes track with the number of applications available.

BGP Element	Monitoring Command/s	Interval Poll
BGP peer failure	<pre>cumulus@switch:~\$ vtysh -c "show ip bgp summary json" cumulus@switch:~\$ cl-bgp summary show json</pre>	60 seconds
BGP route table	<pre>cumulus@switch:~\$ vtysh -c "show ip bgp json" cumulus@switch:~\$ cl-bgp route show</pre>	600 seconds

BGP Logs	Log Location	Log Entries
BGP peer down	/var/log /syslog /var/log /quagga/*.log	<pre>bgpd[3000]: %NOTIFICATION: sent to neighbor swp1 4/0 (Hold Timer Expired) 0 bytes bgpd[3000]: %ADJCHANGE: neighbor swp1 Down BGP Notification send</pre>



OSPF

When monitoring OSPF, check if OSPF peers are operational. There is not much value in alerting on the current operational state of the peer as monitoring the transition is more valuable, and this is done by monitoring syslog.

Monitoring the routing table provides trending on the size of the infrastructure. This is especially useful when integrated with host-based solutions (such as Routing on the Host) when the routes track with the number of applications available.

OSPF Element	Monitoring Command(s)	Interval Poll
OSPF protocol peer failure	<pre>cumulus@switch:~\$ vtysh -c "show ip ospf neighbor all json" cumulus@switch:~\$ cl-ospf summary show json</pre>	60 seconds
OSPF link state database	<pre>cumulus@switch:~\$ vtysh - c "show ip ospf database"</pre>	600 seconds

Route and Host Entries

OSPF Element	Monitoring Command(s)	Interval Poll
Host Entries	<pre>cumulus@switch:~\$ cl-resource-query cumulus@switch:~\$ cl-resource-query -k</pre>	600 seconds
Route Entries	<pre>cumulus@switch:~\$ cl-resource-query cumulus@switch:~\$ cl-resource-query -k</pre>	600 seconds



Routing Logs

Layer 3 Logs	Log Location	Log Entries
Routing protocol process crash	/var /log /sys log	<pre>quagga[1824]: Starting Quagga daemons (prio: 10).. zebra. bgpd. bgpd[1847]: BGPd 1.0.0+cl3u7 starting: vty@2605, bgp@<all>:179 zebra[1840]: client 12 says hello and bids fair to announce only bgp routes watchquagga[1853]: watchquagga 1.0.0+cl3u7 watching [zebra bgpd], mode [phased zebra restart] watchquagga[1853]: bgpd state -> up : connect succeeded watchquagga[1853]: bgpd state -> down : read returned EOF cumulus-core: Running cl-support for core files bgpd.3030.1470341944.core.core_helper core_check.sh[4992]: Please send /var/support /cl_support_spine01_20160804_201905.tar.xz to Cumulus support watchquagga[1853]: Forked background command [pid 6665]: /usr/sbin/service quagga restart bgpd watchquagga[1853]: watchquagga 0.99.24+cl3u2 watching [zebra bgpd ospfd], mode [phased zebra restart] watchquagga[1853]: zebra state -> up : connect succeeded watchquagga[1853]: bgpd state -> up : connect succeeded watchquagga[1853]: Watchquagga: Notifying Systemd we are up and running</pre>

Logging

The table below covers the various log files, and what they should be used for:

OSPF Element	Monitoring Command/s	Log Location
syslog	Catch all log file. Identifies memory leaks and CPU spikes.	



OSPF Element	Monitoring Command/s	Log Location
		<code>/var /log /sys log</code>
switchd functionality	Hardware Abstraction Layer (HAL).	<code>/var /log /swi tchd .log</code>
Routing daemons	Quagga zebra daemon details	<code>/var /log /dae mon. log</code>
Routing protocol	<p>The log file is configurable in Quagga. When quagga first boots, it boots using the non-integrated config so each routing protocol has its own log file. After booting up, quagga switches over to using the integrated configuration which means that all logs go to a single place.</p> <p>To edit where log files go use the command <code>log file <location></code>. By default, Quagga logs are not sent to syslog. This can be enabled using the command <code>log syslog <level></code>. After this, logs go through rsyslog and into <code>/var/log/syslog</code>.</p>	<code>/var /log /qua gga /zeb ra. log /var /log /qua gga/ {pro toco l}. log</code>



OSPF Element	Monitoring Command/s	Log Location
		/var /log /qua gga /Qua gga. log

Protocols and Services

NTP

Run the following command to confirm the NTP process is working correctly, and that the switch clock is synced with NTP:

```
cumulus@switch:~$ /usr/bin/ntpq -p
```

Device Management

Device Access Logs

Access Logs	Log Location	Log Entries
User Authentication and Remote Login	/var /log /syslog	sshd[31830]: Accepted publickey for cumulus from 192.168.0.254 port 45582 ssh2: RSA 38:e6:3b:cc:04:ac:41:5e:c9:e3:93:9d:cc:9e:48:25 sshd[31830]: pam_unix(sshd:session): session opened for user cumulus by (uid=0)



Device Super User Command Logs

Super User Command Logs	Log Location	Log Entries
Executing commands using sudo	/var /log /syslog	<pre>sudo: cumulus : TTY=unknown ; PWD=/home /cumulus ; USER=root ; COMMAND=/tmp /script_9938.sh -v sudo: pam_unix(sudo:session): session opened for user root by (uid=0) sudo: pam_unix(sudo:session): session closed for user root</pre>

Network Solutions

Data Center Host to ToR Architecture

This chapter discusses the various architectures and strategies available from the top of rack (ToR) switches all the way down to the server hosts.

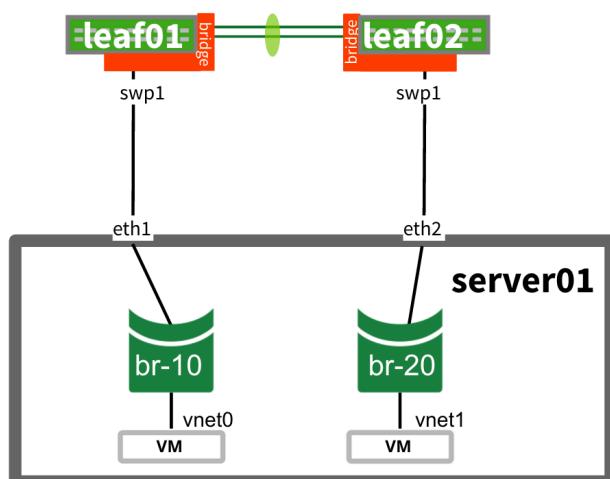
Contents

This chapter covers ...

- Layer 2 - Architecture (see page 744)
 - Traditional Spanning Tree - Single Attached (see page 744)
 - MLAG (see page 746)
- Layer 3 Architecture (see page 748)
 - Single-attached Hosts (see page 748)
 - Redistribute Neighbor (see page 750)
 - Routing on the Host (see page 751)
 - Routing on the VM (see page 752)
 - Virtual Router (see page 753)
 - Anycast with Manual Redistribution (see page 754)
- Network Virtualization (see page 756)
 - LNV with MLAG (see page 756)

Layer 2 - Architecture

Traditional Spanning Tree - Single Attached





Summary

Bond (see page 268)/Etherchannel is not configured on host to multiple switches (bonds can still occur but only to one switch at a time), so leaf01 and leaf02 see two different MAC addresses.

Configurations

leaf01 Config

```
auto bridge
iface bridge
    bridge-vlan-aware yes
    bridge-ports swp1 peerlink
    bridge-vids 1-2000
    bridge-stp on

auto bridge.10
iface bridge.10
    address 10.1.10.2/24

auto peerlink
iface peerlink
    bond-slaves glob swp49-50

auto swp1
iface swp1
    mstptctl-portadminedge yes
    mstptctl-bpduguard yes
```

Example Host Config (Ubuntu)

```
auto eth1
iface eth1 inet manual

auto eth1.10
iface eth1.10 inet manual

auto eth2
iface eth1 inet manual

auto eth2.20
iface eth2.20 inet manual

auto br-10
iface br-10 inet manual
    bridge-ports eth1.10 vnet0
```

More Information

Benefits

- Established technology
 - Interoperability with other vendors
 - Easy configuration for customer
 - Immense documentation from multiple vendors and industry
- Ability to use [spanning tree \(see page 237\)](#) commands
 - mstptctl-portadminedge
 - [BPDU guard \(see page 244\)](#)
- Layer 2 reachability to all VMs

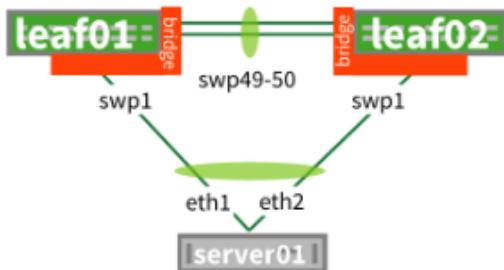
Caveats

- The load balancing mechanism on the host can cause problems. If there is only host pinning to each NIC, there are no problems, but if you are doing a bond, you need to look at an MLAG solution.
- No active-active host links. Some operating systems allow HA (NIC failover), but this still does not utilize all the bandwidth. VMs are using one NIC, not two.

