# Cumulus NetQ 1.4.0

# Telemetry User Guide

# Table of Contents

This guide is intended for network administrators who are responsible for monitoring and troubleshooting the network in their data center environment. NetQ 1.4 offers the ability to easily monitor and manage your data center network infrastructure and operational health. This guide provides instructions and information about monitoring both individual components of the network, the network as a whole, and the NetQ software itself using the NetQ command line interface (CLI).

This guide is organized into the following topics:

# Telemetry Preface

A variety of resources are available for you to become familiar with Cumulus NetQ and to take advantage of its monitoring and analytic capabilities. These resources are identified here along with information about how the content is presented.

## Contents

This topic describes...

## What's New in Cumulus NetQ 1.4.0

Cumulus NetQ 1.4.0 includes the following new features:

- Added
  - support for monitoring up to 200 Cumulus Linux nodes
  - validation of symmetric VXLAN routes through CLI
  - validation of forward error correction (FEC) operation through NetQL
- Up dated
  - color cues for `netq show services` command to more easily view status of services at a glance
  - NetQ CLI syntax for creating NetQ Notifier filters t o improve usability and operation
  - trace functionality to improve usability and operation
- Early access feature
  - Image and Provisioning Management (IPM) application

This version of NetQ includes a number of CLI changes. Refer to NetQ Command Line Overview (see page 10) for details.

For further information regarding bug fixes and known issues present in this release, refer to the release notes.

## Available Documentation

The NetQ documentation set has been reorganized and updated from prior releases. They still provide the information you need to proactively monitor your Linux-based network fabric using Cumulus NetQ. They assume that you have already installed Cumulus Linux and NetQ.

You may start anywhere in the documentation or read it from start to finish depending on your role and familiarity with the NetQ software and Linux networking. If you are new to NetQ, you may want to read the Cumulus NetQ Primer before reading the other available documents to gain a high-level understanding of the product capabilities and operation .

The following NetQ documents are available:

- Cumulus NetQ Primer
- Cumulus NetQ Deployment Guide
- Cumulus NetQ Telemetry User Guide (this guide in PDF )
- Cumulus NetQ Image and Provisioning Management User Guide
- Cumulus NetQ Release Notes
- Cumulus NetQ Data Sheet

# Document Formatting

The Cumulus NetQ Deployment Guide uses the following typographical and note conventions.

## Typographical Conventions

Throughout the guide, text formatting is used to convey contextual information about the content.

| Text Format | Meaning |
|---|---|
| Green text | Link to additional content within the topic or to another topic |
| `Text in Monospace font` | Filename, directory and path names, and command usage |
| [`Text within square brackets`] | Optional command parameters; may be presented in mixed case or all caps text |
| <`Text within angle brackets`> | Required command parameter values–variables that are to be replaced with a relevant value; may be presented in mixed case or all caps text |

## Note Conventions

Several note types are used throughout the document. The formatting of the note indicates its intent and urgency.

✓ **Tip or Best Practice**

Offers information to improve your experience with the tool, such as time-saving or shortcut options, or i ndicates the common or recommended method for performing a particular task or process

### ⓘ Information

Provides additional information or a reminder about a task or process that may impact your next step or selection

### ⚠ Caution

Advises that failure to take or avoid specific action can result in possible data loss

### ⓘ Warning

Advises that failure to take or avoid specific action can result in possible physical harm to yourself, hardware equipment, or facility

# NetQ Command Line Overview

The NetQ CLI provides access to all of the network state and event information collected by the NetQ Agents. It behaves the same way most CLIs behave, with groups of commands used to display related information, the ability to use TAB completion when entering commands, and to get help for given commands and options. The commands are grouped into four categories: check and show, agent and notifier, trace, and resolve.

⚠ The NetQ command line interface only runs on switches and server hosts implemented with Intel x86 or ARM-based architectures. If you are unsure what architecture your switch or server employs, check the Cumulus Hardware Compatibility List and verify the value in the **Platforms** tab > **CPU** column.

## Contents

This topic describes...

# CLI Access

When NetQ is installed, the CLI is also installed and enabled (refer to the Install NetQ topic). Simply log in to any network node to access the command line. If you want to run the CLI on the Telemetry Server (TS), Cumulus Networks recommends using netq-shell. While most other Linux commands can work from this shell, Cumulus Networks recommends you only run `netq` commands here.

To access the CLI from a switch or server:

1. Log in to device. This example uses a username of *Cumulus* and a hostname of *switch*.

```
<computer>:~Cumulus$ ssh switch
```

2. Enter your password, if required, to reach the command prompt. For example:

```
Enter passphrase for key '/Users/<username>/.ssh/id_rsa':
Welcome to Ubuntu 16.04.3 LTS (GNU/Linux 4.4.0-112-generic
x86_64)
 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage
Last login: Thu Aug 16 06:28:12 2018 from 10.50.11.103
<username>@<hostname>:~$
```

3. Run commands. For example:

```
<username>@<hostname>:~$netq show agents
<username>@<hostname>:~$netq check bgp
```

To access the CLI from a Telemetry Server:

1. Log in to TS. This example uses a username of *Cumulus* and a TS with a hostname of *ts* .

```
<computer>:~Cumulus$ ssh ts
```

2. Run netq-shell.

```
cumulus@ts:~$ netq-shell
Welcome to Cumulus (R) Linux (R)

For support and online technical documentation, visit
http://www.cumulusnetworks.com/support
```

```
The registered trademark Linux (R) is used pursuant to a
sublicense from LMI, the exclusive licensee of Linux Torvalds,
owner of the mark on a worldwide basis.

TIP: Type `netq` to access NetQ CLI.
cumulus@ts:~$
```

3. Run commands. For example:

```
Cumulus@ts:~$ netq show agent
Matching agents records:
Hostname    Status    Ntp Sync
Version                             Sys Uptime    Agent
Uptime    Reinitialize Time    Last Changed
----------  --------  ----------
----------------------------------  ------------
--------------  -------------------  --------------
leaf01      Fresh     yes          1.3.0-cl3u9~1522713084.
b08ca60      2h:42m:27s    2h:15m:36s      2h:15m:36s
23.663727s
leaf02      Fresh     yes          1.3.0-cl3u9~1522713084.
b08ca60      2h:42m:2s     2h:15m:37s      2h:15m:37s
35.518794s
leaf03      Fresh     yes          1.3.0-cl3u9~1522713084.
b08ca60      2h:42m:13s    2h:15m:36s      2h:15m:36s
9.191086s
leaf04      Fresh     yes          1.3.0-cl3u9~1522713084.
b08ca60      2h:42m:28s    2h:15m:37s      2h:15m:37s
9.809986s
server01    Fresh     yes          1.3.0-ub16.04u9~1522713679.
b08ca60  2h:29m:14s    2h:13m:41s      2h:13m:41s
12.207030s
server02    Fresh     yes          1.3.0-ub16.04u9~1522713679.
b08ca60  2h:29m:14s    2h:1m:8s        2h:1m:8s
31.850285s
server03    Fresh     yes          1.3.0-ub16.04u9~1522713679.
b08ca60  2h:29m:14s    2h:0m:21s       2h:0m:21s
15.317886s
server04    Fresh     yes          1.3.0-ub16.04u9~1522713679.
b08ca60  2h:29m:14s    2h:16m:33s      2h:16m:33s
22.853980s
spine01     Fresh     yes          1.3.0-cl3u9~1522713084.
b08ca60      2h:42m:42s    2h:15m:36s      2h:15m:36s
21.486093s
spine02     Fresh     yes          1.3.0-cl3u9~1522713084.
b08ca60      2h:42m:55s    2h:15m:37s      2h:15m:37s
6.269588s
Cumulus@ts:~$ netq check agents
Checked nodes: 12, Rotten nodes: 0
```

# Command Line Basics

This section describes the core structure and behavior of the NetQ CLI. It includes the following:

- Command Line Structure (see page )
- Command Syntax (see page )
- Command Output (see page )
- Command Prompts (see page )
- Command Completion (see page )
- Command Help (see page )
- Command History (see page 16)

## Command Line Structure

The Cumulus NetQ command line has a flat structure as opposed to a modal structure. This means that all commands can be run from the primary prompt instead of only in a specific mode. For example, some command lines require the administrator to switch between a configuration mode and an operation mode. Configuration commands can only be run in the configuration mode and operational commands can only be run in operation mode. This structure requires the administrator to switch between modes to run commands which can be tedious and time consuming. Cumulus NetQ command line enables the administrator to run all of its commands at the same level.

## Command Syntax

NetQ CLI commands all begin with `netq`. Their basic syntax is as follows:

```
netq [<hostname>] (check|show) <object> <options>
netq trace <options>
netq resolve
netq config (agent|notifier) <action> [<options>] [vrf <vrf>]
```

| Symbols | Meaning |
|---|---|
| Parentheses ( ) | Enter one of the objects or keywords |
| Square brackets [ ] | Optional parameter; enter keyword or keyword-value pair as needed |
| Angle brackets < > | Variable value for a keyword or option; required, enter according to your deployment nomenclature |
| Pipe \| | Separates keyword options, also separates value options; enter one keyword and zero or one value |

For example, in the `netq check` command:

- [<hostname>] is an optional parameter with a variable value named *hostname*
- <object> represents a number of possible key words, such as *agents, bgp, clag,* and so forth
- <options> represents a number of possible conditions for the given object, such as *around, vrf,* or *json*

Thus some valid commands are:

- `netq check agents json`
- `netq show bgp`
- `netq agent restart`

## Command Output

The command output presents results in color for many commands. Results with errors are shown in red , and warnings are shown in yellow . Results without errors or warnings are shown in either black or green . VTEPs are shown in blue . A node in the *pretty* output is shown in **bold**, and a router interface is wrapped in angle brackets (< >). To view the output with only black text, run the `netq config del color` command. You can view output with colors again by running `netq config add color`.

All check and show commands are run with a default timeframe of now to one hour ago, unless you specify an approximate time ( `around` keyword) or a range ( `between` keyword). For example, running `netq check bgp` shows the status of BGP over the last hour. Running `netq show bgp around 3h` shows the status of BGP three hours ago. Running `netq show bgp changes between now and 3h` shows changes that have been made to BGP configuration in the past three hours.

## Command Prompts

NetQ code examples use the following prompts:

- `cumulus@switch:~$` Indicates the user *cumulus* is logged in to a switch to run the example command
- `cumulus@ts:~$` Indicates the user *cumulus* is logged in to the Telemetry Server (TS) to run the example command
- `cumulus@host:~$` Indicates the user *cumulus* is logged in to a host to run the example command

The switches and TS must be running the Cumulus Linux operating system (OS) and NetQ. The hosts must be running CentOS, RHEL, or Ubuntu OS and NetQ. Refer to the Install NetQ topic for details.

## Command Completion

As you enter commands, you can get help with the valid keywords or options using the Tab key. For example, using Tab completion with `netq check` displays the possible objects for the command, and returns you to the command prompt to complete the command.

```
cumulus@switch:~$ netq check <<press Tab>>
    agents      :  Netq agent
    bgp         :  BGP info
    clag        :  Cumulus Multi-chassis LAG
    evpn        :  EVPN
    interfaces  :  network interface port
    license     :  License information
```

```
     lnv          :  Lightweight Network Virtualization info
     mtu          :  Link MTU
     ntp          :  NTP
     ospf         :  OSPF info
     sensors      :  Temperature/Fan/PSU sensors
     vlan         :  VLAN
     vxlan        :  VXLAN data path
cumulus@oob-mgmt-server:~$ netq check
```

## Command Help

As you enter commands, you can get help with command syntax by entering *help* at various points within a command entry. For example, to find out what options are available for a BGP check, enter *h elp* after entering a portion of the `netq check` command. In this example, you can see that there are two possible commands related to BGP checks and the display shows the options available for each.

```
cumulus@ts:~$ netq check bgp help
Commands:
   netq example check bgp
   netq check bgp [vrf <vrf>] [around <text-time>] [json]
Keywords:
   check bgp                    : Check BGP Status Across the Fabric
cumulus@ts:~$
```

To see an exhaustive list of commands, run:

```
cumulus@switch:~$ netq help list verbose
```

## Command History

The CLI stores commands issued within a session, which enables you to review and rerun commands that have already been run. At the command prompt, press the **Up Arrow** and **Down Arrow** keys to move back and forth through the list of commands previously entered. When you have found a given command, you can run the command by pressing **Enter**, just as you would if you had entered it manually. Optionally you can modify the command before you run it.

# Command Categories

While the CLI has a flat structure, the commands can be conceptually grouped into four functional categories:

- Check and Show Commands (see page )
- Agent and Notifier Commands (see page )
- Trace Command (see page )
- Resolve Command (see page )

## Check and Show Commands

The `check` and `show` commands enable the network administrator to view the current and historical state of the network by manually monitoring for errors and misconfigurations in the network. Check commands run validation checks against various components and configured protocols and services to determine the network is operating as expected. Show commands present details about the current or historical configuration and status of the various component, protocol or service.

Validation checks can be performed for the following:

- agents: NetQ Agents operation on all switches and hosts
- bgp: BGP (Border Gateway Protocol) operation across the network fabric
- clag: Cumulus Multi-chassis LAG (link aggregation) operation
- evpn: EVPN (Ethernet Virtual Private Network) operation
- interfaces: network interface port operation
- license: License status
- lnv: Lightweight Network Virtualization operation
- mtu: Link MTU (maximun transmission unit) consistency across fabric
- ntp: NTP (Network Time Protocol) operation
- ospf: OSPF (Open Shortest Path First) operation
- sensors: Temperature/Fan/PSU sensor operation
- vlan: VLAN (Virtual Local Area Network) operation
- vxlan: VXLAN (Virtual Extensible LAN) data path operation

The configuration and status can be shown for the following:

- agents: NetQ Agents status on switches and hosts
- bgp: BGP status across the network fabric
- change: For a given component, protocol or service, lists changes over time frame
- clag: CLAG status
- docker: Docker Swarm, container and network status
- evpn: EVPN status
- interfaces: network interface port status
- inventory: hardware component information
- ip: IPv4 status
- ipv6: IPv6 status
- kubernetes: Kubernetes cluster, daemon, pod, node, service and replication status
- lldp: LLDP status
- lnv: Lightweight Network Virtualization status
- macs: MAC table or address information
- ntp: NTP status
- ospf: OSPF status
- sensors: Temperature/Fan/PSU sensor status
- services: System services status

- vlan: VLAN status
- vxlan: VXLAN data path status

The commands take the form of `netq [<hostname>] (check|show) <object> <options>`, where the object is one of the components, protocols, or services listed here and the options vary according to the object. The commands can be restricted from checking or showing the information for *all* devices to checking or showing information for a *selected* device using the *hostname* keyword.

## Agent and Notifier Commands

The agent and notifier commands enable the network administrator to configure individual NetQ Agents and the NetQ Notifier on the TS. Refer to the Cumulus NetQ Primer and Configure Optional NetQ Capabilities topics for details about these NetQ components.

The agent configuration commands enable you to add and remove agents from switches and hosts, start and stop agent operations, add and remove docker and kubernetes container monitoring, add or remove sensors, debug the agent, and add or remove FRR (Free Range Routing). Commands apply to one agent at a time, and are run from the switch or host where the NetQ Agent resides.

The agent configuration commands include:

```
netq config (start|stop|status|restart) agent
netq config add agent docker-monitor [poll-period <text-duration-
period>]
netq config del agent docker-monitor
netq config add agent kubernetes-monitor [poll-period <text-duration-
period>]
netq config del agent kubernetes-monitor
netq config (add|del) agent (stats|sensors)
netq config add agent loglevel [debug|info|warning|error]
netq config add agent frr-monitor [<text-frr-docker-name>]
netq config del agent (loglevel|frr-monitor)
netq config show agent [kubernetes-monitor|docker-
monitor|loglevel|stats|sensors|frr-monitor] [json]
```

The notifier configuration commands enable you to start and stop the NetQ Notifier, add and remove notification application integrations, debug the notifier operation, and view its configuration. The commands must be run on the Telemetry Server where the NetQ Notifier resides.

The notifier configuration commands include:

```
netq config ts add notifier integration slack <text-integration-name>
webhook <text-webhook-url>
    [severity info | severity warning | severity error | severity
debug | severity info] [tag <text-slack-tag>]
netq config ts add notifier integration pagerduty <text-integration-
name> api-access-key <text-api-access-key> api-integration-key
    <text-api-integration-key> [severity info | severity warning |
severity error | severity debug | severity info]
netq config ts add notifier integration pagerduty <text-integration-
name> api-integration-key <text-api-integration-key>
```

```
    api-access-key <text-api-access-key> [severity info | severity
warning | severity error | severity debug | severity info]
netq config ts add notifier filter <text-filter-name> [before <text-
filter-name-anchor> | after <text-filter-name-anchor>]
    [rule <text-rule-key> <text-rule-value>] [output <text-
integration-name-anchor>]
netq config ts add notifier loglevel [debug|info|warning|error]
netq config ts del notifier loglevel
netq config ts del notifier integration (slack|pagerduty) <text-
integration-name-anchor>
netq config ts del notifier filter <text-filter-name-anchor>
netq config ts (start|stop|status|restart) notifier
netq config ts show notifier [json]
```

Notice that the `netq config ts add notifier integration pagerduty` is presented twice here because the `api-access-key` and the `api-integration-key` are not order dependent. Either can be entered first. The rest of the syntax is the same.