Summary	More Information
<pre>auto br-20 iface br-20 inet manual bridge-ports eth2.20 vnet1</pre>	

Active-Active Mode	Active-Passive Mode	L2 to L3 Demarcation
<ul style="list-style-type: none"> None (not possible with traditional spanning tree) 	<ul style="list-style-type: none"> VRR (see page 337) vrrpd 	<ul style="list-style-type: none"> ToR layer (recommended) Spine layer Core/edge/exit <p>More Info... VRR or vrrpd can be configured on a pair of switches at any level in the network. However, the higher up the network you configure it, the larger the L2 domain becomes. The benefit here is L2 reachability. The drawback is the L2 domain is more difficult to troubleshoot, does not scale as well, and the pair of switches running VRR/vrrpd needs to carry the entire MAC address table of everything below it in the network. Minimizing the L2 domain as much as possible is recommended by Cumulus Professional Services. Please see this presentation for more information.</p>

MLAG



Summary	More Information
<p>MLAG (see page 300) (multi-chassis link aggregation) is when both uplinks are utilized at the same time. VRR gives the ability for both spines to act as gateways simultaneously for HA (high availability) and active-active mode (see page 406) (both are being used at the same time).</p> <p>Configurations</p>	<p>Benefits</p> <ul style="list-style-type: none"> 100% of links utilized <p>Caveats</p> <ul style="list-style-type: none"> More complicated (more moving parts) More configuration



Summary

leaf01 Config

```
auto bridge
iface bridge
    bridge-vlan-aware yes
    bridge-ports host-01 peerlink
    bridge-vids 1-2000
    bridge-stp on

auto bridge.10
iface bridge.10
    address 172.16.1.2/24
    address-virtual 44:38:39:00:00:10
    172.16.1.1/24

auto peerlink
iface peerlink
    bond-slaves glob swp49-50

auto peerlink.4094
iface peerlink.4094
    address 169.254.1.2
    clagd-enable yes
    clagd-peer-ip 169.254.1.2
    clagd-system-mac 44:38:39:FF:40:94

auto host-01
iface host-01
    bond-slaves swp1
    clag-id 1
    {bond-defaults removed for brevity}
```

More Information

- No interoperability between vendors
- ISL (inter-switch link) required

Additional Comments

- Can be done with either the [traditional \(see page 272\)](#) or [VLAN-aware \(see page 277\)](#) bridge driver depending on overall STP needs
- There are a few different solutions including Cisco VPC and Arista MLAG, but none of them interoperate and are very vendor specific
- Cumulus Networks Layer 2 HA validated design guide

Example Host Config (Ubuntu)

```
auto bond0
iface bond0 inet manual
    bond-slaves eth0 eth1
    {bond-defaults removed for brevity}

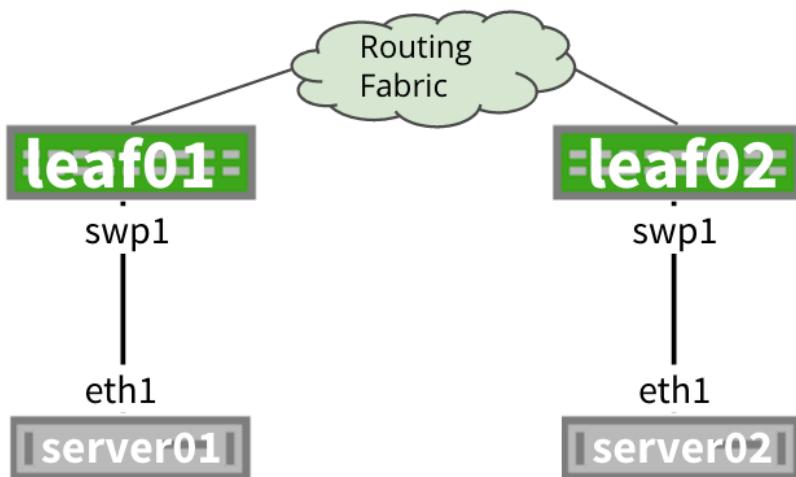
auto bond0.10
iface bond0.10 inet manual

auto vm-br10
iface vm-br10 inet manual
    bridge-ports bond0.10 vnet0
```

Active-Active Mode	Active-Passive Mode	L2->L3 Demarcation
<ul style="list-style-type: none"> • VRR (see page 337) 	<ul style="list-style-type: none"> • vrrpd 	<ul style="list-style-type: none"> • ToR layer (recommended) • Spine layer • Core/edge/exit

Layer 3 Architecture

Single-attached Hosts

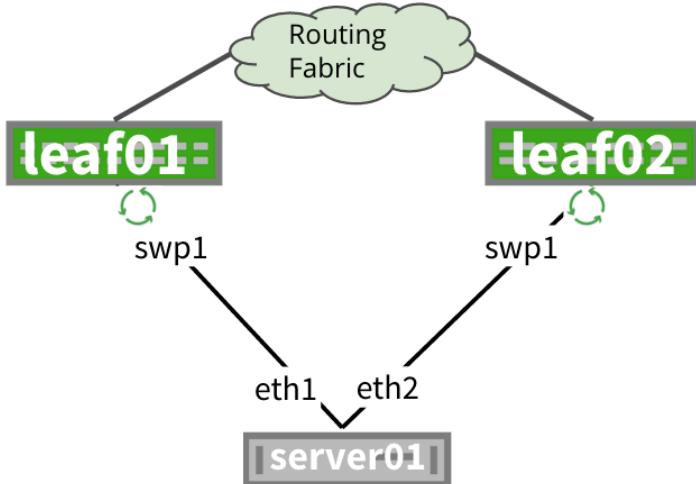


Summary	More Information
<p>The server (physical host) has only has one link to one ToR switch.</p> <p>Configurations</p> <p>leaf01 Config</p> <pre>/etc/network/interfaces</pre> <pre>auto swp1 iface swp1 address 172.16.1.1/30</pre> <p><pre>/etc/quagga/Quagga.conf</pre></p> <pre>router ospf router-id 10.0.0.11 interface swp1</pre>	<p>Benefits</p> <ul style="list-style-type: none"> • Relatively simple network configuration • No STP • No MLAG • No L2 loops • No crosslink between leafs • Greater route scaling and flexibility <p>Caveats</p> <ul style="list-style-type: none"> • No redundancy for ToR, upgrades would cause downtime • Many customers do not have software to support application layer redundancy <p>Additional Comments</p> <ul style="list-style-type: none"> • For additional bandwidth links between host and leaf may be bonded



Summary	More Information
<pre>ip ospf area 0</pre>	
<p>leaf02 Config</p> <p>/etc/network/interfaces</p> <pre>auto swp1 iface swp1 address 172.16.2.1/30</pre>	
<p>/etc/quagga/Quagga.conf</p> <pre>router ospf router-id 10.0.0.12 interface swp1 ip ospf area 0</pre>	
<p>host1 Example Config (Ubuntu)</p> <pre>auto eth1 iface eth1 inet static address 172.16.1.2/30 up ip route add 0.0.0.0/0 nexthop via 172.16.1.1</pre>	
<p>host2 Example Config (Ubuntu)</p> <pre>auto eth1 iface eth1 inet static address 172.16.2.2/30 up ip route add 0.0.0.0/0 nexthop via 172.16.2.1</pre>	
FHR (First Hop Redundancy)	More Information
<ul style="list-style-type: none">• No redundancy, uses single ToR as gateway.	<ul style="list-style-type: none">• Big Data validated design guide uses single attached ToR

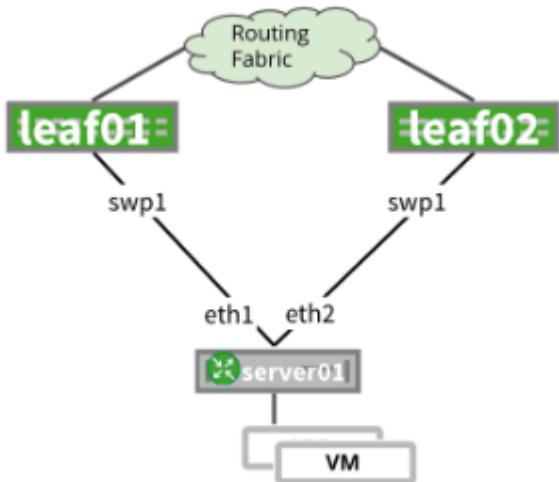
Redistribute Neighbor



Summary	More Information
<p>Redistribute neighbor daemon grabs ARP entries dynamically, utilizes redistribute table for Quagga to grab these dynamic entries and redistribute them into the fabric.</p>	<p>Benefits</p> <ul style="list-style-type: none"> • Configuration in Quagga is simple (route-map + redist table) • Supported by Cumulus Networks <p>Caveats</p> <ul style="list-style-type: none"> • Silent hosts don't receive traffic (depending on ARP). • IPv4 only. • If two VMs are on same L2 domain, they could learn about each other directly rather than utilizing gateway, which causes problems (VM migration for example, or getting their network routed). Put hosts on /32 (no other L2 adjacency). • VM move does not trigger route withdrawal from original leaf (4 hour timeout). • Clearing ARP impacts routing. May not be obvious. • No L2 adjacency between servers without VXLAN.
<p>FHR (First Hop Redundancy)</p> <ul style="list-style-type: none"> • Equal cost route installed on server/host /hypervisor to both ToRs to load balance evenly. 	<p>More Information</p> <ul style="list-style-type: none"> • Cumulus Networks blog post introducing redistribute neighbor

Summary	More Information
<ul style="list-style-type: none"> For host/VM/container mobility, use the same default route on all hosts (such as x.x.x.1) but don't distribute or advertise the .1 on the ToR into the fabric. This allows the VM to use the same gateway no matter which pair of leafs it is cabled to. 	

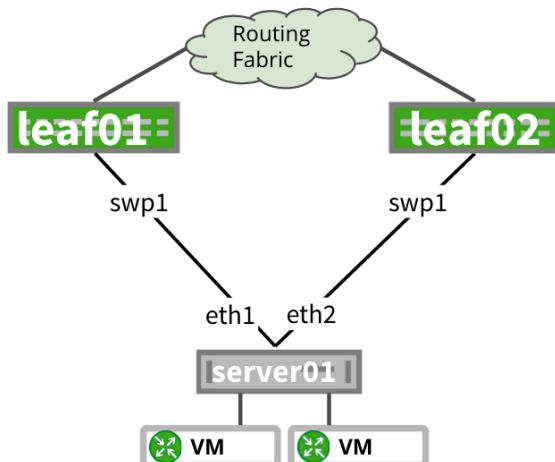
Routing on the Host



Summary	More Information
<p>Routing on the host means there is a routing application (such as Cumulus Networks Quagga) either on the bare metal host (no VMs /containers) or the hypervisor (for example, Ubuntu with KVM). This is highly recommended by the Cumulus Networks Professional Services team.</p>	<p>Benefits</p> <ul style="list-style-type: none"> No requirement for MLAG No spanning-tree or layer 2 domain No loops 3 or more ToRs can be used instead of usual 2 Host and VM mobility Traffic engineering can be used to migrate traffic from one ToR to another for upgrading both hardware and software <p>Caveats</p>

Summary	More Information
	<ul style="list-style-type: none"> • Certain hypervisors or host OSes might not support a routing application like Quagga and will require a virtual router on the hypervisor • No L2 adjacency between servers without VXLAN
FHR (First Hop Redundancy)	More Information
<ul style="list-style-type: none"> • The first hop is still the ToR, just like redistribute neighbor • A default route can be advertised by all leaf/ToRs for dynamic ECMP paths 	<ul style="list-style-type: none"> • Routing on the Host: An Introduction • Installing the Cumulus Linux Quagga Package on an Ubuntu Server • Configuring Quagga (see page 489)

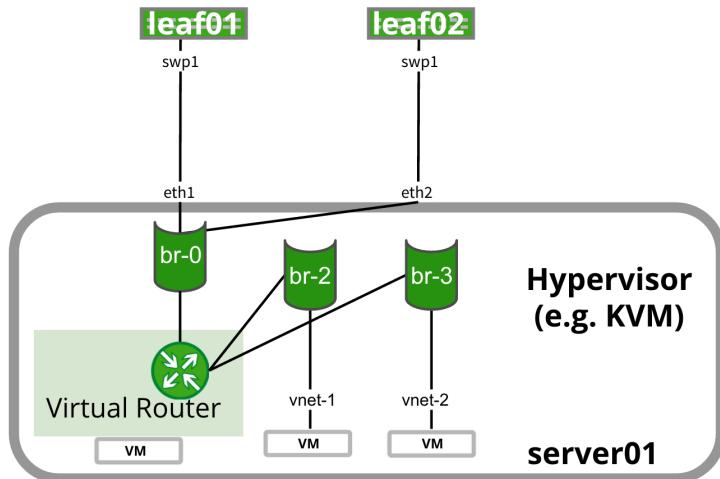
Routing on the VM



Summary	More Information
<p>Instead of routing on the hypervisor, each virtual machine utilizes its own routing stack.</p>	<p>Benefits</p> <ul style="list-style-type: none"> • In addition to routing on host: <ul style="list-style-type: none"> • Hypervisor/base OS does not need to be able to do routing • VMs can be authenticated into routing fabric

Summary	More Information
	<p>Caveats</p> <ul style="list-style-type: none"> • All VMs must be capable of routing • Scale considerations might need to be taken into account — instead of one routing process, there are as many as there are VMs • No L2 adjacency between servers without VXLAN
FHR (First Hop Redundancy)	<p>More Information</p> <ul style="list-style-type: none"> • Routing on the host: An Introduction • Installing the Cumulus Linux Quagga Package on an Ubuntu Server • Configuring Quagga (see page 489)

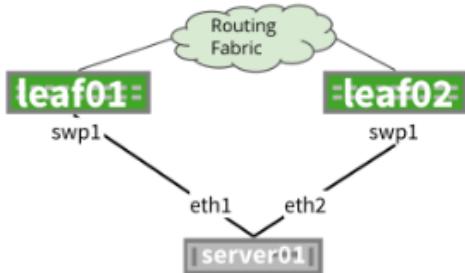
Virtual Router



Summary	More Information
<p>Virtual router (vRouter) runs as a VM on the hypervisor/host, sends routes to the ToR using BGP (see page 744) or OSPF (see page 500).</p>	<p>Benefits</p> <p>In addition to routing on a host:</p> <ul style="list-style-type: none"> • Multi-tenancy can work (multiple customers sharing same racks) • Base OS does not need to be routing capable <p>Caveats</p>

Summary	More Information
	<ul style="list-style-type: none"> • ECMP (see page 556) might not work correctly (load balancing to multiple ToRs); Linux kernel in older versions is not capable of ECMP per flow (does it per packet) • No L2 adjacency between servers without VXLAN
FHR (First Hop Redundancy)	More Information <ul style="list-style-type: none"> • Routing on the Host: An Introduction • Installing the Cumulus Linux Quagga Package on an Ubuntu Server • Configuring Quagga (see page 489)

Anycast with Manual Redistribution



Summary	More Information
<p>In contrast to routing on the host (preferred), this method allows a user to route to the host. The ToRs are the gateway, as with redistribute neighbor, except because there is no daemon running, the networks must be manually configured under the routing process. There is a potential to black hole unless a script is run to remove the routes when the host no longer responds.</p> <p>Configurations</p> <p>leaf01 Config</p> <pre>/etc/network/interfaces</pre> <pre> auto swp1 iface swp1 address 172.16.1.1/30 </pre> <p>/etc/quagga/Quagga.conf</p>	<p>Benefits</p> <ul style="list-style-type: none"> • Most benefits of routing on the host • No requirement for host to run routing • No requirement for redistribute neighbor <p>Caveats</p> <ul style="list-style-type: none"> • Removing a subnet from one ToR and re-adding it to another (hence, network statements from your router process) is a manual process

Summary

```
router ospf
  router-id 10.0.0.11
  interface swp1
    ip ospf area 0
```

leaf02 Config

/etc/network/interfaces

```
auto swp2
iface swp2
  address 172.16.1.1/30
```

/etc/quagga/Quagga.conf

```
router ospf
  router-id 10.0.0.12
  interface swp1
    ip ospf area 0
```

Example Host Config (Ubuntu)

```
auto lo
iface lo inet loopback

auto lo:1
iface lo:1 inet static
  address 172.16.1.2/32
  up ip route add 0.0.0.0/0 nexthop via 172.16
  .1.1 dev eth0 onlink nexthop via 172.16.1.1
  dev eth1 onlink

auto eth1
iface eth2 inet static
  address 172.16.1.2/32

auto eth2
iface eth2 inet static
  address 172.16.1.2/32
```

More Information

- Network team and server team would have to be in sync, or server team controls the ToR, or automation is being used whenever VM migration happens
- When using VMs /containers it is very easy to black hole traffic, as the leafs continue to advertise prefixes even when VM is down
- No L2 adjacency between servers without VXLAN

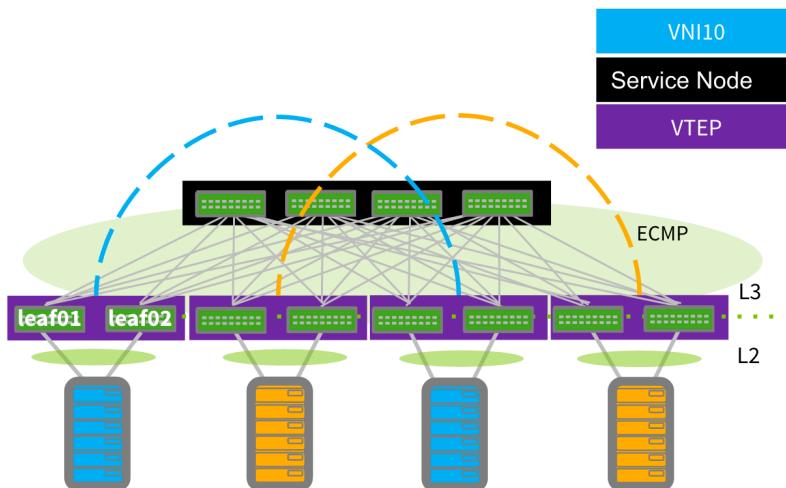
FHR (First Hop Redundancy)

More Information

- The gateways would be the ToRs, exactly like redistribute neighbor with an equal cost route installed

Network Virtualization

LNV with MLAG



Summary	More Information
<p>The host runs LACP (Etherchannel/bond) to the pair of ToRs. LNV (see page 378) (Lightweight Network Virtualization) then transports the L2 bridges across an L3 fabric.</p> <p>Configurations</p> <p>leaf01 Config</p> <pre data-bbox="204 1277 571 1309">/etc/network/interfaces</pre> <pre data-bbox="228 1383 832 1911"> auto lo iface lo inet loopback address 10.0.0.11/32 vxrd-src-ip 10.0.0.11 vxrd-svcnode-ip 10.10.10.10 clagd-vxlan-anycast-ip 36.0.0.11 auto vni-10 iface vni-10 vxlan-id 10 vxlan-local-tunnelip 10.0.0.11 auto br-10 iface br-10 bridge-ports swp1 vni-10 </pre>	<p>Benefits</p> <ul style="list-style-type: none"> Layer 2 domain is reduced to the pair of ToRs Aggregation layer is all L3 (VLANs do not have to exist on spine switches) Greater route scaling and flexibility High availability <p>Caveats</p> <ul style="list-style-type: none"> Needs MLAG (with the same caveats from the MLAG section (see page 746) above) and spanning tree (see page 237)