## Trace Command

The `trace` command enables the network administrator to view the available paths between two nodes on the network currently and at a time in the past. You can base the trace on MAC or IP addresses, perform the trace in only one direction or both, and view the output in one of three formats (*json, pretty,* and *detail*). JSON output provides the output in a JSON file format for ease of importing to other applications or software. Pretty output lines up the paths in a pseudo-graphical manner to help visualize multiple paths. Detail output is useful for traces with higher hop counts where the pretty output wraps lines, making it harder to interpret the results. The detail output displays a table with a row for each path. The trace command also has a detailed usage example for reference.

The trace command syntax is:

```
netq trace <mac> [vlan <1-4096>] from (<src-hostname>|<ip-src>) [vrf
<vrf>] [around <text-time>] [bidir] [json|detail|pretty] [debug]
netq trace <ip> from (<src-hostname>|<ip-src>) [vrf <vrf>] [around
<text-time>] [bidir] [json|detail|pretty] [debug]
```

**Example**: Running a trace based on the destination IP address, in *pretty* output with a small number of resulting paths

```
cumulus@switch:~# netq trace 192.168.0.11 from Leaf04 pretty
Number of Paths: 2
Number of Paths with Errors: 0
Number of Paths with Warnings: 0
Path MTU: 9202

 Leaf04 uplink-2 -- downlink-5 Spine02 downlink-1 -- uplink-2 Leaf01
<uplink-2>
        uplink-1 -- downlink-5 Spine01 downlink-1 -- uplink-1 Leaf01
<uplink-2>
```

**Example**: Running a trace based on the destination MAC address, in *pretty* output with a larger number of resulting paths

```
cumulus@switch:mgmt-vrf:~# netq trace A0:00:00:00:00:11 vlan 1001
from Server03 detail
Number of Paths: 6
Number of Paths with Errors: 0
Number of Paths with Warnings: 0
Path MTU: 9152

 Server03 bond1.1001 -- swp7 <vlan1001> Leaf02 vni: 34 swp5 -- swp4
Spine03 swp7 -- swp5 vni: 34 Leaf04 swp6 -- swp1.1001 Server03 <swp1.
1001>
                                                       swp4 -- swp4
Spine02 swp7 -- swp4 vni: 34 Leaf04 swp6 -- swp1.1001 Server03 <swp1.
1001>
                                                       swp3 -- swp4
Spine01 swp7 -- swp3 vni: 34 Leaf04 swp6 -- swp1.1001 Server03 <swp1.
1001>
          bond1.1001 -- swp7 <vlan1001> Leaf01 vni: 34 swp5 -- swp3
Spine03 swp7 -- swp5 vni: 34 Leaf04 swp6 -- swp1.1001 Server03 <swp1.
1001>
                                                       swp4 -- swp3
Spine02 swp7 -- swp4 vni: 34 Leaf04 swp6 -- swp1.1001 Server03 <swp1.
1001>
                                                       swp3 -- swp3
Spine01 swp7 -- swp3 vni: 34 Leaf04 swp6 -- swp1.1001 Server03 <swp1.
1001>
```

**Example**: View the detailed usage example for the trace command

```
cumulus@switch:~$ netq example trace

Control Path Trace
==================

Commands
========
   netq trace <mac> [vlan <1-4096>] from (<src-hostname>|<ip-src>)
[vrf <vrf>] [around <text-time>] [bidir] [json|detail|pretty] [debug]
   netq trace <ip> from (<src-hostname>|<ip-src>) [vrf <vrf>] [around
<text-time>] [bidir] [json|detail|pretty] [debug]

Usage
=====
netq trace provides control path tracing (no real packets are sent)
from a specified source to a specified destination. The trace covers
complete
```

```
end-to-end path tracing including bridged, routed and VXLAN overlay
paths. ECMP is supported as well as checking for forwarding loops,
MTU consistency
across all paths, and VLAN consistency across all paths.  Reverse
path trace is also available as an option.

...
```

## Resolve Command

The `resolve` command enables the network administrator to view Cumulus Linux command results with more contextual information and colored highlights. By piping the commands through netq resolve, the output shows hostnames and interfaces in green , for example.

To show routes installed by the kernel, you would run the `ip route show proto kernel` command:

```
cumulus@leaf01:~$ ip route show proto kernel
3.0.2.128/26 dev VlanA-1.103  scope link  src 3.0.2.131
3.0.2.128/26 dev VlanA-1-103-v0  scope link  src 3.0.2.129
3.0.2.192/26 dev VlanA-1.104  scope link  src 3.0.2.195
3.0.2.192/26 dev VlanA-1-104-v0  scope link  src 3.0.2.193
3.0.3.0/26 dev VlanA-1.105  scope link  src 3.0.3.3
3.0.3.0/26 dev VlanA-1-105-v0  scope link  src 3.0.3.1
3.0.3.64/26 dev VlanA-1.106  scope link  src 3.0.3.67
3.0.3.64/26 dev VlanA-1-106-v0  scope link  src 3.0.3.65
169.254.0.8/30 dev peerlink-1.4094  scope link  src 169.254.0.10
192.168.0.0/24 dev eth0  scope link  src 192.168.0.15
```

You can enhance the output to display the node names and interfaces by piping the output through `netq resolve` so the output looks like this:

```
cumulus@leaf01:~$ ip route show proto kernel | netq resolve
10.0.0.0/22 (
multiple:
) dev eth0  scope link  src 10.0.0.165 (
cel-smallxp-13
:
eth0
)
3.0.2.128/26 (
server02
 :
torbond1.103
 ) dev VlanA-1.103  scope link  src 3.0.2.131 (
leaf02
 :
```

```
VlanA-1.103
 )
3.0.2.128/26 (
server02
 :
torbond1.103
 ) dev VlanA-1-103-v0  scope link  src 3.0.2.129 (
leaf02
 :
VlanA-1-103-v0
 )
3.0.2.192/26 (
leaf02
 :
VlanA-1-104-v0
 ) dev VlanA-1.104  scope link  src 3.0.2.195 (
leaf02
 :
VlanA-1.104
 )
3.0.2.192/26 (
leaf02
 :
VlanA-1-104-v0
 ) dev VlanA-1-104-v0  scope link  src 3.0.2.193 (
leaf02
 :
VlanA-1-104-v0
 )
3.0.3.0/26 (
server01
 :
torbond1.105
 ) dev VlanA-1.105  scope link  src 3.0.3.3 (
leaf02
 :
VlanA-1.105
 )
3.0.3.0/26 (
server01
 :
torbond1.105
 ) dev VlanA-1-105-v0  scope link  src 3.0.3.1 (
leaf02
 :
```

```
VlanA-1-105-v0
 )
3.0.3.64/26 (
server02
 :
torbond1.106
 ) dev VlanA-1.106  scope link  src 3.0.3.67 (
leaf02
 :
VlanA-1.106
 )
3.0.3.64/26 (
server02
 :
torbond1.106
 ) dev VlanA-1-106-v0  scope link  src 3.0.3.65 (
leaf01
 :
VlanA-1-106-v0
 )
169.254.0.8/30 (
leaf02
 :
peerlink-1.4094
 ) dev peerlink-1.4094  scope link  src 169.254.0.10 (
leaf02
 :
peerlink-1.4094
 )
192.168.0.0/24 (
server02
 :
eth0
 ) dev eth0  scope link  src 192.168.0.15 (
leaf01
 :
eth0
 )
```

## Detailed Usage Examples

Additional help is available to understand key commands using the examples provided with NetQ. Each example includes details about a command's usage and operation, as well as specific examples to help you monitor and manage your network, and solve issues you may find.

Run any of the example commands to view its detailed information:

```
netq example check bgp
netq example check clag
netq example check mtu
netq example find-duplicates
netq example find-origin
netq example ha-setup
netq example query
netq example regexp
netq example resolve macs
netq example resolve routes
netq example startup
netq example stats
netq example trace
```

**Example**: View Example for Duplicate IP or MAC Address

```
cumulus@switch:~$ netq example find-duplicates
Find Duplicate IP or MAC
========================
Commands
========
    - netq show ip routes [<ipv4>|<ipv4/prefixlen>] [vrf <vrf>]
origin [around <text-time>] [json]
    - netq show ipv6 routes [<ipv6>|<ipv6/prefixlen>] [vrf <vrf>]
origin count [around <text-time>] [json]
    - netq show macs [<mac>] [vlan <0-4096>] origin [around <text-
time>] [json]

Usage
=====
Using the 'origin' option coupled with the 'count' option, its easy
to find duplicate route announcements.

    cumulus@switch:mgmt-vrf:~$ netq show ip routes 3.0.0.0/26 origin
count
    Count of matching routes: 3

The example above shows that the ip route 3.0.0.0/26 has been
announced from three nodes in the network. You can look at which
nodes by issuing the same
command without the count option. JSON output is of course available
for both commands.

...
```

# Command Changes

A number of commands have changed in this release to accommodate the addition of new keywords and options or to simplify their syntax. Additionally, new commands have been added and others have been removed. A summary of those changes is provided here.

## New Commands

The following table summarizes the new commands available with this release.

| | Command | Summary |
|---|---|---|
| 1 | netq config (add\|del) color | Add or remove color from CLI output. Default displays colored output. |
| 2 | netq config (status\|restart) cli | Show whether the CLI daemon is running, or restart the CLI daemon if it is not running |
| 3 | netq [<hostname>] show agents changes [between <text-time> and <text-endtime>] [json] | Show NetQ Agent configuration or status changes within the specified timeframe. When the timeframe is not specified, the default is 1 hour. |
| 4 | netq [<hostname>] show docker swarm cluster [node-name <cluster-node>] [around <text-time>] [json] | Show Docker Swarm container clusters at an earlier point in time |
| 5 | netq <hostname> show docker swarm cluster changes [between <text-time> and <text-endtime>] [json] | Show Docker Swarm container cluster configuration or status changes within the specified timeframe. When the timeframe is not specified, the default is 1 hour. |
| 6 | netq config add agent frr-monitor [<text-frr-docker-name>] | Add Free Range Routing (FRR) monitoring to the switch or host server |
| 7 | netq config ts del notifier integration (slack\|pagerduty) <text-integration-name-anchor> | Remove an event notification integration using its anchor name |
| 8 | netq config ts del notifier filter <text-filter-name-anchor> | Remove an event filter using its anchor name |

## Modified Commands

The following table summarizes the commands that have been changed with this release.

| | Command | What Changed |
|---|---|---|
| 1 | netq check agents [around <text-time>] [json] | Added around keyword-value pair |

| | Command | What Changed |
|---|---|---|
| 2 | netq [<hostname>] show agents [around <text-time>] [json] | Added around keyword-value pair |
| 3 | netq config (add\|del) agent (stats\|sensors) | Added sensors keyword |
| 4 | netq config del agent (loglevel\|frr-monitor) | Added frr-monitor keyword |
| 5 | netq config show agent [kubernetes-monitor\|docker-monitor\|loglevel\|stats\|sensors\|frr-monitor] [json] | Added sensors and frr-monitor keywords |
| 6 | netq config ts add notifier integration slack <text-integration-name> webhook <text-webhook-url> [severity info \| severity warning \| severity error \| severity debug \| severity info] [tag <text-slack-tag>] | Added integration keyword |
| 7 | netq config ts add notifier integration pagerduty <text-integration-name> api-integration-key <text-api-integration-key> api-access-key <text-api-access-key> [severity info \| severity warning \| severity error \| severity debug \| severity info]<br>netq config ts add notifier integration pagerduty <text-integration-name> api-access-key <text-api-access-key> api-integration-key <text-api-integration-key> [severity info \| severity warning \| severity error \| severity debug \| severity info] | Added integration keyword and allowed user-preferred order of api-integration-key and api-access-key keywords |
| 8 | netq config ts add notifier filter <text-filter-name> [before <text-filter-name-anchor> \| after <text-filter-name-anchor>] [rule <text-rule-key> <text-rule-value>] [output <text-integration-name-anchor>] | Combined separate NetQ Notifier commands into single command |
| 9 | netq trace <mac> [vlan <1-4096>] from (<src-hostname>\|<ip-src>) [vrf <vrf>] [around <text-time>] [bidir] [json\|detail\|pretty] [debug] | Added ip-source as alternate for src-hostname. Added bidir as option to perform the trace in both directions. Added detail (tabular) and pretty (tree-like) output options. Added debug keyword. |
| 10 | netq trace <ip> from (<src-hostname>\|<ip-src>) [vrf <vrf>] [around <text-time>] [bidir] [json\|detail\|pretty] [debug] | Added bidir as option to perform the trace in both directions. Added detail (tabular) and pretty (tree-like) doutput options. Added debug keyword. |

## Deprecated commands

The following table summarizes the commands that have been removed and a recommended alternative, if appropriate.

|  | Command | Alternative Command |
|---|---|---|
| 1 | netq config ts show notifier loglevel [json] | netq config ts show notifier [json] |

# NetQ Service Console

The NetQ Telemetry Server provides access to the NetQ Service Console, a graphical user interface (GUI) for NetQ. The Service Console provides a command line interface for running NetQ commands.

> ⓘ The Cumulus NetQ Service Console utilizes elements of Portainer. You can read the Portainer license file here.

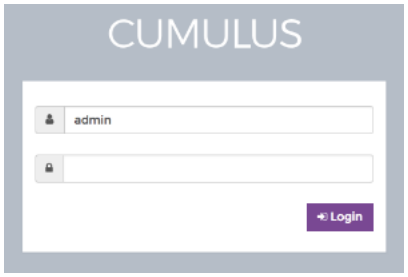
## Contents

This topic describes how to...

## Connect to the Service Console

To connect to the Service Console:

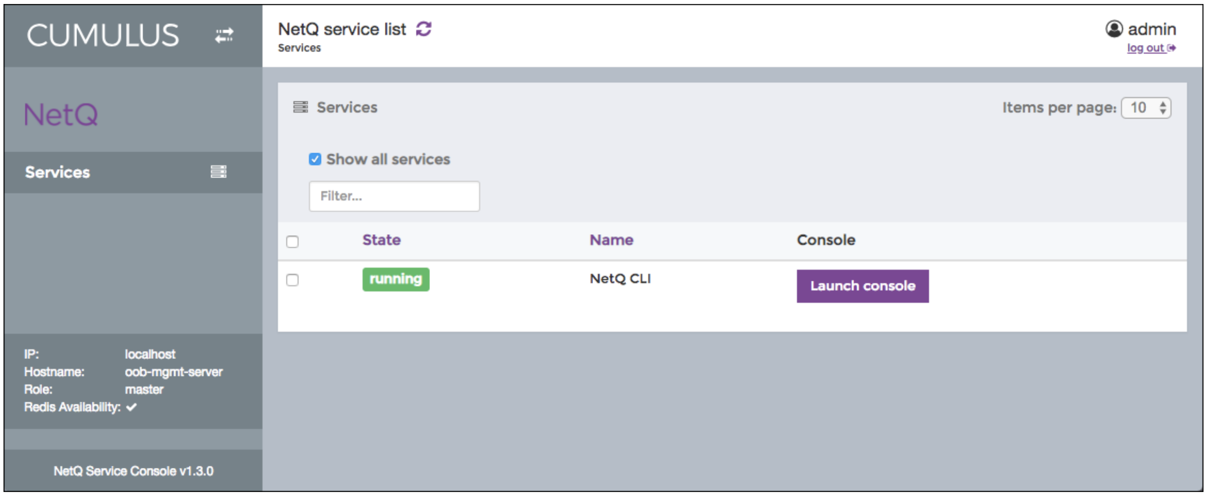
1. Open an Internet browser.
2. In the Address Bar, type <telemetry-server-ip-address:port>; for example, *http://172.28.128.20:9000.* If you don't know or remember the IP address of your Telemetry Server, refer to the Installing NetQ chapter for details. The default port is 9000.



CUMULUS

admin

3. Enter your username and password to open the Service Console.
You can use the same credentials that you use to access the Telemetry Server VM. The Service Console user accounts are managed in the Telemetry Server itself, just like any Linux user account.