	Summary	More Information
	<p>leaf02 Config</p> <pre>/etc/network/interfaces</pre> <pre>auto lo iface lo inet loopback address 10.0.0.12/32 Vxrd-src-ip 10.0.0.12 vxrd-svcnode-ip 10.10.10.10 clagd-vxlan-anycast-ip 36.0.0.11 auto vni-10 iface vni-10 vxlan-id 10 vxlan-local-tunnelip 10.0.0.12 auto br-10 iface br-10 bridge-ports swp1 vni-10</pre>	

Active-Active Mode	Active-Passive Mode	Demarcation	
<ul style="list-style-type: none">• VRR (see page 337)	<ul style="list-style-type: none">• vrrpd (not recommended, not tested, will be asymmetrical)	<ul style="list-style-type: none">• ToR layer or exit leafs	
		More Information	
		<ul style="list-style-type: none">• Cumulus Linux Lightweight Network Virtualization (LNV) documentation (see page 378)	

Cumulus Networks Services Demos

The Cumulus Networks Services team demos provide a virtual environment built using either VirtualBox or libvirt using Vagrant to manage the VMs. This environment utilizes the reference topology shown below. Vagrant and Cumulus VX can be used together to build virtual simulations of production networks to validate configurations, develop automation code and simulate failure scenarios.

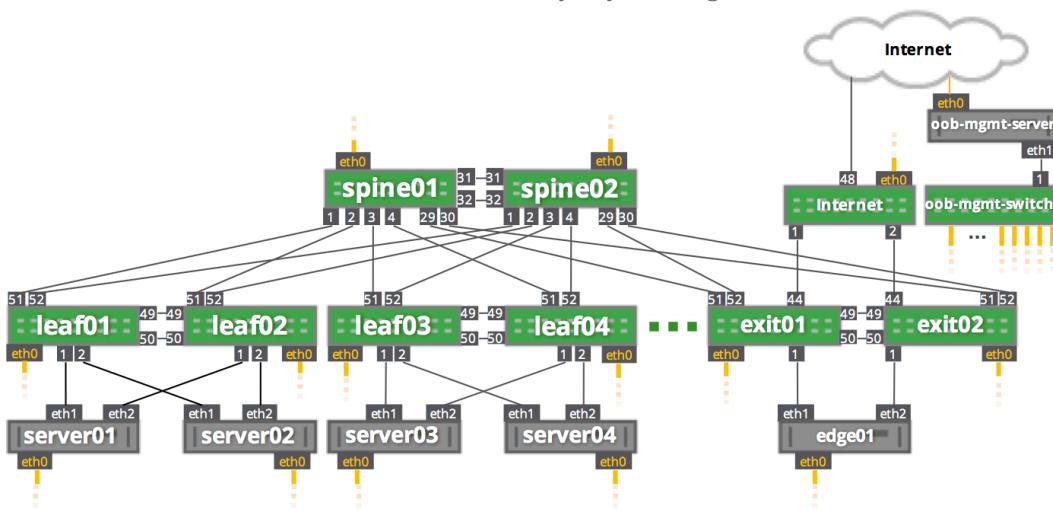
Contents

This chapter covers ...

- Reference Topology (see page 758)
 - IP and MAC Addressing (see page 758)
 - Building the Topology (see page 759)
 - Virtual Appliance (see page 759)
 - Hardware (see page 759)
 - Demos (see page 760)

Reference Topology

The Cumulus Networks *reference topology* includes cabling (in DOT format for dual use with [PTM \(see page 257\)](#)), MAC addressing, IP addressing, switches and servers. This topology is blessed by the Professional Services Team at Cumulus Networks to fit a majority of designs seen in the field.



IP and MAC Addressing

Hostname	eth0 IP	eth0 MAC	Interface Count
oob-mgmt-server	192.168.0.254	any	
oob-mgmt-switch	192.168.0.1	any	Cumulus RMP
leaf01	192.168.0.11	A0:00:00:00:00:11	48x10g w/ 6x40g uplink
leaf02	192.168.0.12	A0:00:00:00:00:12	48x10g w/ 6x40g uplink
leaf03	192.168.0.13	A0:00:00:00:00:13	48x10g w/ 6x40g uplink
leaf04	192.168.0.14		48x10g w/ 6x40g uplink



Hostname	eth0 IP	eth0 MAC	Interface Count
		A0:00:00:00:00:14	
spine01	192.168.0.21	A0:00:00:00:00:21	32x40g
spine02	192.168.0.22	A0:00:00:00:00:22	32x40g
server01	192.168.0.31	A0:00:00:00:00:31	10g NICs
server02	192.168.0.32	A0:00:00:00:00:32	10g NICs
server03	192.168.0.33	A0:00:00:00:00:33	10g NICs
server04	192.168.0.34	A0:00:00:00:00:34	10g NICs
exit01	192.168.0.41	A0:00:00:00:00:41	48x10g w/ 6x40g uplink (exit leaf)
exit02	192.168.0.42	A0:00:00:00:00:42	48x10g w/ 6x40g uplink (exit leaf)
edge01	192.168.0.51	A0:00:00:00:00:51	10g NICs (customer edge device, firewall, load balancer, etc.)
internet	192.168.0.253	any	(represents internet provider edge device)

Building the Topology

Virtual Appliance

You can build out the reference topology in hardware or using Cumulus VX (the free Cumulus Networks virtual appliance). The [Cumulus Reference Topology using Vagrant](#) is essentially the reference topology built out inside Vagrant with VirtualBox or KVM. The installation and setup instructions for bringing up the entire reference topology on a laptop or server are on the [cldemo-vagrant GitHub repo](#).

Hardware

Any switch from the [hardware compatibility list](#) is compatible with the topology as long as you follow the interface count from the table above. Of course, in your own production environment, you don't have to use exactly the same devices and cabling as outlined above.



Demos

You can find an up to date list of all the demos in the [cldemo-vagrant GitHub repository](#), which is available to anyone free of charge.

Docker on Cumulus Linux

Cumulus Linux 3.2 is based on Linux kernel 4.1, which supports the [Docker](#) engine. This means you can install Docker directly on a Cumulus Linux switch and you can run Docker containers natively on the switch.

```
cumulus@switch:~$ uname -r  
4.1.0-cl-1-amd64
```

Contents

This chapter covers ...

- [Installing Docker \(see page 760\)](#)
 - [Verifying the Docker Install \(see page 760\)](#)
- [Caveats and Errata \(see page 761\)](#)
 - [iptables and Docker \(see page 761\)](#)
 - [Management VRF and Docker \(see page 762\)](#)

Installing Docker

Before installing Docker, add the Debian Jessie `apt` repository to Cumulus Linux:

```
cumulus@switch:~$ echo "deb http://ftp.us.debian.org/debian/ jessie  
main contrib non-free" | sudo tee -a /etc/apt/sources.list
```

To install the Docker Engine, follow the [instructions](#) for installing it on Debian Linux, as they work for Cumulus Linux as is.



There's a community-supported Ansible playbook for installing Docker on Cumulus Linux on [GitHub](#).

Verifying the Docker Install

Docker provides a "Hello World" testing application that can be run on Cumulus Linux to validate that Docker Engine was installed correctly.

```
cumulus@switch:~$ docker run ubuntu /bin/echo 'Hello world'
```



```
Hello world
```

Caveats and Errata

iptables and Docker

By default, Docker Engine creates `iptables` rules to manage Docker container connectivity and provide IP masquerade (NAT). Docker Engine adds NAT entries to the `nat` table. These ACL rules do not properly sync with the hardware of Cumulus Linux, so when you run `cl-acltool -i` to install new hardware based ACLs, the Docker Engine rules get removed.

When using Docker Engine on Cumulus Linux, you must disable the `iptables` functionality within Docker Engine. To do so, use the `--iptables=false` option when you start Docker Engine. To do this automatically, you can modify the `systemd` service startup parameters:

1. Create the directory `/etc/systemd/system/docker.service.d`:

```
cumulus@switch:~$ sudo mkdir /etc/systemd/system/docker.service.d
```

2. Create the file `/etc/systemd/system/docker.service.d/noiptables.conf` and populate it with the following:

```
cumulus@switch:~$ sudo nano /etc/systemd/system/docker.service.d/noiptables.conf
```

```
[Service]
ExecStart=
ExecStart=/usr/bin/docker daemon -H fd:// --iptables=false
```

3. Reload the `systemd` configuration settings; this has no impact on currently running services.

```
cumulus@switch:~$ sudo systemctl daemon-reload
```

4. If Docker Engine was already started, reset the nat table entries, then reset the software `iptables` rules:

```
cumulus@switch:~$ sudo iptables -t nat -F
cumulus@switch:~$ sudo cl-acltool -i
```

5. Restart the Docker Engine:

```
cumulus@switch:~$ sudo systemctl restart docker.service
```



To check if the Docker Engine `iptables` rules have been installed, run `iptables -L DOCKER` to print the current software `iptables` rules. If the Docker Engine `iptables` rules are installed, you will see output similar to the following:

```
cumulus@switch:~$ sudo iptables -L DOCKER
Chain DOCKER (1 references)
target  prot opt source destination
cumulus@switch:~$
```

If the Docker Engine `iptables` rules are not installed, an error gets returned:

```
cumulus@switch:~$ sudo iptables -L DOCKER
iptables: No chain/target/match by that name.
```

Management VRF and Docker

If you have a management VRF configured on a Cumulus Linux 3.0.z switch, you cannot run Docker commands that go to another node (for example, `docker pull`) over the management VRF.

OpenStack Neutron ML2 and Cumulus Linux

➊ Early Access Feature

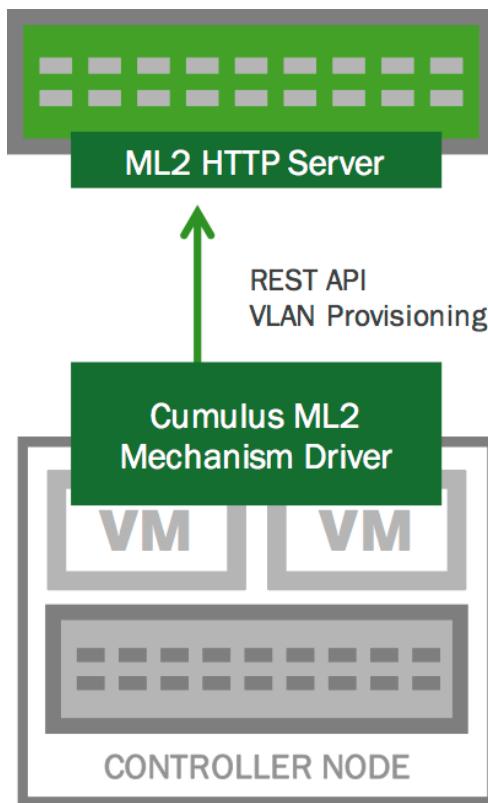
The REST API component is an [early access feature](#) in Cumulus Linux 3.2.1. Before you can install the API, you must enable the Early Access repository. For more information about the Cumulus Linux repository, read [this knowledge base article](#).

The Modular Layer 2 (ML2) plugin is a framework that allows OpenStack Networking to utilize a variety of non-vendor-specific layer 2 networking technologies. The ML2 framework simplifies adding support for new layer 2 networking technologies, requiring much less initial and ongoing effort — specifically, it enables dynamic provisioning of VLAN/VXLAN on switches in OpenStack environment instead of manually provisioning L2 connectivity for each VM.

The plugin supports configuration caching. The cached configuration is replayed back to the Cumulus Linux switch from Cumulus ML2 mechanism driver when a switch or process restart is detected.

In order to deploy [OpenStack ML2](#) in a network with Cumulus Linux switches, you need to install two packages:

- A REST API, which you install on your Cumulus Linux switches. It is in the Cumulus Linux [early access](#) repository.
- The Cumulus Networks Modular Layer 2 (ML2) mechanism driver for OpenStack, which you install on the OpenStack Neutron controller node. It's available as a Python package from upstream.



Contents

- Installing and Configuring the REST API (see page 763)
- Installing and Configuring the Cumulus Networks Modular Layer 2 Mechanism Driver (see page 764)
- Demo (see page 765)

Installing and Configuring the REST API

To install the `python-falcon` and `python-cumulus-restapi` packages, follow these instructions:

1. Open the `/etc/apt/sources.list` file in a text editor.
2. Uncomment the early access repository lines and save the file:

```
#deb http://repo3.cumulusnetworks.com/repo CumulusLinux-3-early-access
cumulus
#deb-src http://repo3.cumulusnetworks.com/repo CumulusLinux-3-early-
access cumulus
```

3. Run the following commands in a terminal to install the early access packages:



```
cumulus@switch:~$ sudo apt-get update  
cumulus@switch:~$ sudo apt-get install python-falcon python-cumulus-  
restapi
```

4. Then configure the relevant settings in /etc/restapi.conf:

```
[ML2]  
#local_bind = 10.40.10.122  
#service_node = 10.40.10.1  
  
# Add the list of inter switch links that  
# need to have the vlan included on it by default  
# Not needed if doing Hierarchical port binding  
#trunk_interfaces = uplink
```

5. Restart the REST API service for the configuration changes to take effect:

```
cumulus@switch:~$ sudo systemctl restart restserver
```

Additional REST API calls have been added to support the configuration of bridge using the bridge name instead of network ID.

Installing and Configuring the Cumulus Networks Modular Layer 2 Mechanism Driver

You need to install the Cumulus Networks ML2 mechanism driver on your Neutron host, which is available upstream:

```
root@neutron:~# git clone https://github.com/openstack/networking-  
cumulus.git  
root@neutron:~# cd networking-cumulus  
root@neutron:~# python setup.py install  
root@neutron:~# neutron-db-manage upgrade head
```

Then configure the host to use the ML2 driver:

```
root@neutron:~# openstack-config --set /etc/neutron/plugins/ml2  
/ml2_conf_cumulus.ini mechanism_drivers linuxbridge,cumulus
```



Finally, list the Cumulus Linux switches to configure. Edit `/etc/neutron/plugins/ml2/ml2_conf_cumulus.ini` in a text editor and add the IP addresses of the Cumulus Linux switches to the `switches` line. For example:

```
[ml2_cumulus]
switches="192.168.10.10,192.168.20.20"
```

The ML2 mechanism driver contains the following configurable parameters. You configure them in the `/etc/neutron/plugins/ml2/ml2_conf_cumulus.ini` file.

- `switches` — The list of Cumulus Linux switches connected to the Neutron host. Specify a list of IP addresses.
- `scheme` — The scheme (for example, HTTP) for the base URL for the ML2 API.
- `protocol_port` — The protocol port for the base URL for the ML2 API. The default value is `8000`.
- `sync_time` — A periodic time interval for polling the Cumulus Linux switch. The default value is `30` seconds.
- `spf_enable` — Enables/disables SPF for the bridge. The default value is `False`.
- `new_bridge` — Enables/disables [VLAN-aware bridge mode \(see page 277\)](#) for the bridge configuration. The default value is `False`, so a traditional mode bridge is created.

Demo

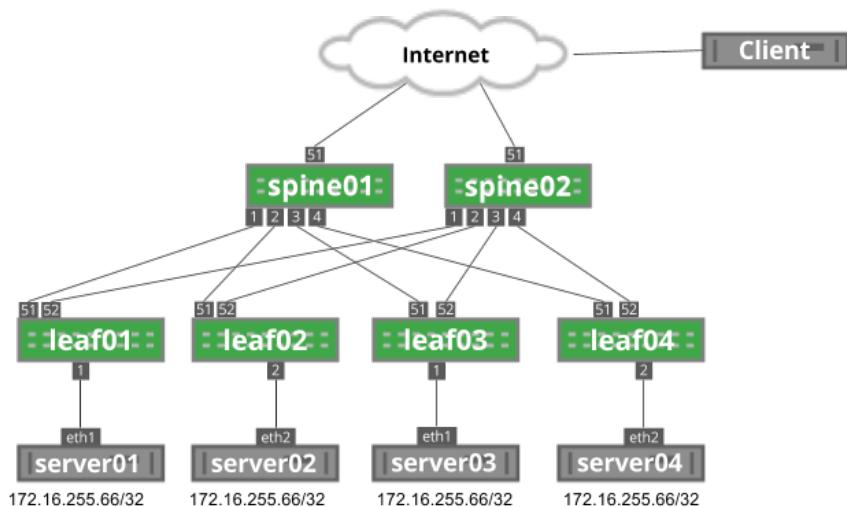
A demo involving OpenStack with Cumulus Linux is available in the [Cumulus Networks knowledge base](#). It demonstrates dynamic provisioning of VLANs using a virtual simulation of two Cumulus VX leaf switches and two CentOS 7 (RDO Project) servers; collectively they comprise an OpenStack environment.

Anycast Design Guide

Cumulus Networks' [Routing on the Host](#) provides the ability to run [OSPF \(see page 500\)](#) or [BGP \(see page 516\)](#) directly on server hosts. This can enable a network architecture known as *anycast*, where many servers can provide the same service without needing layer 2 extensions or load balancer appliances.

Anycast is not a new protocol or protocol implementation and does not require any additional network configuration. Anycast leverages the [equal cost multipath \(see page 556\)](#) (ECMP) capabilities inherent in layer 3 networks to provide stateless load sharing services.

The following image depicts an example anycast network. Each server is advertising the 172.16.255.66/32 anycast IP address.

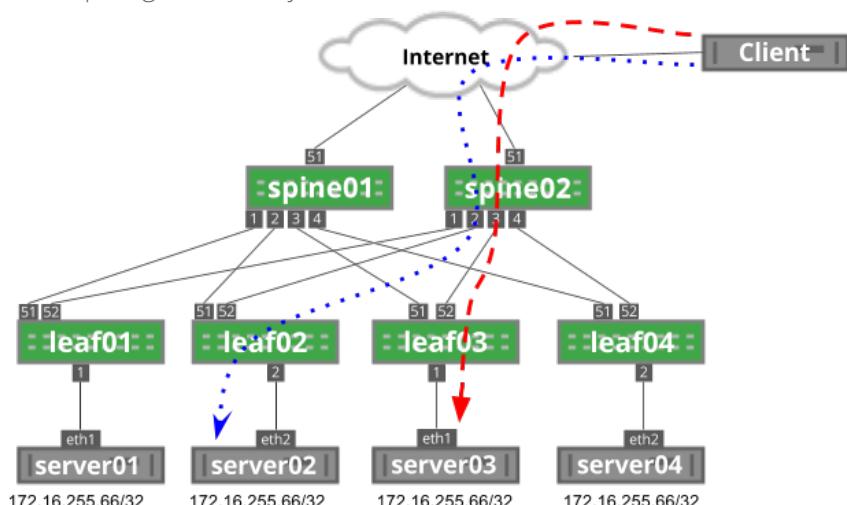


Anycast Architecture

Anycast relies on layer 3 equal cost multipath functionality to provide load sharing throughout the network. Each server announces a route for a service. As the route is propagated through the network, each network device sees the route as originating from multiple places. As an end user connects to the anycast IP, each network device performs a hardware hash of the layer 3 and layer 4 headers to determine which path to use.