NetQ service list
Services

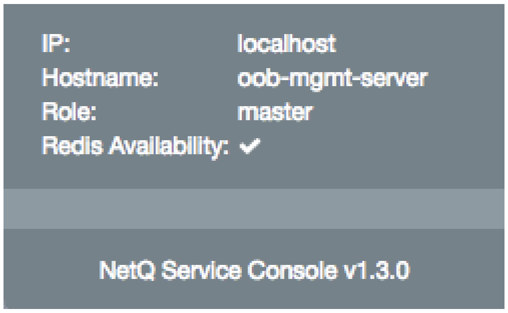
| □ | State | Name | Console |
|---|---|---|---|
| □ | running | NetQ CLI | Launch console |

IP: localhost
Hostname: oob-mgmt-server
Role: master
Redis Availability: ✓

NetQ Service Console v1.3.0

## View Service Console Information

The lower lefthand corner of the Service Console window displays information about the Telemetry Server:



IP: localhost
Hostname: oob-mgmt-server
Role: master
Redis Availability: ✓

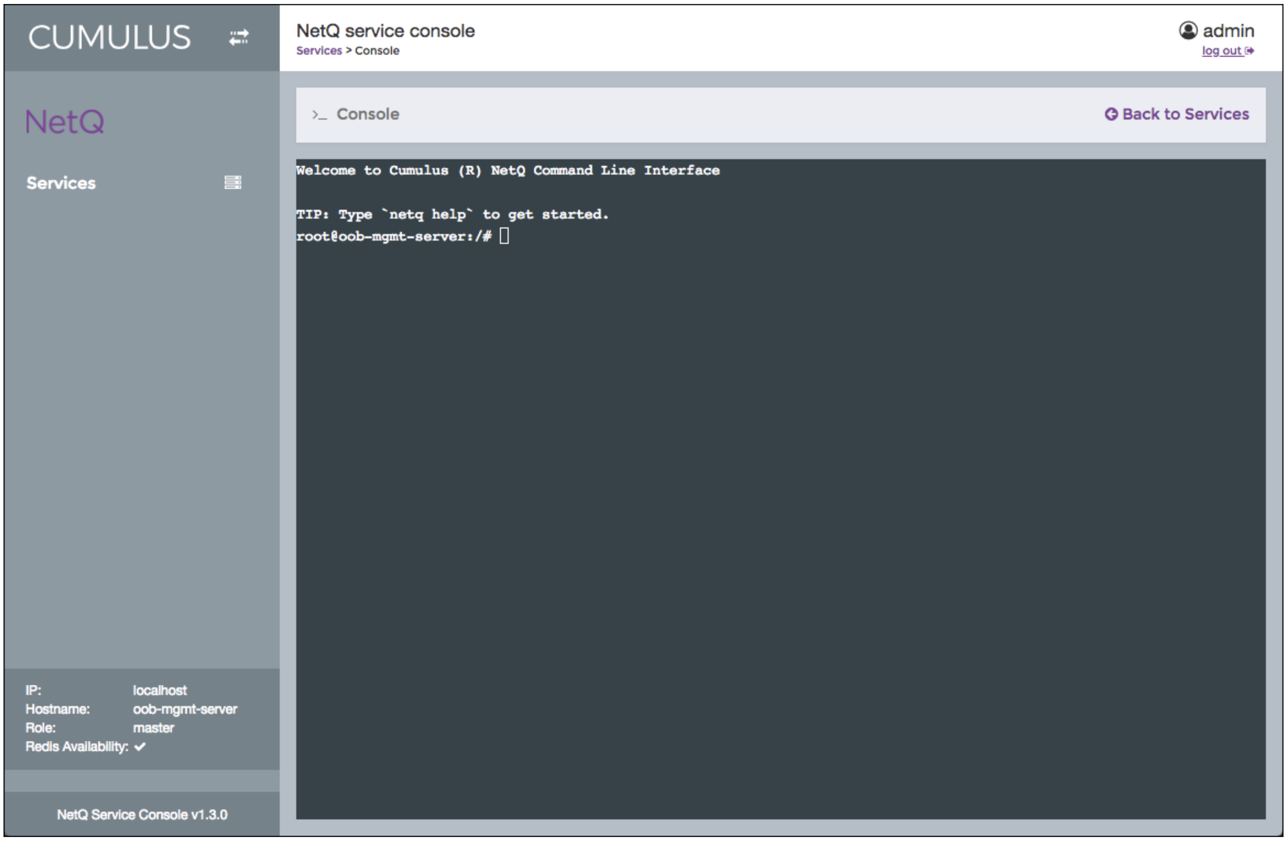
NetQ Service Console v1.3.0

- **IP**: The IP address of the Telemetry Server VM. In the default configuration, the IP field is empty. To have this field display the IP address, edit `/etc/cts/redis/host.conf` and set the `HOST_IP` variable to the Telemetry Server's IP address, then restart the `netq-gui` service with `sudo systemctl restart netq-gui.service`.
- **Hostname**: The hostname of the Telemetry Server VM. The hostname is based on the *%H* environment value in the `systemd` service configuration. If you change the hostname, you should restart the `netq-gui` service so the new hostname displays in the Service Console.
- **Role**: The role that the NetQ database is in, which currently can be *master* or *replica*, if high availability (HA) mode is enabled. If it's not enabled, *master* appears here. If the role is set to *replica*, this indicates that the node is part of an HA cluster, since there is no replica in a non-HA environment.
- **High Availability**: A check mark appears if HA mode is enabled and the current node is the *master* node. This also determines that the master referred to in the role above is also the master for the Redis cluster in HA mode.
- **Redis availability**: Indicates whether or not the Redis database on the Telemetry Server VM is reachable.
- **Version**: Indicates the Service Console version installed. This should match your NetQ version.

## Access the NetQ Command Line

The Service Console runs within the NetQ CLI container. You can use it to connect to the NetQ command line locally within the container. You can also use it to access the container's `/etc/cts/netq` directory to edit or add configuration files under /`config.d`. You cannot use it to connect to the NetQ CLI on a remote system; nor can you access the container's `systemd` services or alter anything else in the container. The filesystem exposed in the console window is actually the container's filesystem.

In the Services window of the console, verify the NetQ CLI **State** is *running*, then click **Launch console**.



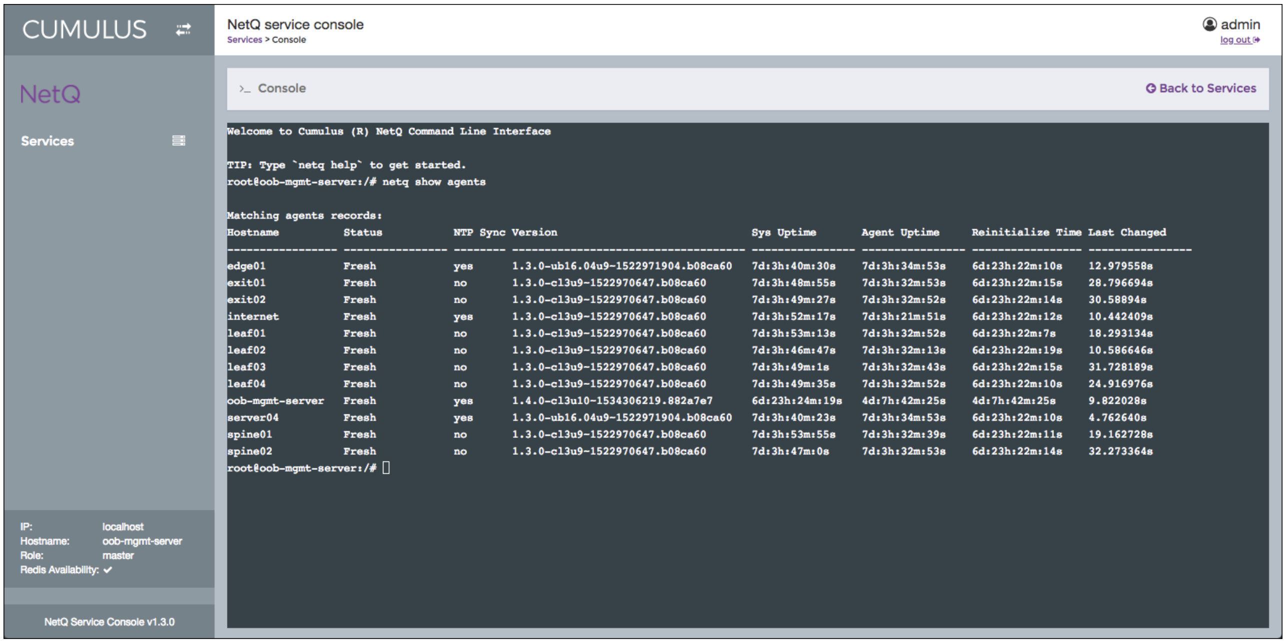
```
Welcome to Cumulus (R) NetQ Command Line Interface

TIP: Type `netq help` to get started.
root@oob-mgmt-server:/# 
```

You are logged in to the Telemetry Server with root user privileges.

## Run NetQ Commands

You can run all NetQ `check` and `show` commands, agent configuration commands, and the `trace` and `resolve` commands from within the console, just as you would if you were logged directly into the network switch or server . Check commands color the output text green to indicate successful results, and red or yellow to indicate errors or warnings.

**Example**: Run `netq show agents`

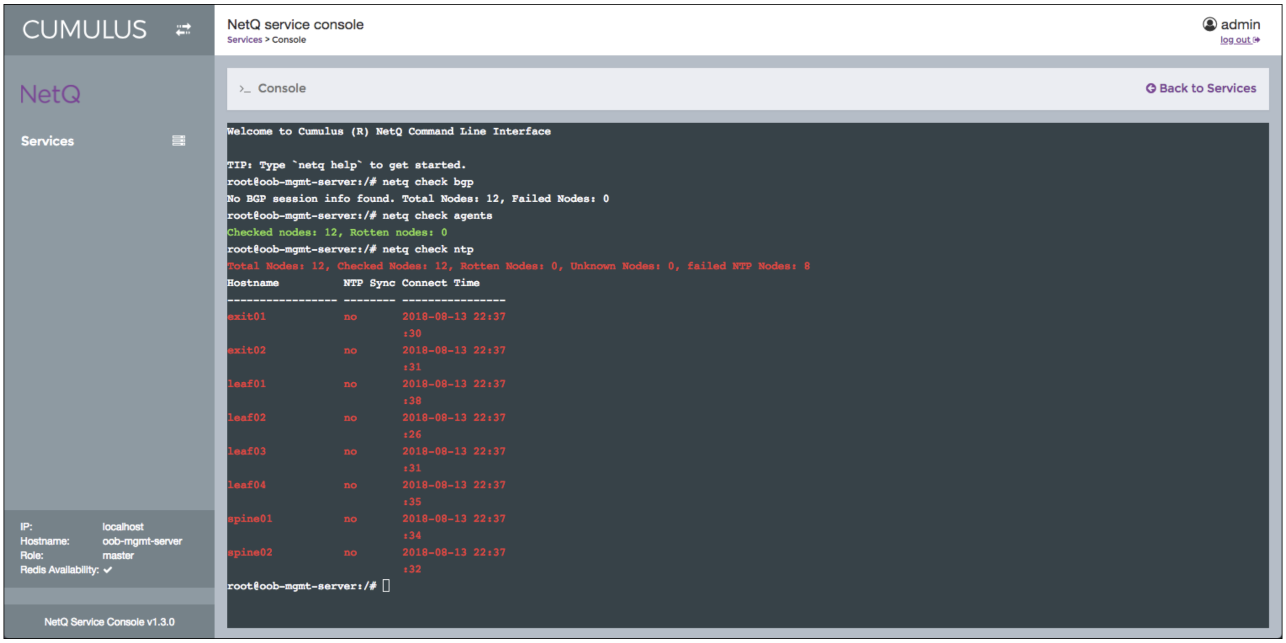


IP: localhost
Hostname: oob-mgmt-server
Role: master
Redis Availability: ✓

```
Welcome to Cumulus (R) NetQ Command Line Interface

TIP: Type `netq help` to get started.
root@oob-mgmt-server:/# netq show agents

Matching agents records:
Hostname          Status           NTP Sync Version                              Sys Uptime       Agent Uptime     Reinitialize Time Last Changed
----------------- ---------------- -------- ------------------------------------ ---------------- ---------------- ----------------- ----------------
edge01            Fresh            yes      1.3.0-ub16.04u9-1522971904.b08ca60   7d:3h:40m:30s    7d:3h:34m:53s    6d:23h:22m:10s    12.979558s
exit01            Fresh            no       1.3.0-cl3u9-1522970647.b08ca60       7d:3h:48m:55s    7d:3h:32m:53s    6d:23h:22m:15s    28.796694s
exit02            Fresh            no       1.3.0-cl3u9-1522970647.b08ca60       7d:3h:49m:27s    7d:3h:32m:52s    6d:23h:22m:14s    30.58894s
internet          Fresh            yes      1.3.0-cl3u9-1522970647.b08ca60       7d:3h:52m:17s    7d:3h:21m:51s    6d:23h:22m:12s    10.442409s
leaf01            Fresh            no       1.3.0-cl3u9-1522970647.b08ca60       7d:3h:53m:13s    7d:3h:32m:52s    6d:23h:22m:7s     18.293134s
leaf02            Fresh            no       1.3.0-cl3u9-1522970647.b08ca60       7d:3h:46m:47s    7d:3h:32m:13s    6d:23h:22m:19s    10.586646s
leaf03            Fresh            no       1.3.0-cl3u9-1522970647.b08ca60       7d:3h:49m:1s     7d:3h:32m:43s    6d:23h:22m:15s    31.728189s
leaf04            Fresh            no       1.3.0-cl3u9-1522970647.b08ca60       7d:3h:49m:35s    7d:3h:32m:52s    6d:23h:22m:10s    24.916976s
oob-mgmt-server   Fresh            yes      1.4.0-cl3u10-1534306219.882a7e7      6d:23h:24m:19s   4d:7h:42m:25s    4d:7h:42m:25s     9.822028s
server04          Fresh            yes      1.3.0-ub16.04u9-1522971904.b08ca60   7d:3h:40m:23s    7d:3h:34m:53s    6d:23h:22m:10s    4.762640s
spine01           Fresh            no       1.3.0-cl3u9-1522970647.b08ca60       7d:3h:53m:55s    7d:3h:32m:39s    6d:23h:22m:11s    19.162728s
spine02           Fresh            no       1.3.0-cl3u9-1522970647.b08ca60       7d:3h:47m:0s     7d:3h:32m:53s    6d:23h:22m:14s    32.273364s
root@oob-mgmt-server:/# 
```

If the output from a given command it too wide for the current console window causing the data rows to wrap over lines, widen the console window by clicking and dragging the right edge of the window and then rerun the command for a cleaner view.

**Example**: Run `netq check bgp, netq check agents` and `netq check ntp`



IP: localhost
Hostname: oob-mgmt-server
Role: master
Redis Availability: ✓

```
Welcome to Cumulus (R) NetQ Command Line Interface

TIP: Type `netq help` to get started.
root@oob-mgmt-server:/# netq check bgp
No BGP session info found. Total Nodes: 12, Failed Nodes: 0
root@oob-mgmt-server:/# netq check agents
Checked nodes: 12, Rotten nodes: 0
root@oob-mgmt-server:/# netq check ntp
Total Nodes: 12, Checked Nodes: 12, Rotten Nodes: 0, Unknown Nodes: 0, failed NTP Nodes: 8
Hostname          NTP Sync Connect Time
----------------- -------- ----------------
exit01            no       2018-08-13 22:37
                           :30
exit02            no       2018-08-13 22:37
                           :31
leaf01            no       2018-08-13 22:37
                           :38
leaf02            no       2018-08-13 22:37
                           :26
leaf03            no       2018-08-13 22:37
                           :31
leaf04            no       2018-08-13 22:37
                           :35
spine01           no       2018-08-13 22:37
                           :34
spine02           no       2018-08-13 22:37
                           :32
root@oob-mgmt-server:/# 
```

Note that in this example, BGP is not configured, so no information was found, NetQ Agents status is all good, and that multiple nodes are not time synchronized (which you would want to fix!).

# Exit the Service Console

When you're finished with the session, click **Back to Services** to close the console window, then click **log out** to close the Service Console.

# Monitor Overall Network Health

NetQ provides the information you need to monitor the health of your network fabric, devices, and interfaces. You are able to easily validate the operation and view the configuration across the entire network from switches to hosts to containers. For example, you can monitor the operation of routing protocols and virtual network configurations, the status of NetQ Agents and hardware components, and the operation and efficiency of interfaces. When issues are present, NetQ makes it easy to identify and resolve them. You can also see when changes have occurred to the network, devices, and interfaces by viewing their operation, configuration, and status at earlier points in time.

## Contents

This topic describes how to...

## Validate Network Health

NetQ `check` commands validate the various elements of your network fabric, looking for inconsistencies in configuration across your fabric, connectivity faults, missing configuration, and so forth, and then and display the results for your assessment. They can be run from any node in the network. Most check commands can be run for a specific device or for the entire network fabric.

### Validate the Network Fabric

You can validate the following network fabric elements:

- BGP and OSPF routing protocols
- VLAN, VXLAN, CLAG, and EVPN virtual constructs
- MTU setting
- NetQ Agents

For example, to determine the status of BGP running on your network:

```
cumulus@switch:~$ netq check bgp
Total Nodes: 15, Failed Nodes: 0, Total Sessions: 16, Failed
Sessions: 0
```

You can see from this output that NetQ has validated the connectivity and configuration of BGP across all of the nodes in the network and found them all to be operating properly. If there were issues with any of the nodes, NetQ would provide information about each node to aid in resolving the issues.

There is a check command for each of the supported routing protocols, virtual constructs, MTU setting and NetQ Agents. They all behave in a similar manner, checking for connectivity, configuration, and other problems, indicating the number of nodes that they have checked and indicating the number that have failed.