Every packet in a flow from an end user has the same source and destination IP address as well as source and destination port numbers. The hash performed by the network devices results in the same answer for every packet, ensuring all packets in a flow are sent to the same destination.

In the following image, the client initiates two flows: the blue, dotted flow and the red dashed flow. Each flow has the same source IP address (the client's IP address), destination IP address (172.16.255.66) and same destination port (depending on the service; for example, DNS is port 53). Each flow has a unique source port generated by the client.



In this example, each flow hashes to different servers based on this source port, which you can see when you run `ip route show` to the destination IP address:

```
cumulus@spine02$ ip route show 172.16.255.66
172.16.255.66 proto zebra metric 20
```

```

nexthop via 169.254.64.0 dev swp1 weight 1
nexthop via 169.254.64.2 dev swp2 weight 1
nexthop via 169.254.64.2 dev swp3 weight 1
nexthop via 169.254.64.0 dev swp4 weight 1

```

On a Cumulus Linux switch, you can see the hardware hash with the `cl-ecmpcalc` command. In Figure 2, two flows originate from a remote user destined to the anycast IP address. Each session has a different source port. Using the `cl-ecmpcalc` command, you can see that the sessions were hashed to different egress ports.

```

cumulus@spine02$ sudo cl-ecmpcalc -p udp -s 10.2.0.100 --sport 32700 -d 172.31.255.66 --dport 53 -i swp51
ecmpcalc: will query hardware
swp2

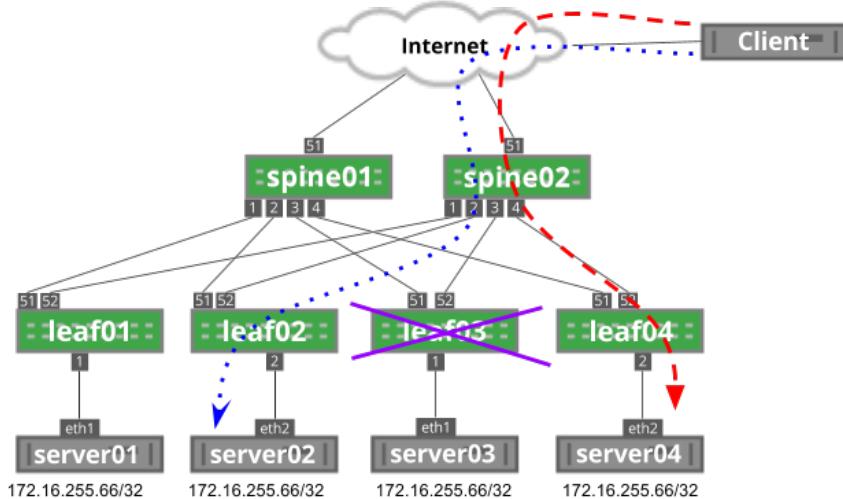
cumulus@spine02$ sudo cl-ecmpcalc -p udp -s 10.2.0.100 --sport 31884 -d 172.31.255.66 --dport 53 -i swp51
ecmpcalc: will query hardware
swp3

```

Anycast with TCP and UDP

A key component to the functionality and cost effective nature of anycast is that the network does not maintain state for flows. Every packet is handled individually through the routing table, saving memory and resources that would be required to track individual flows, similar to the functionality of a load balancing appliance.

As previously described, every packet in a flow hashes to the same next hop. However, if that next hop is no longer valid, the traffic flows to another anycast next hop instead. For example, in the image below, if leaf03 fails, traffic flows to a different anycast address; in this case, server04:



For stateless applications that rely on UDP, like DNS, this does not present a problem. However, for stateful applications that rely on TCP, like HTTP, this breaks any existing traffic flows, such as a file download. If the TCP three-way handshake was established on server03, after the failure, server04 would have no connection built and would send a TCP reset message back to the client, restarting the session.

This is not to say that it is not possible to use TCP-based applications for anycast. However, TCP applications in an anycast environment should have short-lived flows (measured in seconds or less) to reduce the impact of network changes or failures.

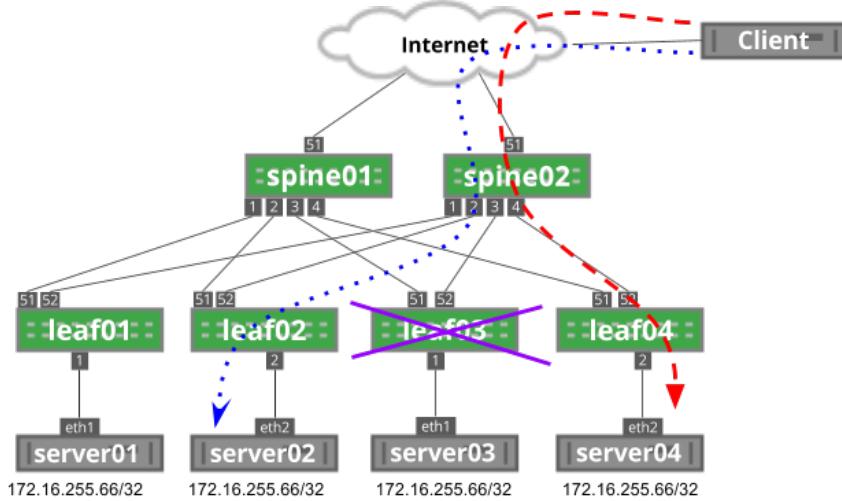
Resilient Hashing

Resilient hashing (see page 560) provides a method to prevent failures from impacting the hash result of unrelated flows. However, resilient hashing does not prevent rehashing when new next hops are added.

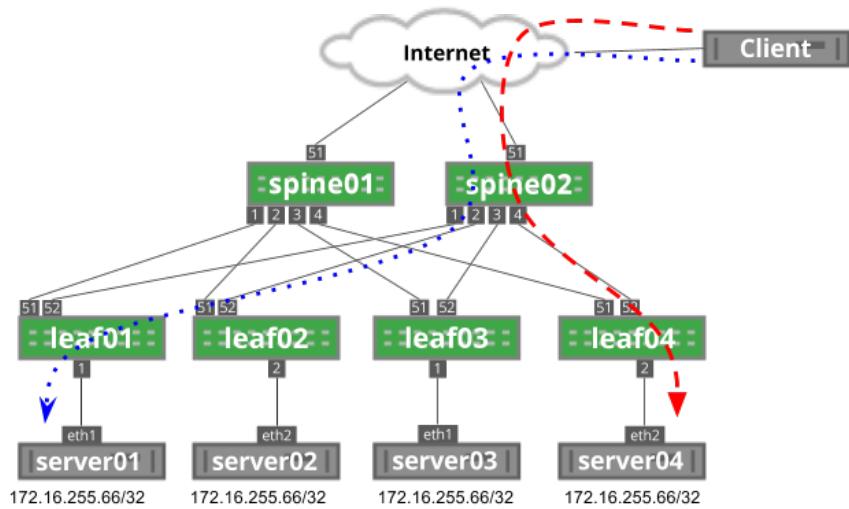
As previously mentioned, the hardware hashing function determines which path gets used for a given flow. The simplified version of that hash is the combination of protocol, source IP address, destination IP address, source layer 4 port and destination layer 4 port. The full hashing function includes not only these fields but also the list of possible layer 3 next hop addresses. The hash result is passed through a *modulo* of the number of next hop addresses. If the number of next hop addresses changes, through either addition or subtraction of the next hops, this changes the hash result for all traffic, including flows that have already established.

Continuing with the example in Figure 3, leaf03 is in a failed state, so traffic is hashing to server04. This is a result of the hash considering three possible next hop IPs (leaf01, leaf02, leaf04). When leaf03 is brought back online, the number of possible next hop IPs grows to four. This changes the modulo value that is part of the hashing function, which may result in traffic being sent to a different server, even if previously unaffected by the change.

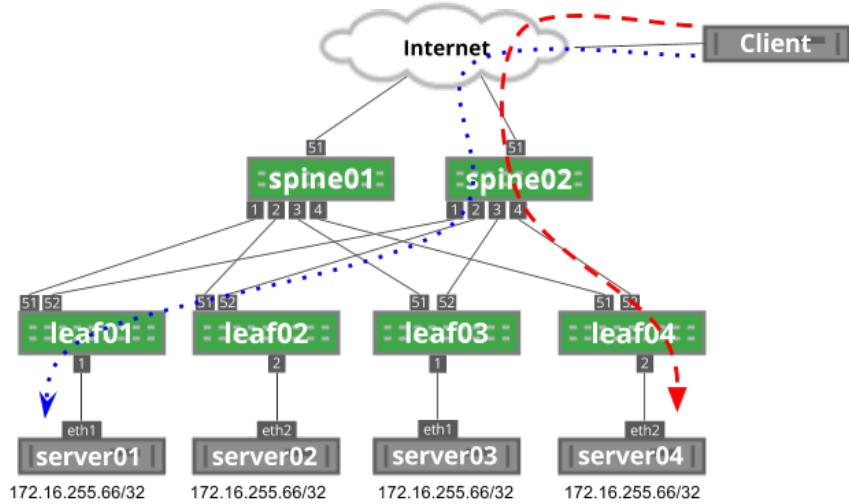
As you can see below, leaf03 is in a failed state. The blue dotted flow uses leaf02 to reach server02.



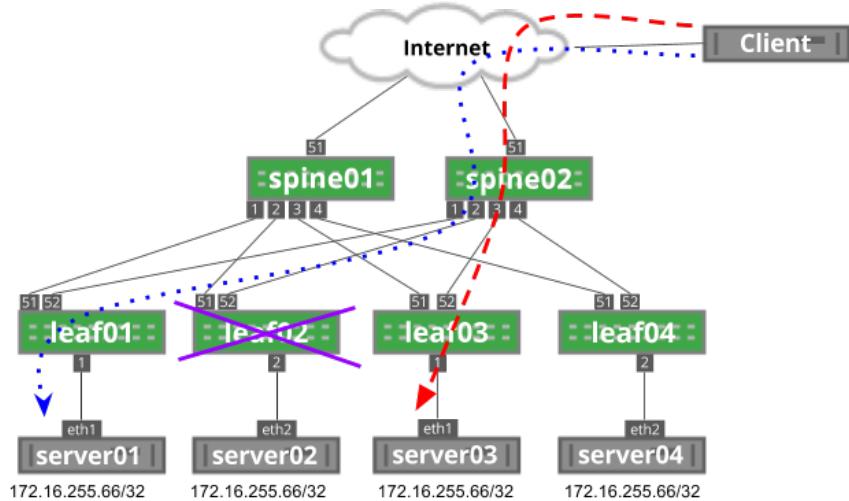
As leaf03 is brought back into service, the hashing function on spine02 changes, impacting the blue dotted flow:



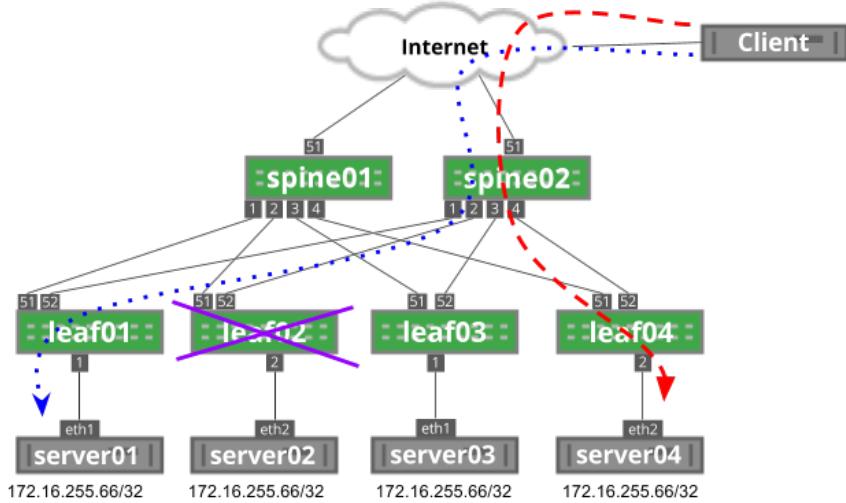
Just as the addition of a device can impact unrelated traffic, the removal of a device can also impact unrelated traffic, since again, the modulo of the hash function is changed. You can see this below, where the blue dotted flow goes through leaf01 and the red dashed line goes through leaf04.



Now, leaf02 has failed. As a result, the modulo on spine02 has changed from four possible next hops to only three next hops. In this example, the red dashed line has rehashed to leaf03:



To help solve this issue, resilient hashing can prevent traffic flows from shifting on unrelated failure scenarios. With resilient hashing enabled, the failure of leaf02 does not impact both existing flows, since they do not currently flow through leaf02:



Although resilient hashing can prevent rehashing on next hop failure, it cannot prevent rehashing on next hop addition.

You can read more information on resilient hashing in the [ECMP chapter \(see page 556\)](#).

Applications for Anycast

As previously mentioned, UDP-based applications are great candidates for anycast architectures, such as NTP or DNS.

When considering applications to be deployed in an anycast scenario, the first two questions to answer are:

- Whether the application relies on TCP for proper sequencing of data.
- Whether the application relies on more than one session as part of the application.

Applications with Multiple Connections

The network has no knowledge of any sessions or relationships between different sessions for the same application. This affects protocols that rely on more than one TCP or UDP connection to function properly — one example being FTP.

FTP data transfers require two connections: one for control and one for the file transfer. These two connections are independent, with their own TCP ports. Consider the scenario where an FTP server was deployed in an anycast architecture. When the secondary data connection is initiated, the traffic is destined initially to the same FTP server IP address, but the network hashes this traffic as a new, unique flow because the ports are different. This may result in the new session ending up on a new server. The new server would only accept that data connection if the FTP server application was capable of robust information sharing, as it has no history of the original request in the control session.

Initiating Traffic vs. Receiving Traffic

It is also important to understand that an outbound TCP session should never be initiated over an anycast IP address, as traffic that originates from an anycast IP address may not return to the same anycast server after the network hash. Contrast this with inbound sessions, where the network hash is the same for all packets in a flow, so the inbound traffic will hash to the same anycast server.



TCP and Anycast

TCP-based applications can be used with anycast, with the following recommendations:

- TCP sessions are short lived.
- The impact of a failed session or TCP reset does not impact the application. For example, a web page refresh is acceptable.
- There is application-level session management that is completely independent of the TCP session.
- A redirection middleware layer handles incorrectly hashed flows.

TCP applications that have longer-lived flows should not be used as anycast services. For example:

- FTP or other large file transfers.
- Transactions that must be completed and journaled. For example, financial transactions.
- Streaming media without application-level automated recovery.

It should be noted that anycast TCP is possible and has been implemented by a number of organizations, one notable example being LinkedIn.

Conclusion

Anycast can provide a low cost, highly scalable implementation for services. However, the limitations inherent in network-based ECMP makes anycast challenging to integrate with some applications. An anycast architecture is best suited for stateless applications or applications that are able to share session state at the application layer.



Index

4

40G ports [214](#)
logical limitations [214](#)

8

802.1p [218](#)
 class of service [218](#)
802.3ad link aggregation [333](#)

A

ABRs [502](#)
 area border routers [502](#)
access control lists [122](#)
access ports [294](#)
ACL policy files [136](#)
ACL rules [221](#)
ACLs [122, 125, 142](#)
 chains [125](#)
 QoS [142](#)
active-active mode [339, 406](#)
 VRR [339](#)
 VXLAN [406](#)
active listener ports [165](#)
Algorithm Longest Prefix Match [480](#)
 routing [480](#)
ALPM mode [480](#)
 routing [480](#)
AOC cables [20](#)
apt-get [62](#)
area border routers [502](#)
 ABRs [502](#)
arp cache [669](#)
ASN [518](#)
 autonomous system number [518](#)
auto-negotiation [202](#)
autonomous system number [518](#)
 BGP [518](#)



autoprovisioning 68

B

BFD 262, 554

 Bidirectional Forwarding Detection 262

 echo function 554

BGP 516, 519, 574

 Border Gateway Protocol 516

 ECMP 519

 virtual routing and forwarding (VRF) 574

BGP peering relationships 532, 532

 external 532

 internal 532

bonds 268, 333

 LACP Bypass 333

boot recovery 632

bpdufilter 247

 and STP 247

BPDU guard 244

 and STP 244

brctl 22

bridge assurance 247

 and STP 247

bridges 272, 273, 274, 274, 277, 291, 294, 294

 access ports 294

 adding IP addresses 274

 MAC addresses 274

 MTU 272

 trunk ports 294

 untagged frames 291

 VLAN-aware 273, 277

C

cable connectivity 20

cabling 257

 Prescriptive Topology Manager 257

chain 125

cl-acltool 122, 222, 670

clagctl 321

class of service 218

cl-cfg 173, 642



cl-ecmpcalc 557
cl-license 20
cl-netstat 664
cl-ospf6 515
Clos topology 486
cl-resource-query 173, 633
cl-support 626
convergence 485
 routing 485
Cumulus Linux 16, 17, 27, 27, 31, 434
 installing 16, 31
 reprovisioning 27
 uninstalling 27
 upgrading 17
 VXLAN 434
cumulus user 95

D

DAC cables 20
daemons 163
datapath 218, 222, 225
 link pause 225
 priority flow control 222
datapath.conf 218
date 89
 setting 89
deb 67
debugging 624
decode-syseeprom 635
differentiated services code point 218
dmidecode 636
dpkg 65
dpkg-reconfigure 88
DSCP 218
 differentiated services code point 218
DSCP marking 221
dual-connected hosts 303
duplex interfaces 209
dynamic routing 264, 488
 and PTM 264
 quagga 488



E

eBGP 518
 external BGP 518
ebtables 122, 130
 memory spaces 130
echo function 554, 554
 BFD 554
 PTM 554
ECMP 488, 513, 519, 563, 1
 BGP 519
 equal cost multi-pathing 488
 monitoring 1
 OSPF 513
 resilient hashing 563
ECMP hashing 557, 560
 resilient hashing 560
EGP 489
 Exterior Gateway Protocol 489
equal cost multipath 557
 ECMP hashing 557
equal cost multi-pathing 488
 ECMP 488
ERSPAN 671
 network troubleshooting 671
Ethernet management port 17
ethtool 217, 663
 switch ports 217
external BGP 518
 eBGP 518