Some additional examples—

Validate that EVPN is running correctly on all nodes:

```
cumulus@switch:~$ netq check evpn
Total Nodes: 15, Failed Nodes: 0, Total Sessions: 0, Failed Sessions:
0, Total VNIs: 0
```

Confirm all monitored nodes are running the NetQ Agent:

```
cumulus@switch:~$ netq check agents
Checked nodes: 25, Rotten nodes: 0
```

Validate that all corresponding interface links have matching MTUs:

```
cumulus@switch:~$ netq check mtu
Checked Nodes: 15, Checked Links: 138, Failed Nodes: 0, Failed Links:
0
No MTU Mismatch found
```

Validate that VXLANs are configured and operating properly:

```
cumulus@switch:~$ netq check vxlan
Checked Nodes: 6, Warning Nodes: 0, Failed Nodes: 6
Nodes with error
Hostname         Reason
----------------
----------------------------------------------------------------
exit01           inconsistent replication list for vni
104001
exit02           inconsistent replication list for vni
104001
leaf01           inconsistent replication list for vni
104001
leaf02           inconsistent replication list for vni
104001
leaf03           inconsistent replication list for vni
104001
leaf04           inconsistent replication list for vni
104001
```

> With NetQ 1.4 both asymmetric and symmetric VXLAN configurations are validated with this command.

You can be more granular in your validation as well, using the additional options available for each of the check commands. For example, validate BGP operation for nodes with VRF:

```
cumulus@switch:~$ netq check bgp vrf DataVrf1081
Total Nodes: 25, Failed Nodes: 1, Total Sessions: 52 , Failed
Sessions: 0
```

Each of the check commands provides a starting point for troubleshooting configuration and connectivity issues within your network in real time. They provide an additional option of viewing the network state at an earlier time, using the `around` option.

For example, if you were notified of an issue on your VLANs that appears to have occurred about 10 minutes ago, you could run:

```
cumulus@switch:~$ netq check vlan around 10m
Checked Nodes: 15, Checked Links: 138, Failed Nodes: 0, Failed Links:
0
No VLAN or PVID Mismatch found
```

## Validate Device Status and Configuration

You can validate the following device elements:

- NTP
- Sensors
- License

It is always important to have your devices in time synchronization to ensure configuration and management events can be tracked and correlations can be made between events. To validate time synchronization, run:

```
cumulus@switch:~$ netq check ntp
Total Nodes: 15, Checked Nodes: 15, Rotten Nodes: 0, Unknown Nodes:
0, failed NTP Nodes: 8
Hostname          NTP Sync Connect Time
----------------- -------- -------------------------
exit01            no       2018-09-12 16:30:39
exit02            no       2018-09-12 16:30:45
leaf01            no       2018-09-12 16:30:43
leaf02            no       2018-09-12 16:30:36
leaf03            no       2018-09-12 16:30:36
leaf04            no       2018-09-12 16:30:34
spine01           no       2018-09-12 16:30:44
spine02           no       2018-09-12 16:30:40
```

This example shows eight nodes that are not in time synchronization. You can now continue to investigate these nodes, validating that the NetQ Agents are active, whether an NTP server has become unreachable, and so forth.

Hardware platforms have a number sensors to provide environmental data about the switches. Knowing these are all within range is a good check point for maintenance. For example, if you had a temporary HVAC failure and you are concerned that some of your nodes are beginning to overheat, you can run:

```
cumulus@switch:~$ netq check sensors
Total Nodes: 25, Failed Nodes: 0, Checked Sensors: 221, Failed
Sensors: 0
```

You can also check for any nodes that have invalid licenses without going to each node. Because switches do not operate correctly without a valid license you might want to verify that your Cumulus Linux licenses on a regular basis:

```
cumulus@switch:~$ netq check license
Total Nodes: 15, Failed Nodes: 0, Checked Licenses: 10, Failed
Licenses: 0
```

This command checks every node, meaning every switch and host in the network. Hosts do not require a Cumulus Linux license, so the number of licenses checked is likely to be smaller than the total number of nodes checked.

## Validate Interface Status and Configuration

As with other netq check commands, you can validate the proper operation of your interfaces across the network:

```
cumulus@switch:~$ netq check interfaces
Checked Nodes: 15, Failed Nodes: 8
Checked Ports: 118, Failed Ports: 8, Unverified Ports: 94
Hostname          Interface                 Peer Hostname     Peer
Interface            Message
----------------- ------------------------- -----------------
------------------------- -----------------------------------
leaf01            swp1                      server01          eth1
                  Autoneg mismatch (off, on)
leaf02            swp2                      server02          eth2
                  Autoneg mismatch (off, on)
leaf03            swp1                      server03          eth1
                  Autoneg mismatch (off, on)
leaf04            swp2                      server04          eth2
                  Autoneg mismatch (off, on)
server01          eth1                      leaf01            swp1
                  Autoneg mismatch (on, off)
```

```
server02          eth2                      leaf02            swp2
                  Autoneg mismatch (on, off)
server03          eth1                      leaf03            swp1
                  Autoneg mismatch (on, off)
server04          eth2                      leaf04            swp2
                  Autoneg mismatch (on, off)
```

When failures are seen, additional information is provided to start your investigation. In this example, some reconfiguration is required for auto-negotiation with peer interfaces.

## View Network Details

The `netq show` commands display a wide variety of content about the network and its various elements. You can show content for the following:

```
cumulus@switch:~$ netq show
    agents      :  Netq agent
    bgp         :  BGP info
    changes     :  How this infomation has changed with time (default
'1h')
    clag        :  Cumulus Multi-chassis LAG
    docker      :  Docker Info
    evpn        :  EVPN
    interfaces  :  network interface port
    inventory   :  Inventory information
    ip          :  IPv4 related info
    ipv6        :  IPv6 related info
    kubernetes  :  Kubernetes Information
    lldp        :  LLDP based neighbor info
    lnv         :  Lightweight Network Virtualization info
    macs        :  Mac table or MAC address info
    ntp         :  NTP
    ospf        :  OSPF info
    sensors     :  Temperature/Fan/PSU sensors
    services    :  System services
    vlan        :  VLAN
    vxlan       :  VXLAN data path
```

For example, to validate the the status of the NetQ agents running in the fabric, run `netq show agents`. A *Fresh* status indicates the Agent is running as expected. The Agent sends a heartbeat every 30 seconds, and if three consecutive heartbeats are missed, its status changes to *Rotten*.

```
cumulus@leaf01:~$ netq show agents

Node             Status    Sys Uptime    Agent Uptime
---------------  --------  ------------  --------------
leaf01           Fresh     2h ago        2h ago
leaf02           Fresh     2h ago        2h ago
```

```
leaf03           Fresh     2h ago        2h ago
leaf04           Fresh     2h ago        2h ago
oob-mgmt-server  Fresh     2h ago        2h ago
server01         Fresh     2h ago        2h ago
server02         Fresh     2h ago        2h ago
server03         Fresh     2h ago        2h ago
server04         Fresh     2h ago        2h ago
spine01          Fresh     2h ago        2h ago
spine02          Fresh     2h ago        2h ago
```

Some additional examples--

View the status of BGP:

```
cumulus@switch:~$ netq show bgp
Matching bgp records:
Hostname          Neighbor                     VRF             ASN
    Peer ASN   PfxRx        Last Changed
----------------- ---------------------------- ---------------
---------- ---------- ------------ -------------------------
exit01            swp44
(internet)              vrf1            65041      25253      2/-
/-        5d:1h:8m:59s
exit01            swp51
(spine01)               default         65041      65020      8/-
/42       5d:1h:8m:59s
exit01            swp52
(spine02)               default         65041      65020      8/-
/42       5d:1h:8m:58s
exit02            swp44
(internet)              vrf1            65042      25253      2/-
/-        5d:1h:9m:3s
exit02            swp51
(spine01)               default         65042      65020      8/-
/42       5d:1h:9m:4s
exit02            swp52
(spine02)               default         65042      65020      8/-
/42       5d:1h:9m:3s
internet          swp1
(exit01)                 default         25253      65041      0/-
/-        5d:1h:8m:58s
internet          swp2
(exit02)                 default         25253      65042      0/-
/-        5d:1h:9m:3s
leaf01            swp51
(spine01)               default         65011      65020      7/-
/24       5d:1h:9m:0s
leaf01            swp52
(spine02)               default         65011      65020      7/-
/24       5d:1h:8m:59s
```

```
leaf02            swp51
(spine01)               default             65012      65020      8/-
/24       5d:1h:9m:0s
leaf02            swp52
(spine02)               default             65012      65020      8/-
/24       5d:1h:8m:59s
leaf03            swp51
(spine01)               default             65013      65020      7/-
/24       5d:1h:9m:0s
leaf03            swp52
(spine02)               default             65013      65020      7/-
/24       5d:1h:8m:59s
leaf04            swp51
(spine01)               default             65014      65020      8/-
/24       5d:1h:9m:0s
leaf04            swp52
(spine02)               default             65014      65020      8/-
/24       5d:1h:8m:59s
spine01           swp1
(leaf01)                 default             65020      65011      2/-
/10       5d:1h:9m:0s
spine01           swp2
(leaf02)                 default             65020      65012      2/-
/10       5d:1h:9m:0s
spine01           swp29
(exit02)                default             65020      65042      1/-
/2        5d:1h:9m:4s
spine01           swp3
(leaf03)                 default             65020      65013      2/-
/10       5d:1h:9m:0s
spine01           swp30
(exit01)                default             65020      65041      1/-
/2        5d:1h:8m:59s
spine01           swp4
(leaf04)                 default             65020      65014      2/-
/10       5d:1h:9m:0s
spine02           swp1
(leaf01)                 default             65020      65011      2/-
/10       5d:1h:8m:59s
spine02           swp2
(leaf02)                 default             65020      65012      2/-
/10       5d:1h:8m:59s
spine02           swp29
(exit02)                default             65020      65042      1/-
/2        5d:1h:9m:4s
spine02           swp3
(leaf03)                 default             65020      65013      2/-
/10       5d:1h:8m:59s
spine02           swp30
(exit01)                default             65020      65041      1/-
/2        5d:1h:8m:58s
```

```
spine02           swp4
(leaf04)                 default         65020      65014      2/-
/10       5d:1h:8m:58s
```

View the status of your VLANs:

```
cumulus@switch:~$ netq show vlan
Matching vlan records:
Hostname          VLANs                     SVIs
Last Changed
----------------- ------------------------- -------------------------
-------------------------
exit01            4001                      4001
4d:20h:10m:21s
exit02            4001                      4001
4d:20h:9m:57s
leaf01            1,13,24,4001              13 24
4001                4d:21h:3m:21s
leaf02            1,13,24,4001              13 24
4001                4d:20h:16m:42s
leaf03            1,13,24,4001              13 24
4001                4d:20h:15m:52s
leaf04            1,13,24,4001              13 24
4001                4d:20h:12m:32s
```

View the status of the hardware sensors:

```
cumulus@switch:~$ netq show sensors all
Matching sensors records:
Hostname          Name            Description
State      Message                             Last Changed
----------------- --------------- -----------------------------------
---------- -----------------------------------
-------------------------
exit01            fan1            fan tray 1, fan
1                   ok                                               4d:
20h:11m:54s
exit01            fan2            fan tray 1, fan
2                   ok                                               4d:
20h:11m:54s
exit01            fan3            fan tray 2, fan
1                   ok                                               4d:
20h:11m:54s
exit01            fan4            fan tray 2, fan
2                   ok                                               4d:
20h:11m:54s
exit01            fan5            fan tray 3, fan
1                   ok                                               4d:
20h:11m:54s
```

```
exit01            fan6            fan tray 3, fan
2                   ok                                             4d:
20h:11m:54s
exit01            psu1fan1        psu1
fan                            ok
        4d:20h:11m:54s
exit01            psu2fan1        psu2
fan                            ok
        4d:20h:11m:54s
exit01            temp1           board sensor near
cpu               ok                                             4d:
20h:11m:54s
exit01            temp2           board sensor near virtual
switch    ok                                             4d:20h:11m:
54s
exit01            temp3           board sensor at front left
corner   ok                                             4d:20h:11m:54s
exit01            temp5           board sensor near
fan               ok                                             4d:
20h:11m:54s
exit02            fan1            fan tray 1, fan
1                   ok                                             4d:
20h:11m:30s
exit02            fan2            fan tray 1, fan
2                   ok                                             4d:
20h:11m:30s
exit02            fan3            fan tray 2, fan
1                   ok                                             4d:
20h:11m:30s
exit02            fan4            fan tray 2, fan
2                   ok                                             4d:
20h:11m:30s
exit02            fan5            fan tray 3, fan
1                   ok                                             4d:
20h:11m:30s
exit02            fan6            fan tray 3, fan
2                   ok                                             4d:
20h:11m:30s
exit02            psu1fan1        psu1
fan                            ok
        4d:20h:11m:30s
exit02            psu2fan1        psu2
fan                            ok
        4d:20h:11m:30s
exit02            temp4           board sensor at front right
corner  ok                                             4d:20h:11m:30s
internet          fan1            fan tray 1, fan
1                   ok                                             5d:
1h:13m:12s
internet          fan2            fan tray 1, fan
2                   ok                                             5d:
1h:13m:12s
```

```
internet           fan3            fan tray 2, fan
1                    ok                                               5d:
1h:13m:12s
...
```

# Monitor Switch Hardware and Software

With NetQ, a network administrator can monitor both the switch hardware and its operating system for misconfigurations or misbehaving services. NetQ provides the ability to:

- Validate configurations
- Validate service operations
- Identify inventory

It helps answer questions such as:

- What switches do I have in the network?
- Are all of my services running?
- Are all switches licensed correctly?
- Do all switches have NetQ agents running?

NetQ uses LLDP (Link Layer Discovery Protocol) to collect port information. NetQ can also identify peer ports connected to DACs ( Direct Attached Cables) and AOCs (Active Optical Cables) without using LLDP, even if the link is not UP.

## Contents

This topic describes how to...

# Monitor Switch and Host Hardware Information

NetQ enables you to view either a summary or details about key components on your switch or host, including the motherboard, ASIC, microprocessor, disk and memory information. The `netq show inventory` command is used to view the information for a single device or for all of your devices at once, depending on what you want to see.

The syntax for this command is:

```
netq [<hostname>] show inventory brief [json]
netq [<hostname>] show inventory asic [vendor <asic-vendor>| model
<asic-model>| model-id <asic-model-id>] [json]
netq [<hostname>] show inventory board [vendor <board-vendor>|model
<board-model>] [json]
netq [<hostname>] show inventory cpu [arch <cpu-arch>] [json]
netq [<hostname>] show inventory disk [name <disk-name>|transport
<disk-transport>| vendor <disk-vendor>] [json]
netq [<hostname>] show inventory memory [type <memory-type>|vendor
<memory-vendor>] [json]
```

> ⓘ The keyword values for the `model`, `disk`, `arch`, `name`, `transport`, and `type` keywords are specific to your deployment. For example, if you have devices with CPU architectures of only one type, say Intel x86, then that is the only option available for the `cpu-arch` keyword value. If you have multiple CPU architectures, say you also have ARMv7, then that would also be an option for you.