F

fast convergence 530
 BGP 530
First Hop Redundancy Protocol 339
 VRR 339

G

globs 197
Graphviz 257



H

hardware [634](#)
 monitoring [634](#)
hardware compatibility list [14](#)
hash distribution [269](#)
HCL [14](#)
head end replication [380](#)
 LNV [380](#)
high availability [487](#)
host entries [633](#)
 monitoring [633](#)
hostname [18](#)
hsflowd [683](#)
hwclock [89](#)

I

iBGP [518](#)
 internal BGP [518](#)
ifdown [186](#)
ifquery [190, 659](#)
ifup [185](#)
ifupdown [185](#)
ifupdown2 [195, 292, 658, 658, 659](#)
 excluding interfaces [659](#)
 logging [658](#)
 purging IP addresses [195](#)
 troubleshooting [658](#)
 VLAN tagging [292](#)
IGMP snooping [327, 342](#)
 MLAG [327](#)
IGP [489](#)
 Interior Gateway Protocol [489](#)
image contents [29](#)
installing [16](#)
 Cumulus Linux [16](#)
interface counters [664](#)
interface dependencies [189](#)
interfaces [201, 215](#)
 statistics [215](#)
internal BGP [518](#)
 iBGP [518](#)
ip6tables [122](#)



IP addresses 195

 purging 195

iproute2 662

 failures 662

iptables 122

IPv4 routes 520

 BGP 520

IPv6 routes 520

 BGP 520

L

LACP 268, 300

 MLAG 300

LACP Bypass 333

layer 3 access ports 23

 configuring 23

LDAP 103

leaf-spine topology 486

license 19

 installing 19

lightweight network virtualization 378, 380, 380, 427

 head end replication 380

 service node replication 380

link aggregation 268

Link Layer Discovery Protocol 251

link-local IPv6 addresses 542

 BGP 542

link pause 225

 datapath 225

link-state advertisement 500

LLDP 251, 257

 SNMP 257

lldpcli 252

lldpd 251, 259

LNV 378, 378, 380, 380, 427, 427

 head end replication 380

 service node replication 380

 VXLAN 378, 427

load balancing 488

logging 627, 658, 658

 ifupdown2 658

 networking service 658

logging neighbor state changes 542



BGP 542
logical switch 300
longest prefix match 480
 routing 480
loopback interface 24
 configuring 24
LSA 500
 link-state advertisement 500
LSDB 500
 link-state database 500
lshw 636

M

MAC entries 633
 monitoring 633
Mako templates 198, 660
 debugging 660
mangle table 222
 ACL rules 222
memory spaces 130
 ebtables 130
MLAG 300, 322, 322, 322, 327, 329, 331
 backup link 322
 IGMP snooping 327
 MTU 329
 peer link states 322
 protodown state 322
 STP 331
MLD snooping 342
monitoring 87, 624, 633, 639, 639, 663, 682, 696
 hardware watchdog 639
 Net-SNMP 696
 network traffic 682
mstpcctl 241, 296
MTU 210, 272, 329, 662
 bridges 272
 failures 662
 MLAG 329
multi-Chassis Link Aggregation 300
 MLAG 300
multiple bridges 290
mz 669
 traffic generator 669



N

name switch service 102
Netfilter 122
Net-SNMP 696
networking service 658
 logging 658
network interfaces 185, 201
 ifupdown 185
network traffic 682
 monitoring 682
network troubleshooting 679
 tcpdump 679
network virtualization 347, 348, 434
 VMware NSX 348
nonatomic updates 131
 switchd 131
non-blocking networks 487
NSS 102
 name switch service 102
NTP 90
 time 90
ntp 90

O

ONIE 16, 28
 rescue mode 28
onie-select 27
Open Network Install Environment 16
Open Shortest Path First Protocol 500, 514
 OSPFv2 500
 OSPFv3 514
open source contributions 14
OSPF 505, 511, 513, 513
 ECMP 513
 reconvergence 513
 summary LSA 505
 unnumbered interfaces 511
ospf6d.conf 515
OSPFv2 500
OSPFv3 514, 516
 unnumbered interfaces 516



over-subscribed networks [487](#)

P

packages [62](#)

 managing [62](#)

packet buffering [218](#)

 datapath [218](#)

packet queueing [218](#)

 datapath [218](#)

packet scheduling [218](#)

 datapath [218](#)

PAM [102](#)

 pluggable authentication modules [102](#)

parent interfaces [192](#)

password [95](#)

 default [95](#)

passwords [17](#)

peer groups [531](#)

 BGP [531](#)

Per VLAN Spanning Tree [238](#)

 PVST [238](#)

ping [668](#)

pluggable authentication modules [102](#)

policy.conf [138](#)

port lists [197](#)

port speeds [209](#)

Prescriptive Topology Manager [257](#)

priority flow control [222](#)

 datapath [222](#)

priority groups [218](#)

 datapath [218](#)

privileged commands [97](#)

protocol tuning [485, 546](#)

 BGP [546](#)

 routing [485](#)

protodown state [322](#)

 MLAG [322](#)

PTM [257, 554](#)

 echo function [554](#)

 Prescriptive Topology Manager [257](#)

ptmctl [265](#)

ptmd [257](#)

PTM scripts [260](#)



PVRST [238](#)
 Rapid PVST [238](#)
PVST [238](#)
 Per VLAN Spanning Tree [238](#)

Q

QoS [142](#)
 ACLs [142](#)
QSFP [665](#)
Quagga [264, 264, 488, 490](#)
 and PTM [264, 264](#)
 configuring [490](#)
 dynamic routing [488](#)
quality of service [227](#)
querier [343](#)
 IGMP/MLD snooping [343](#)

R

Rapid PVST [238](#)
 PVRST [238](#)
read-only mode [545](#)
 BGP [545](#)
recommended configuration [46](#)
reconvergence [513](#)
 OSPF [513](#)
repositories [66](#)
 other packages [66](#)
rescue mode [28](#)
resilient hashing [560, 563](#)
 ECMP [563](#)
restart [173](#)
 switchd [173](#)
root user [17, 95](#)
route advertisements [518](#)
 BGP [518](#)
route maps [480, 512, 545](#)
 BGP [480, 512, 545](#)
route reflectors [518](#)
 BGP [518](#)
routes [633](#)
 monitoring [633](#)



routing protocols [485](#)

RSTP [238](#)

S

sensors command [636](#)

serial console management [17](#)

service node replication [380](#)

 LNV [380](#)

services [163](#)

sFlow [682](#)

sFlow visualization tools [685](#)

SFP [217, 665](#)

 switch ports [217](#)

single user mode [632](#)

smonctl [638](#)

smond [638](#)

snmpd [696](#)

sources.list [66](#)

SPAN [671](#)

 network troubleshooting [671](#)

spanning tree parameters [248](#)

Spanning Tree Protocol [237, 278](#)

 STP [237](#)

 VLAN-aware bridges [278](#)

static routing [479](#)

 with ip route [479](#)

storm control [248](#)

 STP [248](#)

STP [237, 247, 248, 331](#)

 and bridge assurance [247](#)

 MLAG [331](#)

 Spanning Tree Protocol [237](#)

 storm control [248](#)

stub areas [506](#)

 OSPF [506](#)

sudo [95, 97](#)

sudoers [97, 97](#)

 examples [97](#)

summary LSA [505](#)

 OSPF [505](#)

SVI [274, 307](#)

 bridges [274](#)

 switched virtual interface [307](#)



switchd 131, 171, 171, 173, 642

configuring 171

counters 642

file system 171

nonatomic updates 131

restarting 173

switched virtual interface 307

SVI 307

switched VLAN interface 274

bridges 274

switch ports 21, 214

configuring 21

logical limitations 214

syslog 627

systemd 327

system management 624

T

tcpdump 679

network troubleshooting 679

templates 198

time 89

setting 89

time zone 88

topology 257, 486

data center 257

traceroute 668

traffic.conf 218, 218

traffic distribution 269

traffic generator 669

mz 669

traffic marking 221

datapath 221

troubleshooting 624, 632, 679

single user mode 632

tcpdump 679

trunk ports 291, 294

tzdata 88

U

U-Boot 16, 624



unnumbered interfaces [511](#), [516](#)

OSPF [511](#)

OSPFv3 [516](#)

untagged frames [291](#)

bridges [291](#)

upgrading [17](#)

Cumulus Linux [17](#)

user accounts [95](#)

cumulus [95](#)

root [95](#)

user authentication [102](#)

user commands [195](#)

interfaces [195](#)

V

virtual device counters [639](#), [642](#), [643](#)

monitoring [639](#)

poll interval [642](#)

VLAN statistics [643](#)

virtual routing and forwarding (VRF) [574](#), [576](#)

BGP [574](#)

table ID [576](#)

visudo [97](#)

VLAN [307](#), [639](#)

statistics [639](#)

switched virtual interface [307](#)

VLAN-aware bridges [273](#), [277](#), [278](#)

Spanning Tree Protocol [278](#)

VLAN tagging [292](#), [293](#), [294](#)

advanced example [294](#)

basic example [293](#)

VLAN translation [299](#)

VTEP [347](#), [350](#)

vtysh [492](#)

quagga CLI [492](#)

VXLAN [347](#), [349](#), [378](#), [406](#), [427](#), [434](#), [639](#)

active-active mode [406](#)

LENV [378](#), [427](#)

no controller [434](#)

statistics [639](#)

VMware NSX [349](#)



W

watchdog [639](#)
monitoring [639](#)

Z

zebra [489](#)
routing [489](#)
zero touch provisioning [68, 69](#)
USB [69](#)
ZTP [68](#)