## View Information about the ASIC on a Switch

You can view the vendor, model, model identifier, core bandwidth capability, and ports of the ASIC installed on your switch motherboard. This example shows all of these for all devices.

```
cumulus@switch:~$ netq show inventory asic
Matching inventory records:
Hostname          Vendor               Model
Model ID                  Core BW        Ports
----------------- -------------------- ------------------------------
------------------------ -------------
-----------------------------------
dell-z9100-05     Broadcom             Tomahawk
BCM56960                  2.0T           32 x 100G-QSFP28
mlx-2100-05       Mellanox             Spectrum
MT52132                   N/A            16 x 100G-QSFP28
mlx-2410a1-05     Mellanox             Spectrum
MT52132                   N/A            48 x 25G-SFP28 & 8 x 100G-
QSFP28
```

```
mlx-2700-11       Mellanox             Spectrum
MT52132                   N/A            32 x 100G-QSFP28
qct-ix1-08        Broadcom             Tomahawk
BCM56960                  2.0T           32 x 100G-QSFP28
qct-ix7-04        Broadcom             Trident3
BCM56870                  N/A            32 x 100G-QSFP28
qct-ix7-04        N/A                  N/A
N/A                       N/A            N/A
st1-l1            Broadcom             Trident2
BCM56854                  720G           48 x 10G-SFP+ & 6 x 40G-QSFP+
st1-l2            Broadcom             Trident2
BCM56854                  720G           48 x 10G-SFP+ & 6 x 40G-QSFP+
st1-l3            Broadcom             Trident2
BCM56854                  720G           48 x 10G-SFP+ & 6 x 40G-QSFP+
st1-s1            Broadcom             Trident2
BCM56850                  960G           32 x 40G-QSFP+
st1-s2            Broadcom             Trident2
BCM56850                  960G           32 x 40G-QSFP+
```

You can filter the results of the command to view devices with a particular characteristic. This example shows all devices that use a Broadcom ASIC.

```
cumulus@netq-ts:~$ netq show inventory asic vendor Broadcom
Matching inventory records:
Hostname          Vendor               Model
Model ID                  Core BW        Ports
----------------- -------------------- ------------------------------
------------------------- --------------
-----------------------------------
dell-z9100-05     Broadcom             Tomahawk
BCM56960                  2.0T           32 x 100G-QSFP28
qct-ix1-08        Broadcom             Tomahawk
BCM56960                  2.0T           32 x 100G-QSFP28
qct-ix7-04        Broadcom             Trident3
BCM56870                  N/A            32 x 100G-QSFP28
st1-l1            Broadcom             Trident2
BCM56854                  720G           48 x 10G-SFP+ & 6 x 40G-QSFP+
st1-l2            Broadcom             Trident2
BCM56854                  720G           48 x 10G-SFP+ & 6 x 40G-QSFP+
st1-l3            Broadcom             Trident2
BCM56854                  720G           48 x 10G-SFP+ & 6 x 40G-QSFP+
st1-s1            Broadcom             Trident2
BCM56850                  960G           32 x 40G-QSFP+
st1-s2            Broadcom             Trident2
BCM56850                  960G           32 x 40G-QSFP+
```

You can filter the results of the command view the ASIC information for a particular switch. This example shows the ASIC information for *st1-11* switch.

```
cumulus@switch:~$ netq leaf02 show inventory asic
Matching inventory records:
Hostname          Vendor               Model
Model ID                  Core BW        Ports
----------------- -------------------- ------------------------------
------------------------- --------------
-----------------------------------
st1-l1            Broadcom             Trident2
BCM56854                  720G           48 x 10G-SFP+ & 6 x 40G-QSFP+
```

## View Information about the Motherboard in a Switch

You can view the vendor, model, base MAC address, serial number, part number, revision, and manufacturing date for a switch motherboard on a single device or on all devices. This example shows all of the motherboard data for all devices.

```
cumulus@switch:~$ netq show inventory board
Matching inventory records:
Hostname          Vendor               Model
Base MAC           Serial No                 Part
No          Rev    Mfg Date
----------------- -------------------- ------------------------------
------------------ ------------------------- ---------------- ------
----------
dell-z9100-05     DELL                 Z9100-
ON                       4C:76:25:E7:42:
C0  CN03GT5N779315C20001      03GT5N           A00    12/04/2015
mlx-2100-05       Penguin              Arctica
1600cs                 7C:FE:90:F5:61:
C0  MT1623X10078              MSN2100-CB2FO    N/A    06/09/2016
mlx-2410a1-
05     Mellanox             SN2410                         EC:0D:9A:
4E:55:C0  MT1734X00067              MSN2410-CB2F_QP3 N/A    08/24/2017
mlx-2700-11       Penguin              Arctica
3200cs                 44:38:39:00:AB:
80  MT1604X21036              MSN2700-CS2FO    N/A    01/31/2016
qct-ix1-08        QCT                  QuantaMesh BMS T7032-
IX1       54:AB:3A:78:69:
51  QTFC07623002C             1IX1UZZ0ST6      H3B    05/30/2016
qct-ix7-
04        QCT                   IX7                            D8:C4:
97:62:37:65  QTFCUW821000A             1IX7UZZ0ST5      B3D    05/07
/2018
qct-ix7-04        QCT                  T7032-
IX7                      D8:C4:97:62:37:
65  QTFCUW821000A             1IX7UZZ0ST5      B3D    05/07/2018
st1-l1            CELESTICA            Arctica
4806xp                 00:E0:EC:27:71:
37  D2060B2F044919GD000011    R0854-F1004-01   Redsto 09/20/2014
```

```
                                                              ne-XP
st1-l2            CELESTICA            Arctica
4806xp                 00:E0:EC:27:6B:
3A  D2060B2F044919GD000060    R0854-F1004-01   Redsto 09/20/2014

                                                              ne-XP
st1-l3            Penguin              Arctica
4806xp                 44:38:39:00:70:49  N/A                      N
/A              N/A    N/A
st1-s1            Dell                 S6000-
ON                       44:38:39:00:80:00  N
/A                       N/A              N/A    N/A
st1-s2            Dell                 S6000-
ON                       44:38:39:00:80:81  N
/A                       N/A              N/A    N/A
```

You can filter the results of the command to capture only those devices with a particular motherboard vendor. This example shows only the devices with *Celestica* motherboards.

```
cumulus@switch:~$ netq show inventory board vendor celestica
Matching inventory records:
Hostname          Vendor               Model
Base MAC           Serial No                 Part
No          Rev    Mfg Date
----------------- -------------------- ------------------------------
------------------ ------------------------- ---------------- ------
----------
st1-l1            CELESTICA            Arctica
4806xp                 00:E0:EC:27:71:
37  D2060B2F044919GD000011    R0854-F1004-01   Redsto 09/20/2014

                                                              ne-XP
st1-l2            CELESTICA            Arctica
4806xp                 00:E0:EC:27:6B:
3A  D2060B2F044919GD000060    R0854-F1004-01   Redsto 09/20/2014

                                                              ne-XP
```

You can filter the results of the command to view the model for a particular switch. This example shows the motherboard vendor for the *st1-s1* switch.

```
cumulus@switch:~$ netq st1-s1 show inventory board
Matching inventory records:
Hostname          Vendor               Model
Base MAC           Serial No                 Part
No          Rev    Mfg Date
```

```
----------------- -------------------- ------------------------------
------------------ ------------------------- ---------------- ------
----------
st1-s1            Dell                 S6000-
ON                       44:38:39:00:80:00  N
/A                       N/A              N/A    N/A
```

## View Information about the CPU on a Switch

You can view the architecture, model, operating frequency, and the number of cores for the CPU on a single device or for all devices. This example shows these CPU characteristics for all devices.

```
cumulus@nswitch:~$ netq show inventory cpu
Matching inventory records:
Hostname          Arch     Model                          Freq
Cores
----------------- -------- ------------------------------ ----------
-----
dell-z9100-05     x86_64   Intel(R) Atom(TM) C2538        2.40GHz    4
mlx-2100-05       x86_64   Intel(R) Atom(TM) C2558        2.40GHz    4
mlx-2410a1-05     x86_64   Intel(R) Celeron(R)  1047UE    1.40GHz    2
mlx-2700-11       x86_64   Intel(R) Celeron(R)  1047UE    1.40GHz    2
qct-ix1-08        x86_64   Intel(R) Atom(TM) C2558        2.40GHz    4
qct-ix7-04        x86_64   Intel(R) Atom(TM) C2558        2.40GHz    4
st1-l1            x86_64   Intel(R) Atom(TM) C2538        2.41GHz    4
st1-l2            x86_64   Intel(R) Atom(TM) C2538        2.41GHz    4
st1-l3            x86_64   Intel(R) Atom(TM) C2538        2.40GHz    4
st1-s1            x86_64   Intel(R) Atom(TM)  S1220       1.60GHz    4
st1-s2            x86_64   Intel(R) Atom(TM)  S1220       1.60GHz    4
```

You can filter the results of the command to view which switches employ a particular CPU architecture using the *arch* keyword. This example shows how to determine which architectures are deployed in your network, and then shows all devices with an *x86_64* architecture.

```
cumulus@switch:~$ netq show inventory cpu arch
    x86_64  :  CPU Architecture

cumulus@switch:~$ netq show inventory cpu arch x86_64
Matching inventory records:
Hostname          Arch     Model                          Freq       C
ores
----------------- -------- ------------------------------ ----------
-----
leaf01            x86_64   Intel Core i7 9xx (Nehalem Cla N/A        1
                           ss Core i7)
leaf02            x86_64   Intel Core i7 9xx (Nehalem Cla N/A        1
                           ss Core i7)
leaf03            x86_64   Intel Core i7 9xx (Nehalem Cla N/A        1
                           ss Core i7)
leaf04            x86_64   Intel Core i7 9xx (Nehalem Cla N/A        1
                           ss Core i7)
oob-mgmt-server   x86_64   Intel Core i7 9xx (Nehalem Cla N/A        1
                           ss Core i7)
server01          x86_64   Intel Core i7 9xx (Nehalem Cla N/A        1
                           ss Core i7)
server02          x86_64   Intel Core i7 9xx (Nehalem Cla N/A        1
                           ss Core i7)
server03          x86_64   Intel Core i7 9xx (Nehalem Cla N/A        1
                           ss Core i7)
server04          x86_64   Intel Core i7 9xx (Nehalem Cla N/A        1
                           ss Core i7)
spine01           x86_64   Intel Core i7 9xx (Nehalem Cla N/A        1
                           ss Core i7)
spine02           x86_64   Intel Core i7 9xx (Nehalem Cla N/A        1
                           ss Core i7)
```

You can filter the results to view CPU information for a single switch, as shown here for *server02*.

```
cumulus@switch:~$ netq server02 show inventory cpu

Matching inventory records:
Hostname          Arch     Model                          Freq       C
ores
----------------- -------- ------------------------------ ----------
-----
server02          x86_64   Intel Core i7 9xx (Nehalem Cla N/A        1
                           ss Core i7)
```

## View Information about the Disk on a Switch

You can view the name or operating system, type, transport, size, vendor, and model of the disk on a single device or all devices. This example shows all of these disk characteristics for all devices.

```
cumulus@switch:~$ netq show inventory disk
Matching inventory records:
Hostname          Name            Type             Transport
Size       Vendor               Model
----------------- --------------- ---------------- ------------------
---------- -------------------- ------------------------------
leaf01            vda             disk             N
/A                6G       0x1af4                 N/A
leaf02            vda             disk             N
/A                6G       0x1af4                 N/A
leaf03            vda             disk             N
/A                6G       0x1af4                 N/A
```

```
leaf04            vda             disk             N
/A                6G         0x1af4               N/A
oob-mgmt-server   vda             disk             N
/A                256G       0x1af4               N/A
server01          vda             disk             N/A
301G       0x1af4               N/A
server02          vda             disk             N/A
301G       0x1af4               N/A
server03          vda             disk             N/A
301G       0x1af4               N/A
server04          vda             disk             N
/A                301G       0x1af4               N/A
spine01           vda             disk             N
/A                6G         0x1af4               N/A
spine02           vda             disk             N
/A                6G         0x1af4               N/A
```

You can filter the results of the command to view the disk information for a particular device. This example shows disk information for *leaf03* switch.

```
cumulus@switch:~$ netq leaf03 show inventory disk
Matching inventory records:
Hostname          Name            Type             Transport
Size       Vendor               Model
----------------- --------------- ---------------- ------------------
---------- -------------------- ------------------------------
leaf03            vda             disk             N
/A                6G         0x1af4               N/A
```

## View Memory Information for a Switch

You can view the name, type, size, speed, vendor, and serial number for the memory installed in a single device or all devices. This example shows all of these characteristics for all devices.

```
cumulus@switch:~$ netq show inventory memory
Matching inventory records:
Hostname          Name            Type             Size
Speed      Vendor               Serial No
----------------- --------------- ---------------- ----------
---------- -------------------- -------------------------
dell-z9100-05     DIMM0 BANK 0    DDR3             8192 MB    1600
MHz   Hynix                14391421
mlx-2100-05       DIMM0 BANK 0    DDR3             8192 MB    1600
MHz   InnoDisk Corporation 00000000
mlx-2410a1-05     ChannelA-DIMM0  DDR3             8192 MB    1600
MHz   017A                 87416232
                  BANK 0
```

```
mlx-2700-11       ChannelA-DIMM0  DDR3             8192 MB    1600
MHz   017A                 73215444
                  BANK 0
mlx-2700-11       ChannelB-DIMM0  DDR3             8192 MB    1600
MHz   017A                 73215444
                  BANK 2
qct-ix1-08        N/A             N/A              7907.45MB  N
/A        N/A                   N/A
qct-ix7-04        DIMM0 BANK 0    DDR3             8192 MB    1600
MHz   Transcend            00211415
st1-l1            DIMM0 BANK 0    DDR3             4096 MB    1333
MHz   N/A                  N/A
st1-l2            DIMM0 BANK 0    DDR3             4096 MB    1333
MHz   N/A                  N/A
st1-l3            DIMM0 BANK 0    DDR3             4096 MB    1600
MHz   N/A                  N/A
st1-s1            A1_DIMM0 A1_BAN DDR3             8192 MB    1333
MHz   A1_Manufacturer0     A1_SerNum0
                  K0
st1-s2            A1_DIMM0 A1_BAN DDR3             8192 MB    1333
MHz   A1_Manufacturer0     A1_SerNum0
                  K0
```

You can filter the results of the command to view devices with a particular memory type or vendor. This example shows all of the devices with memory from *QEMU* .

```
cumulus@switch:~$ netq show inventory memory vendor QEMU
Matching inventory records:
Hostname          Name            Type             Size
Speed      Vendor               Serial No
----------------- --------------- ---------------- ----------
---------- -------------------- -------------------------
leaf01            DIMM 0          RAM              1024 MB
Unknown    QEMU                 Not Specified
leaf02            DIMM 0          RAM              1024 MB
Unknown    QEMU                 Not Specified
leaf03            DIMM 0          RAM              1024 MB
Unknown    QEMU                 Not Specified
leaf04            DIMM 0          RAM              1024 MB
Unknown    QEMU                 Not Specified
oob-mgmt-server   DIMM 0          RAM              4096 MB
Unknown    QEMU                 Not Specified
server01          DIMM 0          RAM              512 MB
Unknown    QEMU                 Not Specified
server02          DIMM 0          RAM              512 MB
Unknown    QEMU                 Not Specified
server03          DIMM 0          RAM              512 MB
Unknown    QEMU                 Not Specified
server04          DIMM 0          RAM              512 MB
Unknown    QEMU                 Not Specified
```

```
spine01           DIMM 0          RAM              1024 MB
Unknown    QEMU                 Not Specified
spine02           DIMM 0          RAM              1024 MB
Unknown    QEMU                 Not Specified
```

You can filter the results to view memory information for a single switch, as shown here for leaf01.

```
cumulus@switch:~$ netq leaf01 show inventory memory

Matching inventory records:
Hostname          Name            Type             Size
Speed      Vendor               Serial No
----------------- --------------- ---------------- ----------
---------- -------------------- -------------------------
leaf01            DIMM 0          RAM              1024 MB
Unknown    QEMU                 Not Specified
```

## View a Summary of All Hardware Information for a Switch

While the detail can be very helpful, sometimes a simple overview of the hardware inventory is better. This example shows the basic hardware information for all devices.

```
cumulus@switch:~$ netq show inventory brief

Matching inventory records:
Hostname          Switch               OS              CPU      ASIC
          Ports
----------------- -------------------- --------------- --------
--------------- ----------------------------------
leaf01            VX                   Cumulus Linux   x86_64   N
/A             N/A
leaf02            VX                   Cumulus Linux   x86_64   N
/A             N/A
leaf03            VX                   Cumulus Linux   x86_64   N
/A             N/A
leaf04            VX                   Cumulus Linux   x86_64   N
/A             N/A
oob-mgmt-server   VX                   Cumulus Linux   x86_64   N
/A             N/A
server01          N/A                  Ubuntu          x86_64   N
/A             N/A
server02          N/A                  Ubuntu          x86_64   N
/A             N/A
server03          N/A                  Ubuntu          x86_64   N
/A             N/A
server04          N/A                  Ubuntu          x86_64   N
/A             N/A
```

```
spine01             VX                  Cumulus Linux   x86_64   N
/A              N/A
spine02             VX                  Cumulus Linux   x86_64   N
/A              N/A
```

## Monitor Switch Software Information

NetQ enables you to view either a summary or details about the operating system and license, and whether NetQ Agents are running on your switch or host. The `netq show inventory` command is used to view the OS and license information for a single device or for all of your devices at once, depending on what you want to see. The `netq show agents` command is used to view the state of the NetQ Agents. You are also able to view the historical state of these items for one or all devices to determine if there have been any changes to their status.

The syntax for this command is:

```
netq [<hostname>] show inventory brief [json]
netq [<hostname>] show inventory license [cumulus] [around <text-
time>] [json]
netq [<hostname>] show inventory license [cumulus] changes [between
<text-time> and <text-endtime>] [json]
netq [<hostname>] show inventory os [version <os-version>|name <os-
name>] [json]
netq [<hostname>] show inventory os [version <os-version>|name <os-
name>] changes [between <text-time> and <text-endtime>] [json]
```

ⓘ The keyword values for the `name` keyword is specific to your deployment. For example, if you have devices with only one type of OS, say Cumulus Linux, then that is the only option available for the `os-name` keyword value. If you have multiple OSs running, say you also have Ubuntu, then that would also be an option for you.

ⓘ When entering a time value, you must include a numeric value *and* the unit of measure:

- w: week(s)
- d: day(s)
- h: hour(s)
- m: minute(s)
- s: second(s)
- now

For time ranges, the `<text-time>` is the most recent time and the `<text-endtime>` is the oldest time. The values do not have to have the same unit of measure.

## View OS Information for a Switch

You can view the name and version of the OS on a switch, and when it was last modified. This example shows the OS information for all devices.

```
cumulus@switch:~$ netq show inventory os

Matching inventory records:
Hostname          Name            Version
Last Changed
----------------- ---------------
------------------------------------ -------------------------
leaf01            Cumulus Linux   3.6.2.1~1533263732.
39254ac           21d:23h:26m:3s
leaf02            Cumulus Linux   3.6.2.1~1533263732.
39254ac           21d:23h:26m:15s
leaf03            Cumulus Linux   3.6.2.1~1533263732.
39254ac           21d:23h:26m:10s
leaf04            Cumulus Linux   3.6.1.0~1748339104.
32814bc           21d:23h:26m:6s
oob-mgmt-server   Cumulus Linux   3.7.0~1533263174.
bce9472             19d:7h:46m:24s
server04          Ubuntu
16.04                                21d:23h:26m:7s
server04          Ubuntu
16.04                                21d:23h:26m:13s
server04          Ubuntu
16.04                                21d:23h:26m:35s
server04          Ubuntu          16.04
21d:23h:26m:52s
spine01           Cumulus Linux   3.6.2.1~1533263732.
39254ac           21d:23h:26m:7s
spine02           Cumulus Linux   3.6.2.1~1533263732.
39254ac           21d:23h:26m:9s
```

You can filter the results of the command to view only devices with a particular operating system or version. This can be especially helpful when you suspect that a particular device has not been upgraded as expected. This example shows all devices with the Cumulus Linux version 3.6.1 installed.

```
cumulus@switch:~$ netq show inventory os version 3.6.1

Matching inventory records:
Hostname          Name            Version
Last Changed
----------------- ---------------
------------------------------------ -------------------------
leaf04            Cumulus Linux   3.6.1.0~1748339104.
32814bc           21d:23h:26m:6s
```

This example shows changes that have been made to the OS on all devices between 16 and 21 days ago. Remember to use measurement units on the time values.

```
cumulus@switch:~$ netq show inventory os changes between 16d and 21d
Matching inventory records:
Hostname           Name
Version                                      DB State   Last Changed
----------------- ---------------
------------------------------------ ----------
-------------------------
mlx-2410a1-05        Cumulus Linux
3.7.0                                      Add           16d:1h:39m:30s
mlx-2700-11        Cumulus Linux
3.7.0                                      Add           16d:1h:39m:32s
mlx-2100-05        Cumulus Linux
3.7.0                                      Add           16d:1h:39m:32s
mlx-2100-05        Cumulus Linux   3.7.0~1533263174.
bce9472                 Add           20d:0h:52m:4s
mlx-2700-11        Cumulus Linux   3.7.0~1533263174.
bce9472                 Add           20d:0h:52m:22s
mlx-2100-05        Cumulus Linux   3.7.0~1533263174.
bce9472                 Add           20d:18h:49m:31s
mlx-2700-11        Cumulus Linux   3.7.0~1533263174.
bce9472                 Add           20d:18h:49m:32s
```

## View License Information for a Switch

You can view the name and current state of the license (whether it valid or not), and when it was last updated for one or more devices. If a license is no longer valid on a switch, it does not operate correctly. This example shows the license information for all devices.

```
cumulus@switch:~$ netq show inventory license

Matching inventory records:
Hostname           Name            State      Last Changed
----------------- --------------- ---------- -------------------------
leaf01             Cumulus Linux   ok         21d:23h:43m:4s
leaf02             Cumulus Linux   ok         21d:23h:43m:16s
leaf03             Cumulus Linux   ok         21d:23h:43m:12s
leaf04             Cumulus Linux   ok         21d:23h:43m:7s
oob-mgmt-server    Cumulus Linux   ok         19d:8h:3m:34s
server01           Cumulus Linux   N/A        21d:23h:43m:8s
server02           Cumulus Linux   N/A        21d:23h:43m:17s
server03           Cumulus Linux   N/A        21d:23h:43m:25s
server04           Cumulus Linux   N/A        21d:23h:43m:31s
spine01            Cumulus Linux   ok         21d:23h:43m:8s
spine02            Cumulus Linux   ok         21d:23h:43m:11s
```

You can view the historical state of licenses using the around and changes keywords. This example shows the license state for all devices about 7 days ago. Remember to use measurement units on the time values.

```
cumulus@switch:~$ netq show inventory license around 7d

Matching inventory records:
Hostname          Name            State      Last Changed
----------------- --------------- ---------- -------------------------
leaf01            Cumulus Linux   ok         14d:23h:43m:4s
leaf02            Cumulus Linux   ok         14d:23h:43m:16s
leaf03            Cumulus Linux   ok         14d:23h:43m:12s
leaf04            Cumulus Linux   ok         14d:23h:43m:7s
oob-mgmt-server   Cumulus Linux   ok         13d:8h:3m:34s
server01          Cumulus Linux   N/A        14d:23h:43m:8s
server02          Cumulus Linux   N/A        14d:23h:43m:17s
server03          Cumulus Linux   N/A        14d:23h:43m:25s
server04          Cumulus Linux   N/A        14d:23h:43m:31s
spine01           Cumulus Linux   ok         14d:23h:43m:8s
spine02           Cumulus Linux   ok         14d:23h:43m:11s
```

You can filter the results to show license changes during a particular timeframe for a particular device. This example shows that there have been no changes to the license state on spine01 between now and two weeks ago.

```
cumulus@switch:~$ netq spine01 show inventory license changes between
now and 2w
No matching inventory records found
```

## View Summary of Operating System on a Switch

As with the hardware information, you can view a summary of the software information using the *brief* keyword. Specify a hostname to view the summary for a specific device.

```
cumulus@switch:~$ netq show inventory brief

Matching inventory records:
Hostname          Switch               OS              CPU      ASIC
          Ports
----------------- -------------------- --------------- --------
--------------- -----------------------------------
leaf01            VX                   Cumulus Linux   x86_64   N
/A             N/A
leaf02            VX                   Cumulus Linux   x86_64   N
/A             N/A
leaf03            VX                   Cumulus Linux   x86_64   N
/A             N/A
```

```
leaf04            VX                   Cumulus Linux   x86_64   N
/A             N/A
oob-mgmt-server   VX                   Cumulus Linux   x86_64   N
/A             N/A
server01          N/A                  Ubuntu          x86_64   N
/A             N/A
server02          N/A                  Ubuntu          x86_64   N
/A             N/A
server03          N/A                  Ubuntu          x86_64   N
/A             N/A
server04          N/A                  Ubuntu          x86_64   N
/A             N/A
spine01           VX                   Cumulus Linux   x86_64   N
/A             N/A
spine02           VX                   Cumulus Linux   x86_64   N
/A             N/A
```

## Validate NetQ Agents are Running

You can confirm that NetQ Agents are running on switches and hosts (if installed) using the `netq show agents` command. Viewing the **Status** column of the output indicates whether the agent is up and current, labelled *Fresh*, or down and stale, labelled *Rotten*. Additional information is provided about the agent status, including whether it is time synchronized, how long it has been up, and the last time its state changed.

This example shows NetQ Agent state on all devices. You can view the state for a single device using the *hostname* keyword.

```
cumulus@switch:~$ netq show agents

Matching agents records:
Hostname          Status           NTP Sync
Version                              Sys Uptime                Agent
Uptime              Reinitialize Time          Last Changed
----------------- ---------------- --------
------------------------------------ -------------------------
------------------------- --------------------------
-------------------------
leaf01            Fresh            no       1.3.0-cl3u9~1522970647.
b08ca60       22d:4h:39m:26s          22d:4h:19m:6s             22d:
0h:8m:20s              18.417234s
leaf02            Fresh            no       1.3.0-cl3u9~1522970647.
b08ca60       22d:4h:33m:0s           22d:4h:18m:26s            22d:
0h:8m:33s              15.413085s
leaf03            Fresh            no       1.3.0-cl3u9~1522970647.
b08ca60       22d:4h:35m:14s          22d:4h:18m:56s            22d:
0h:8m:28s              31.478846s
leaf04            Fresh            no       1.3.0-cl3u9~1522970647.
b08ca60       22d:4h:35m:48s          22d:4h:19m:5s             22d:
0h:8m:23s              17.819782s
```

```
oob-mgmt-server   Fresh            yes      1.4.0-cl3u10~1534306219.
882a7e7      22d:0h:10m:32s          19d:8h:28m:38s            19d:
8h:28m:38s             12.330358s
server01          Fresh            yes      1.3.0-ub16.
04u9~1522971904.b08ca60   22d:4h:25m:58s            11m:46.982
s                11m:46.982s                11.973292s
server02          Fresh            yes      1.3.0-ub16.
04u9~1522971904.b08ca60   22d:4h:25m:57s            10m:11.888
s                10m:11.888s                7.469695s
server03          Fresh            yes      1.3.0-ub16.
04u9~1522971904.b08ca60   22d:4h:26m:9s             9m:49.763
s                 9m:49.763s                 15.437087s
server04          Fresh            yes      1.3.0-ub16.
04u9~1522971904.b08ca60   22d:4h:26m:36s            22d:4h:21m:
6s               22d:0h:8m:23s              13.428345s
spine01           Fresh            no       1.3.0-cl3u9~1522970647.
b08ca60       22d:4h:40m:8s           22d:4h:18m:53s            22d:
0h:8m:24s              32.40132s
spine02           Fresh            no       1.3.0-cl3u9~1522970647.
b08ca60       22d:4h:33m:13s          22d:4h:19m:7s             22d:
0h:8m:27s              23.748967s
```

You can view the state of NetQ Agents at an earlier time using the *around* and *changes* keywords.

## Monitor Software Services

Cumulus Linux and NetQ run a number of services to deliver the various features of these products. You can monitor their status using the `netq show services` command. The services related to system-level operation are described here. Monitoring of other services, such as those related to routing, are described with those topics. NetQ automatically monitors t he following services:

- bgpd: BGP (Border Gateway Protocol) daemon
- clagd: MLAG (Multi-chassis Link Aggregation) daemon
- cumulus-chassis-ssh: Secure Shell for hardware chassis
- cumulus-chassisd: Chassis daemon
- ledmgrd: Switch LED manager daemon
- lldpd: LLDP (Link Layer Discovery Protocol) daemon
- mstpd: MSTP (Multiple Spanning Tree Protocol) daemon
- neighmgrd: Neighbor Manager daemon for BGP and OSPF
- netq-agent: NetQ Agent service
- netq-notifier: NetQ Notifier service
- netqd: NetQ telemetry application daemon
- ntp: NTP service
- ospf6d : OSPFv6 (Open Shortest Path First) daemon
- ospfd: OSPF daemon
- ptmd: PTM (Prescriptive Topology Manager) daemon

- pwmd : PWM ( Password Manager) daemon
- rsyslog: Rocket-fast system event logging processing service
- smond: System monitor daemon
- ssh: Secure Shell service for switches and servers
- status: License validation service
- syslog: System event logging service
- vrf: VRF (Virtual Route Forwarding) service
- vxrd: Registration daemon for VXLAN BUM ( broadcast, unknown unicast, and multicast) Flooding (VXFLD)
- vxsnd: Service node daemon for VXFLD
- zebra: GNU Zebra routing daemon

The CLI syntax for viewing the status of services is:

```
netq [<hostname>] show services [<service-name>] [vrf <vrf>]
[active|monitored] [around <text-time>] [json]
netq [<hostname>] show services [<service-name>] [vrf <vrf>]
[active|monitored] changes [between <text-time> and <text-endtime>]
[json]
netq [<hostname>] show services [<service-name>] [vrf <vrf>] status
(ok|warning|error|fail) [around <text-time>] [json]
netq [<hostname>] show services [<service-name>] [vrf <vrf>] status
(ok|warning|error|fail) changes [between <text-time> and <text-
endtime>] [json]
netq [<hostname>] show services [<service-name>] [vrf <vrf>] status
(ok|warning|error|fail) changes [json]
```

ⓘ The *active* and *monitored* keywords are not processed correctly in this release. Refer to the Release Notes for more detail.

## View All Services on All Devices

This example shows all of the available services on each device and whether each is enabled, active, and monitored, along with how long the service has been running and the last time it was changed.

```
cumulus@switch:~$ netq show services

Matching services records:
Hostname          Service              PID   VRF             Enabled
Active Monitored Status           Uptime                    Last
Changed
----------------- -------------------- ----- --------------- -------
------ --------- ---------------- -------------------------
-------------------------
```

```
leaf02            bgpd                 2013  default         yes     y
es    yes       ok               5h:1m:10s                  6m:41.781s
leaf02            clagd                1169  default         yes     y
es    yes       ok               5h:2m:0s                   6m:42.777s
leaf02            cumulus-chassis-ssh  n
/a   default         no      no     no         n/a              2d:4h:
15m:58s             2d:4h:15m:58s
leaf02            cumulus-chassisd     n
/a   default         no      no     no         n/a              2d:4h:
15m:58s             2d:4h:15m:58s
leaf02            ledmgrd              612   default         yes     y
es    no        ok               5h:2m:15s                  6m:50.946s
leaf02            lldpd                1160  default         yes     y
es    yes       ok               5h:2m:3s                   6m:28.313s
leaf02            mstpd                448   default         yes     y
es    yes       ok               5h:2m:19s                  6m:42.884s
leaf02            neighmgrd            1094  default         yes     y
es    no        ok               5h:2m:5s                   6m:50.940s
leaf02            netq-agent           n
/a   default         no      no     yes        n/a              6m:
50.943s                6m:50.943s
leaf02            netq-notifier        n
/a   default         no      no     yes        n/a              2d:4h:
15m:58s             2d:4h:15m:58s
leaf02            netqd                n
/a   default         no      no     yes        n/a              6m:
50.944s                6m:50.944s
leaf02            ntp                  n
/a   default         no      no     yes        n/a              6m:
50.936s                6m:50.936s
leaf02            ospf6d               n
/a   default         no      no     no         n/a              2d:4h:
15m:58s             2d:4h:15m:58s
leaf02            ospfd                n
/a   default         no      no     yes        n/a              6m:
41.922s                6m:41.922s
leaf02            ptmd                 1162  default         yes     y
es    no        ok               5h:2m:3s                   6m:51.441s
leaf02            pwmd                 613   default         yes     y
es    no        ok               5h:2m:15s                  6m:50.948s
leaf02            smond                609   default         yes     y
es    yes       ok               5h:2m:15s                  6m:28.279s
leaf02            ssh                  1125  default         yes     y
es    no        ok               5h:2m:5s                   6m:50.935s
leaf02            syslog               393   default         yes     y
es    no        ok               5h:2m:19s                  6m:50.934
leaf02            vxrd                 n
/a   default         no      no     yes        n/a              6m:
9.742s                 6m:9.742s
leaf02            vxsnd                n
/a   default         no      no     yes        n/a              6m:
9.756s                 6m:9.756s
```

```
leaf02           zebra                2006  default          yes      y
es    yes        ok              6m:28.244s                 6m:28.244s
server01         lldpd                1359  default          yes      y
es    yes        ok              2h:0m:25s                  4h:59m:45s
server01         netq-
agent            1363  default           yes      yes     yes        ok
          2h:0m:26s                 5h:0m:7s
server01         netq-notifier        n
/a    default        no       no      yes        n/a                2d:4h:
16m:1s             2d:4h:16m:1s
server01         netqd                1355  default          yes      y
es    yes        ok              2h:0m:26s                  5h:0m:7s
server01         ntp                  0     default          yes      y
es    yes        ok              2h:0m:20s                  5h:0m:7s
server01         ssh                  1358  default          yes      y
es    no         ok              2h:0m:25s                  5h:0m:7s
server01         syslog               967   default          yes      y
es    no         ok              2h:0m:27s                  5h:0m:7s
...
```

You can also view services information in JSON format:

```
cumulus@switch:~$ netq show services json
{
    "services":[
        {
            "status":"ok",
            "uptime":1537904537.0,
            "monitored":"yes",
            "service":"netqd",
            "lastChanged":1537893777.617677927,
            "pid":"1047",
            "hostname":"edge01",
            "enabled":"yes",
            "vrf":"default",
            "active":"yes"
        },
        {
            "status":"ok",
            "uptime":1537904537.0,
            "monitored":"yes",
            "service":"netq-agent",
            "lastChanged":1537893777.6185410023,
            "pid":"1052",
            "hostname":"edge01",
            "enabled":"yes",
            "vrf":"default",
            "active":"yes"
        },
...
```

If you want to view the service information for a given device, simply use the *hostname* variable to the command.

## View Information about a Given Service on All Devices

You can view the status of a given service at the current time, at a prior point in time, or view the changes that have occurred for the service during a specified timeframe.

This example shows how to view the status of the NTP service across the network. In this case, VRF is configured so the NTP service runs on both the default and management interface. You can perform the same command with the other services, such as `netq-agent`, `netq-notifier`, and `syslog`.

```
cumulus@switch:~$ netq show services ntp

Matching services records:
Hostname          Service              PID   VRF             Enabled
Active Monitored Status           Uptime                    Last
Changed
----------------- -------------------- ----- --------------- -------
------ --------- ---------------- -------------------------
-------------------------
edge01            ntp                  0     default         yes     y
es    yes       ok               2d:1h:24m:10s             2d:4h:23m:
36s
exit01            ntp                  1238  mgmt            yes     y
es    yes       ok               5h:9m:8s                  8m:4.578s
exit01            ntp                  n
/a   default         no      no      yes       n/a             8m:
4.583s                 8m:4.583s
exit02            ntp                  1233  mgmt            yes     y
es    yes       ok               5h:9m:8s                  7m:41.133s
exit02            ntp                  n
/a   default         no      no      yes       n/a             7m:
41.137s                7m:41.137s
internet          ntp                  n
/a   default         no      no      yes       n/a             5h:9m:
6s                  5h:9m:6s
internet          ntp                  n
/a   mgmt            yes     no      yes       n/a             5h:8m:
51s                 5h:8m:51s
leaf01            ntp                  1555  mgmt            yes     y
es    yes       ok               5h:9m:10s                 1h:1m:5s
leaf01            ntp                  n
/a   default         no      no      yes       n/a             1h:1m:
5s                  1h:1m:5s
leaf02            ntp                  1565  mgmt            yes     y
es    yes       ok               5h:9m:9s                  14m:25.774s
leaf02            ntp                  n
/a   default         no      no      yes       n/a             14m:
25.778s               14m:25.778s
```

```
leaf03            ntp                  1564  mgmt            yes      y
es    yes       ok               5h:9m:9s                  13m:36.464s
leaf03            ntp                  n
/a   default         no      no     yes        n/a              13m:
36.469s               13m:36.469s
leaf04            ntp                  1551  mgmt            yes      y
es    yes       ok               5h:9m:8s                  10m:15.960s
leaf04            ntp                  n
/a   default         no      no     yes        n/a              10m:
15.964s               10m:15.964s
oob-mgmt-
server    ntp                 813   default         yes     yes    yes
      ok               2d:4h:25m:35s             2d:4h:23m:9s
server01          ntp                  0     default         yes      y
es    yes       ok               2h:7m:55s                 5h:7m:42s
server02          ntp                  0     default         yes      y
es    yes       ok               2h:7m:55s                 5h:7m:42s
server03          ntp                  0     default         yes      y
es    yes       ok               2h:7m:55s                 5h:7m:42s
server04          ntp                  0     default         yes      y
es    yes       ok               2h:7m:55s                 5h:7m:42s
spine01           ntp                  1188  mgmt            yes      y
es    yes       ok               5h:9m:8s                  9m:32.856s
spine01           ntp                  n
/a   default         no      no     yes        n/a              9m:
32.861s                9m:32.861s
spine02           ntp                  1188  mgmt            yes      y
es    yes       ok               5h:9m:7s                  9m:4.722s
spine02           ntp                  n
/a   default         no      no     yes        n/a              9m:
4.726s                 9m:4.726s
```

This example shows the status of the BGP daemon.

```
cumulus@switch:~$ netq show services bgpd

Matching services records:
Hostname          Service              PID   VRF             Enabled
Active Monitored Status          Uptime                    Last
Changed
----------------- -------------------- ----- --------------- -------
------ --------- ---------------- -------------------------
-------------------------
exit01            bgpd                 1627  default         yes      y
es    yes       ok               5h:25m:43s                24m:55.103s
exit02            bgpd                 1628  default         yes      y
es    yes       ok               5h:25m:36s                24m:33.633s
internet          bgpd                 1493  default         yes      y
es    yes       ok               5h:25m:34s                5h:25m:20s
```

```
leaf01            bgpd                 2009  default         yes     y
es    yes       ok               5h:25m:44s                1h:17m:57s
leaf02            bgpd                 2013  default         yes     y
es    yes       ok               5h:25m:44s                31m:16.166s
leaf03            bgpd                 2010  default         yes     y
es    yes       ok               5h:25m:44s                30m:27.992s
leaf04            bgpd                 1998  default         yes     y
es    yes       ok               5h:25m:43s                27m:7.428s
spine01           bgpd                 1559  default         yes     y
es    yes       ok               5h:25m:35s                26m:24.326s
spine02           bgpd                 1553  default         yes     y
es    yes       ok               5h:25m:35s                25m:56.141s
```

# Monitor Physical Layer Components

With NetQ, a network administrator can monitor OSI Layer 1 physical components on network devices, including interfaces, ports, links, and peers. NetQ provides the ability to:

- Manage physical inventory: view the performance and status of various components of a switch or host server
- Validate configurations: verify the configuration of network peers and ports

It helps answer questions such as:

- Are any individual or bonded links down?
- Are any links flapping?
- Is there a link mismatch anywhere in my network?
- Which interface ports are empty?
- Which transceivers are installed?
- What is the peer for a given port?

NetQ uses LLDP (Link Layer Discovery Protocol) to collect port information. NetQ can also identify peer ports connected to DACs ( Direct Attached Cables) and AOCs (Active Optical Cables) without using LLDP, even if the link is not UP.

## Contents

This topic describes how to...

# Monitor Physical Layer Inventory

Keeping track of the various physical layer components in your switches and servers ensures you have a fully functioning network and provides inventory management and audit capabilities. You can monitor ports, transceivers, and cabling deployed on a per port (interface), per vendor, per part number and so forth. NetQ enables you to view the current status and the status an earlier point in time. From this information, you can, among other things:

- determine which ports are empty versus which ones have cables plugged in and thereby validate expected connectivity
- audit transceiver and cable components used by vendor, giving you insights for estimated replacement costs, repair costs, overall costs, and so forth to improve your maintenance and purchasing processes
- identify changes in your physical layer, and when they occurred

The `netq show interfaces physical` command is used to obtain the information from the devices. Its syntax is:

```
netq [<hostname>] show interfaces physical [<physical-port>]
[empty|plugged] [peer] [vendor <module-vendor> | model <module-
model>| module] [around <text-time>] [json]
netq [<hostname>] show interfaces physical [<physical-port>]
[empty|plugged] [vendor <module-vendor> | model <module-model> |
module] changes [between <text-time> and <text-endtime>] [json]
```

> When entering a time value, you must include a numeric value *and* the unit of measure:
>
> - d: day(s)
> - h: hour(s)
> - m: minute(s)
> - s: second(s)
> - now
>
> For time ranges, the `<text-time>` is the most recent time and the `<text-endtime>` is the oldest time. The values do not have to have the same unit of measure.

## View Detailed Cable Information for All Devices

You can view what cables are connected to each interface port for all devices, including the module type, vendor, part number and performance characteristics. You can also view the cable information for a given device by adding a hostname to the `show` command. This example shows cable information and status for all interface ports on all devices.

```
cumulus@switch:~$ netq show interfaces physical
Matching cables records:
```

```
Hostname          Interface                 State      Speed
AutoNeg Module    Vendor               Part No          Last Changed
----------------- ------------------------- ---------- ----------
------- --------- -------------------- ----------------
-------------------------
leaf01            swp1                         up         1G
off     SFP       AVAGO                AFBR-5715PZ-JU1  15d:22h:22m:4s
leaf01            swp2                      up          10G
off     SFP       OEM                  SFP-10GB-LR      15d:22h:22m:4s
leaf01            swp47                        up
10G           off     SFP       JDSU
PLRXPLSCS4322N   15d:22h:22m:4s
leaf01              swp48                    up
40G           off     QSFP+     Mellanox              MC2210130-
002    15d:22h:22m:4s
leaf01              swp49                    down         unknown
off     empty     n/a                   n/a              15d:22h:22m:
5s
leaf01              swp50                    up
1G           off     SFP       FINISAR CORP.         FCLF8522P2BTL
15d:22h:22m:5s
leaf01             swp51                    up
1G            off     SFP       FINISAR CORP.
FTLF1318P3BTL    15d:22h:22m:5s
leaf01            swp52                     down         unknown
off     SFP       CISCO-AGILENT         QFBR-5766LP      15d:22h:21m:
59s
leaf02            swp1                      up
1G            on      RJ45      n/a                   n
/a              15d:22h:21m:59s
leaf02            swp2                      up
10G           off     SFP       Mellanox              MC2609130-
003    15d:22h:22m:0s
leaf02            swp47                        up
10G          off     QSFP+     CISCO                 AFBR-7IER05Z-CS1
15d:22h:22m:0s
leaf02            swp48                        up
10G           off     QSFP+     Mellanox              MC2609130-
003    15d:22h:22m:0s
leaf02            swp49                     up
10G           off     SFP       FIBERSTORE            SFP-10GLR-
31     15d:22h:22m:0s
leaf02            swp50                     up
1G            off     SFP       OEM                   SFP-GLC-
T        15d:22h:22m:0s
leaf02            swp51                      up
10G           off     SFP       Mellanox              MC2609130-
003    15d:22h:22m:0s
leaf02            swp52                     up
1G            off     SFP       FINISAR CORP.
FCLF8522P2BTL    15d:22h:22m:0s
```

```
leaf03            swp1                      up
10G           off     SFP       Mellanox            MC2609130-003
15d:22h:21m:54s
leaf03            swp2                         up
10G          off     SFP       Mellanox              MC3309130-
001    15d:22h:21m:54s
leaf03            swp47                      up           10G
off     SFP       CISCO-AVAGO          AFBR-7IER05Z-CS1 15d:22h:21m:
54s
leaf03            swp48                     up
10G           off     SFP       Mellanox             MC3309130-
001    15d:22h:21m:54s
leaf03            swp49                      down         unknown
off     SFP       FINISAR CORP.        FCLF8520P2BTL    15d:22h:21m:
54s
leaf03            swp50                     up           1G
off     SFP       FINISAR CORP.        FCLF8522P2BTL    15d:22h:21m:
54s
leaf03               swp51                       up
10G          off     QSFP+     Mellanox            MC2609130-003
15d:22h:21m:54s
...
oob-mgmt-server   swp1                      up         1G
off     RJ45      n/a                 n/a              15d:22h:21m:4s
oob-mgmt-server   swp2                      up         1G
off     RJ45      n/a                 n/a              15d:22h:21m:4s
```

## View Detailed Module Information for a Given Device

You can view detailed information about the transceiver modules on each interface port, including serial number, transceiver type, connector and attached cable length. You can also view the module information for a given device by adding a hostname to the **show** command. This example shows the detailed module information for the interface ports on leaf02 switch.

```
cumulus@switch:~$ netq leaf02 show interfaces physical module
Matching cables records are:
Hostname          Interface                 Module
Vendor               Part No          Serial No
Transceiver      Connector        Length Last Changed

----------------- ------------------------- ---------
-------------------- ---------------- ------------------------
---------------- ---------------- ------ ------------------------
leaf02              swp1                      RJ45      n
/a                    n/a              n/a                      n
/a              n/a              n/a    15d:22h:49m:25s
leaf02            swp2                      SFP
Mellanox                MC2609130-003    MT1507VS05177
1000Base-CX,Copp Copper pigtail   3m     15d:22h:36m:25s
```

```
er Passive,Twin

Axial Pair (TW)
leaf02            swp47                     QSFP+
CISCO                   AFBR-7IER05Z-CS1 AVE1823402U               n
/a              n/a              5m     15d:21h:49m:25s
leaf02            swp48                     QSFP28    TE
Connectivity      2231368-1        15250052                   100G
Base-CR4 or n/a             3m     15d:22h:49m:25s

25G Base-CR CA-L

,40G Base-CR4
leaf02            swp49                     SFP
OEM                     SFP-10GB-LR      ACSLR130408
10G Base-LR      LC               10km,  15d:22h:49m:25s

10000m
leaf02            swp50                     SFP
JDSU                    PLRXPLSCS4322N   CG03UF45M
10G Base-SR,Mult LC               80m,   15d:22h:21m:25s

imode,                            30m,

50um (M5),Multim                  300m

ode,

62.5um (M6),Shor

twave laser w/o

OFC (SN),interme

diate distance (

I)
leaf02            swp51                      SFP
Mellanox                MC2609130-003    MT1507VS05177
1000Base-CX,Copp Copper pigtail   3m     15d:22h:49m:25s

er Passive,Twin

Axial Pair (TW)
leaf02            swp52                     SFP       FINISAR
CORP.           FCLF8522P2BTL    PTN1VH2                  1000Base-
T       RJ45             100m   15d:22h:49m:25s
```

## View Ports without Cables Connected for a Given Device

Checking for empty ports enables you to compare expected versus actual deployment. This can be very helpful during deployment or during upgrades. You can also view the cable information for a given device by adding a hostname to the `show` command. This example shows the ports that are empty on leaf01 switch.

```
cumulus@switch:~$ netq leaf01 show interfaces physical empty
Matching cables records are:
Hostname         Interface State Speed      AutoNeg Module
Vendor           Part No          Last Changed
---------------- --------- ----- ---------- ------- ---------
---------------- ---------------- --------------
Leaf01           swp49     down  Unknown    on      empty     n
/a                  n/a              1d:0h:16m:34s
Leaf01             swp52     down  Unknown    on      empty     n
/a                  n/a              1d:0h:16m:34s
```

## View Ports with Cables Connected for a Given Device

In a similar manner as checking for empty ports, you can check for ports that have cables connected, enabling you to compare expected versus actual deployment. You can also view the cable information for a given device by adding a hostname to the `show` command. If you add the around keyword, you can view which interface ports had cables connected at a previous time. This example shows the ports of *st1-11* switch that have attached cables.

```
cumulus@switch:~$ netq st1-11 show interfaces physical plugged
Matching cables records:
Hostname          Interface                 State      Speed
AutoNeg Module    Vendor                   Part No          Last Changed
----------------- ------------------------ ---------- ----------
------- --------- -------------------- ----------------
-------------------------
st1-l1            eth0                      up         1G
on      RJ45      n/a                  n/a               4h:31m:29s
st1-l1            swp1                      up         10G
off     SFP       Amphenol             610640005         4h:31m:29s
st1-l1            swp2                      up         10G
off     SFP       Amphenol             610640005         4h:31m:29s
st1-l1            swp3                      down       10G
off     SFP       Mellanox             MC3309130-001     4h:31m:29s
st1-l1            swp33                     down       10G
off     SFP       OEM                  SFP-H10GB-CU1M    4h:31m:27s
st1-l1            swp34                     down       10G
off     SFP       Amphenol             571540007         4h:31m:28s
st1-l1            swp35                     down       10G
off     SFP       Amphenol             571540007         4h:31m:28s
```

```
st1-l1            swp36                     down       10G
off     SFP       OEM                  SFP-H10GB-CU1M   4h:31m:28s
st1-l1            swp37                     down       10G
off     SFP       OEM                  SFP-H10GB-CU1M   4h:31m:28s
st1-l1            swp38                     down       10G
off     SFP       OEM                  SFP-H10GB-CU1M   4h:31m:25s
st1-l1            swp39                     down       10G
off     SFP       Amphenol             571540007        4h:31m:29s
st1-l1            swp40                     down       10G
off     SFP       Amphenol             571540007        4h:31m:25s
st1-l1            swp49                     up         40G
off     QSFP+     Amphenol             624410001        4h:31m:25s
st1-l1            swp5                      down       10G
off     SFP       Amphenol             571540007        4h:31m:27s
st1-l1            swp50                     down       40G
off     QSFP+     Amphenol             624410001        4h:31m:27s
st1-l1            swp51                     down       40G
off     QSFP+     Amphenol             603020003        4h:31m:27s
st1-l1            swp52                     up         40G
off     QSFP+     Amphenol             603020003        4h:31m:26s
st1-l1            swp54                     down       40G
off     QSFP+     Amphenol             624410002        4h:31m:27s
```

## View Components from a Given Vendor

By filtering for a specific cable vendor, you can collect information such as how many ports use components from that vendor and when they were last updated. This information may be useful when you run a cost analysis of your network. This example shows all the ports that are using components by an *OEM* vendor.

```
cumulus@switch:~$ netq st1-l1 show interfaces physical vendor OEM
Matching cables records:
Hostname          Interface                 State      Speed
AutoNeg Module    Vendor               Part No          Last Changed
----------------- ------------------------- ---------- ----------
------- --------- -------------------- ----------------
-------------------------
st1-l1            swp33                     down       10G
off     SFP       OEM                  SFP-H10GB-CU1M   4h:31m:37s
st1-l1            swp36                     down       10G
off     SFP       OEM                  SFP-H10GB-CU1M   4h:31m:39s
st1-l1            swp37                     down       10G
off     SFP       OEM                  SFP-H10GB-CU1M   4h:31m:39s
st1-l1            swp38                     down       10G
off     SFP       OEM                  SFP-H10GB-CU1M   4h:31m:36s
```

## View All Devices Using a Given Component

You can view all of the devices with ports using a particular component. This could be helpful when you need to change out a particular component for possible failure issues, upgrades, or cost reasons. This example first determines which models (part numbers) exist on all of the devices and then those devices with a part number of QSFP-H40G-CU1M installed.

```
cumulus@switch:~$ netq show interfaces physical model
    2231368-1         :  2231368-1
    624400001         :  624400001
    QSFP-H40G-CU1M    :  QSFP-H40G-CU1M
    QSFP-H40G-CU1MUS  :  QSFP-H40G-CU1MUS
    n/a               :  n/a

cumulus@switch:~$ netq show interfaces physical model QSFP-H40G-CU1M
Matching cables records:
Hostname          Interface                 State      Speed
AutoNeg Module    Vendor               Part No          Last Changed
----------------- ------------------------- ---------- ----------
------- --------- -------------------- ----------------
-------------------------
leaf01              swp50                     up
1G           off     QSFP+     OEM                     QSFP-H40G-
CU1M   15d:22h:22m:5s
leaf02            swp52                     up
1G            off     QSFP+     OEM                     QSFP-H40G-
CU1M   15d:22h:22m:0s
```

## View Changes to Physical Components

Because components are often changed, NetQ enables you to determine what, if any, changes have been made to the physical components on your devices. This can be helpful during deployments or upgrades.

You can select how far back in time you want to go, or select a time range using the between keyword. Note that time values must include units to be valid. If no changes are found, a "No matching cable records found" message is displayed. This example illustrates each of these scenarios for all devices in the network.

```
cumulus@switch:~$ netq show interfaces physical changes between now
and 30d
Matching cables records:
Hostname          Interface                 State      Speed
AutoNeg Module    Vendor               Part No          Last Changed
----------------- ------------------------- ---------- ----------
------- --------- -------------------- ----------------
-------------------------
leaf01            swp1                         up         1G
off     SFP       AVAGO                AFBR-5715PZ-JU1  15d:22h:22m:4s
```

```
leaf01            swp2                      up           10G
off     SFP       OEM                  SFP-10GB-LR      15d:22h:22m:4s
leaf01            swp47                        up
10G           off     SFP       JDSU
PLRXPLSCS4322N   15d:22h:22m:4s
leaf01              swp48                     up
40G           off     QSFP+     Mellanox              MC2210130-
002    15d:22h:22m:4s
leaf01              swp49                     down
10G           off     empty     n/a                   n
/a              15d:22h:22m:5s
leaf01              swp50                     up
1G           off     SFP       FINISAR CORP.        FCLF8522P2BTL
15d:22h:22m:5s
leaf01             swp51                     up
1G            off     SFP       FINISAR CORP.
FTLF1318P3BTL    15d:22h:22m:5s
leaf01            swp52                     down
1G            off     SFP       CISCO-AGILENT         QFBR-
5766LP      15d:22h:21m:59s
leaf02            swp1                      up
1G            on      RJ45      n/a                   n
/a              15d:22h:21m:59s
leaf02            swp2                      up
10G           off     SFP       Mellanox              MC2609130-
003    15d:22h:22m:0s
leaf02            swp47                       up
10G          off     QSFP+     CISCO                AFBR-7IER05Z-CS1
15d:22h:22m:0s
leaf02            swp48                       up
10G           off     QSFP+     Mellanox              MC2609130-
003    15d:22h:22m:0s
leaf02            swp49                     up
10G           off     SFP       FIBERSTORE            SFP-10GLR-
31     15d:22h:22m:0s
leaf02            swp50                     up
1G            off     SFP       OEM                   SFP-GLC-
T        15d:22h:22m:0s
leaf02            swp51                      up
10G           off     SFP       Mellanox              MC2609130-
003    15d:22h:22m:0s
leaf02            swp52                     up
1G            off     SFP       FINISAR CORP.
FCLF8522P2BTL    15d:22h:22m:0s
leaf03            swp1                      up
10G           off     SFP       Mellanox             MC2609130-003
15d:22h:21m:54s
leaf03            swp2                         up
10G          off     SFP       Mellanox                MC3309130-
001    15d:22h:21m:54s
```

```
leaf03            swp47                      up           10G
off     SFP       CISCO-AVAGO           AFBR-7IER05Z-CS1 15d:22h:21m:
54s
leaf03            swp48                     up
10G           off     SFP       Mellanox              MC3309130-
001    15d:22h:21m:54s
leaf03            swp49                      down
1G            off     SFP       FINISAR CORP.
FCLF8520P2BTL    15d:22h:21m:54s
leaf03            swp50                     up           1G
off     SFP       FINISAR CORP.         FCLF8522P2BTL    15d:22h:21m:
54s
leaf03               swp51                       up
10G          off      QSFP+     Mellanox             MC2609130-003
15d:22h:21m:54s
...
oob-mgmt-server   swp1                      up         1G
off     RJ45      n/a                  n/a                15d:22h:21m:4s
oob-mgmt-server   swp2                      up         1G
off     RJ45      n/a                  n/a                15d:22h:21m:4s

cumulus@switch:~$ netq show interfaces physical changes between 6d
and 16d
Matching cables records:
Hostname          Interface                 State      Speed
AutoNeg Module    Vendor               Part No          Last Changed
----------------- ------------------------- ---------- ----------
------- --------- -------------------- ----------------
-------------------------
leaf01            swp1                         up         1G
off     SFP       AVAGO                AFBR-5715PZ-JU1  15d:22h:22m:4s
leaf01            swp2                      up           10G
off     SFP       OEM                  SFP-10GB-LR      15d:22h:22m:4s
leaf01            swp47                        up
10G           off     SFP       JDSU
PLRXPLSCS4322N   15d:22h:22m:4s
leaf01              swp48                     up
40G           off     QSFP+     Mellanox              MC2210130-
002    15d:22h:22m:4s
leaf01              swp49                     down
10G           off     empty     n/a                   n
/a              15d:22h:22m:5s
leaf01              swp50                     up
1G           off      SFP       FINISAR CORP.         FCLF8522P2BTL
15d:22h:22m:5s
leaf01             swp51                     up
1G            off     SFP       FINISAR CORP.
FTLF1318P3BTL    15d:22h:22m:5s
leaf01            swp52                     down
1G            off     SFP       CISCO-AGILENT         QFBR-
5766LP      15d:22h:21m:59s
...
```

```
cumulus@switch:~$ netq show interfaces physical changes between 0s
and 5h
No matching cables records found
```

## Validate Physical Layer Configuration

Beyond knowing what physical components are deployed, it is valuable to know that they are configured and operating correctly. NetQ enables you to confirm that peer connections are present, discover any misconfigured ports, peers, or unsupported modules, and monitor for link flaps.

NetQ checks peer connections using LLDP. For DACs and AOCs, NetQ determines the peers using their serial numbers in the port EEPROMs, even if the link is not UP.

### Confirm Peer Connections

You can validate peer connections for all devices in your network or for a specific device or port. This example shows the peer hosts and their status for leaf03 switch.

```
cumulus@switch:~$ netq leaf03 show interfaces physical peer
Matching cables records:
Hostname          Interface                 Peer Hostname     Peer
Interface            State      Message
----------------- ------------------------- -----------------
------------------------- ----------
-----------------------------------
leaf03            swp1                      oob-mgmt-switch
swp7                       up
leaf03
swp2
down        Peer port unknown
leaf03            swp47                     leaf04
swp47                      up
leaf03            swp48                     leaf04
swp48                      up
leaf03            swp49                     leaf04
swp49                      up
leaf03            swp50                     leaf04
swp50                      up
leaf03            swp51                     exit01
swp51                     up
leaf03
swp52
down       Port cage empty
```

This example shows the peer data for a specific interface port.

```
cumulus@switch:~$ netq leaf01 show interfaces physical swp47 peer
```

```
Matching cables records:
Hostname          Interface                 Peer Hostname     Peer
Interface            State      Message
----------------- ------------------------- -----------------
------------------------- ----------
-----------------------------------
leaf01              swp47                     leaf02
swp47                      up
```

## Discover Misconfigurations

You can verify that the following configurations are the same on both sides of a peer interface:

- Admin state
- Operational state
- Link speed
- Auto-negotiation setting

The `netq check interfaces` command is used to determine if any of the interfaces have any continuity errors. This command only checks the physical interfaces; it does not check bridges, bonds or other software constructs. You can check all interfaces at once, or for a given device, or check the connection between a given device and its peer. It enables you to compare the current status of the interfaces, as well as their status at an earlier point in time. The command syntax is:

```
netq check interfaces [unverified] [<physical-hostname> <physical-
port> | <physical-hostname>] [around <text-time>] [json]
netq check interfaces <physical-hostname> <physical-port> and <peer-
physical-hostname> <peer-physical-port> [around <text-time>] [json]
```

If NetQ cannot determine a peer for a given device, the port is marked as *unverified*.

If you find a misconfiguration, use the `netq show interfaces physical` command for clues about the cause.

**Example: Find Mismatched Operational States**

In this example, we check all of the interfaces for misconfigurations and we find that one interface port has an error. We look for clues about the cause and see that the Operational states do not match on the connection between leaf 03 and leaf04: leaf03 is up, but leaf04 is down. If the misconfiguration was due to a mismatch in the administrative state, the message would have been *Admin state mismatch (up, down)* or *Admin state mismatch (down, up)*.

```
cumulus@switch:~$ netq check interfaces
Checked Nodes: 18, Failed Nodes: 8
Checked Ports: 741, Failed Ports: 1, Unverified Ports: 414

cumulus@switch:~$ netq show interfaces physical peer
Matching cables records:
```

```
Hostname          Interface                 Peer Hostname     Peer
Interface            State      Message
----------------- ------------------------- -----------------
------------------------- ----------
-----------------------------------
...
leaf03            swp1                      oob-mgmt-switch
swp7                       up
leaf03
swp2
down        Peer port unknown
leaf03            swp47                     leaf04
swp47                      up
leaf03            swp48                     leaf04
swp48                      up          State mismatch (up, down)
leaf03            swp49                     leaf04
swp49                      up
leaf03            swp50                     leaf04
swp50                      up
leaf03
swp52
down       Port cage empty
...
```

### Example: Find Mismatched Peers

This example uses the *and* keyword to check the connections between two peers. An error is seen, so we check the physical peer information and discover that the incorrect peer has been specified. After fixing it, we run the check again, and see that there are no longer any interface errors.

```
cumulus@switch:~$ netq check interfaces leaf01 swp50 and leaf02 swp50
Checked Nodes: 1, Failed Nodes: 1
Checked Ports: 1, Failed Ports: 1, Unverified Ports: 0
cumulus@switch:~$ netq show interfaces physical peer

Matching cables records:
Hostname          Interface                 Peer Hostname     Peer
Interface            State      Message
----------------- ------------------------- -----------------
------------------------- ----------
-----------------------------------
leaf01              swp50                      leaf04
swp49                                 Incorrect peer specified. Real
peer

is leaf04 swp50

cumulus@switch:~$ netq check interfaces leaf01 swp50 and leaf02 swp50
Checked Nodes: 1, Failed Nodes: 0
Checked Ports: 1, Failed Ports: 0, Unverified Ports: 0
```

### Example: Find Mismatched Link Speeds

This example checks for for configuration mismatches and finds a link speed mismatch on server03. The link speed on swp49 is *40G* and the peer port swp50 is *unspecified*.

```
cumulus@switch:~$ netq check interfaces
Checked Nodes: 10, Failed Nodes: 1
Checked Ports: 125, Failed Ports: 2, Unverified Ports: 35
Hostname          Interface                 Peer Hostname     Peer
Interface            Message
----------------- ------------------------- -----------------
------------------------- -----------------------------------
server03          swp49                     server03
swp50                     Speed mismatch (40G, Unknown)
server03          swp50                     server03          swp49
                  Speed mismatch (Unknown, 40G)
```

### Example: Find Mismatched Auto-negotiation Settings

This example checks for configuration mismatches and finds auto-negotation setting mismatches between the servers and leafs. Auto-negotiation is *off* on the leafs, but *on* on the servers.

```
cumulus@switch:~$ netq check interfaces
Checked Nodes: 15, Failed Nodes: 8
Checked Ports: 118, Failed Ports: 8, Unverified Ports: 94
Hostname          Interface                 Peer Hostname     Peer
Interface            Message
----------------- ------------------------- -----------------
------------------------- -----------------------------------
leaf01            swp1                      server01          eth1
                  Autoneg mismatch (off, on)
leaf02            swp2                      server02          eth2
                  Autoneg mismatch (off, on)
leaf03            swp1                      server03          eth1
                  Autoneg mismatch (off, on)
leaf04            swp2                      server04          eth2
                  Autoneg mismatch (off, on)
server01          eth1                      leaf01            swp1
                  Autoneg mismatch (on, off)
server02          eth2                      leaf02            swp2
                  Autoneg mismatch (on, off)
server03          eth1                      leaf03            swp1
                  Autoneg mismatch (on, off)
server04          eth2                      leaf04            swp2
                  Autoneg mismatch (on, off)
```

## Identify Flapping Links

You can also determine whether a link is flapping using the `netq check interfaces` and `netq show interfaces physical peer` commands. If a link is flapping, NetQ indicates this in a message:

```
cumulus@switch:~$ netq check interfaces
Checked Nodes: 18, Failed Nodes: 8
Checked Ports: 741, Failed Ports: 1, Unverified Ports: 414

cumulus@switch:~$ netq show interfaces physical peer
Matching cables records:
Hostname          Interface                 Peer Hostname     Peer
Interface            State      Message
----------------- ------------------------- -----------------
------------------------- ----------
-----------------------------------
leaf02            -                             -
-                            -           Link flapped 11 times in
last 5

mins
```

# Monitor Data Link Layer Devices and Protocols

With NetQ, a network administrator can monitor OSI Layer 2 devices and protocols, including switches, bridges, link control, and physical media access. Keeping track of the various data link layer devices in your network ensures consistent and error-free communications between devices. NetQ provides the ability to:

- Monitor and validate device and protocol configurations
- View available communication paths between devices

It helps answer questions such as:

- Is a VLAN misconfigured?
- Is there an MTU mismatch in my network?
- Is MLAG configured correctly?
- Is there an STP loop?
- Can device A reach device B using MAC addresses?

## Contents

This topic describes how to...

- View Paths between Two Switches with Detailed Output (see page 99)

## Monitor LLDP Operation

LLDP is used by network devices for advertising their identity, capabilities, and neighbors on a LAN. You can view this information for one or more devices. You can also view the information at an earlier point in time or view changes that have occurred to the information during a specified timeframe. NetQ enables you to view LLDP information for your devices using the **netq show lldp** command. The syntax for this command is:

```
netq [<hostname>] show lldp [<remote-physical-interface>] [around
<text-time>] [json]
netq [<hostname>] show lldp [<remote-physical-interface>] changes
[between <text-time> and <text-endtime>] [json]
```

### View LLDP Information for All Devices

This example shows the interface and peer information that is advertised for each device.

```
cumulus@switch:~$ netq show lldp

Matching lldp records:
Hostname          Interface                 Peer Hostname     Peer
Interface            Last Changed
----------------- ------------------------- -----------------
------------------------- -------------------------
leaf01            eth0                      oob-mgmt-
switch   swp6                      4h:22m:57s
leaf01            swp1                      server01          eth1
                  4h:15m:40s
leaf01            swp51                     spine01           swp1
                  4h:16m:12s
leaf01            swp52                     spine02           swp1
                  4h:16m:12s
leaf02            eth0                      oob-mgmt-
switch   swp7                      4h:22m:53s
leaf02            swp2                      server02          eth2
                  4h:15m:38s
leaf02            swp51                     spine01           swp2
                  4h:16m:9s
leaf02            swp52                     spine02           swp2
                  4h:16m:9s
leaf03            eth0                      oob-mgmt-
switch   swp8                      4h:23m:5s
leaf03            swp1                      server03          eth1
                  4h:15m:50s
leaf03            swp51                     spine01           swp3
                  4h:16m:21s
```

```
leaf03            swp52                     spine02           swp3
                  4h:16m:21s
leaf04            eth0                      oob-mgmt-
switch   swp9                      4h:23m:1s
leaf04            swp2                      server04          eth2
                  4h:15m:46s
leaf04            swp51                     spine01           swp4
                  4h:16m:17s
leaf04            swp52                     spine02           swp4
                  4h:16m:17s
oob-mgmt-server   eth1                      oob-mgmt-
switch   swp1                      4h:23m:4s
server01          eth0                      oob-mgmt-
switch   swp2                      4h:23m:8s
server01          eth1                      leaf01            swp1
                  4h:15m:28s
server02          eth0                      oob-mgmt-
switch   swp3                      4h:22m:59s
server02          eth2                      leaf02            swp2
                  4h:15m:29s
server03          eth0                      oob-mgmt-
switch   swp4                      4h:23m:5s
server03          eth1                      leaf03            swp1
                  4h:15m:28s
server04          eth0                      oob-mgmt-
switch   swp5                      4h:23m:2s
server04          eth2                      leaf04            swp2
                  4h:15m:28s
spine01           eth0                      oob-mgmt-
switch   swp10                     4h:23m:6s
spine01           swp1                      leaf01            swp51
                  4h:16m:22s
spine01           swp2                      leaf02            swp51
                  4h:16m:22s
spine01           swp3                      leaf03            swp51
                  4h:16m:22s
spine01           swp4                      leaf04            swp51
                  4h:16m:22s
spine02           eth0                      oob-mgmt-
switch   swp11                     4h:23m:7s
spine02           swp1                      leaf01            swp52
                  4h:16m:22s
spine02           swp2                      leaf02            swp52
                  4h:16m:22s
spine02           swp3                      leaf03            swp52
                  4h:16m:22s
spine02           swp4                      leaf04            swp52
                  4h:16m:22s
```

## View Changes to LLDP Information

If you are experiencing a connectivity issue with a particular device, using the **changes** keyword can help determine if a configuration change might be a cause. If no changes are found, a *No matching lldp records found* message is displayed.

This example shows the current LLDP information and all changes that have occurred in the LLDP information for *tor-1*.

```
cumulus@switch:~$ netq tor-1 show lldp
Matching lldp records:
Hostname          Interface                 Peer Hostname     Peer
Interface            Last Changed
----------------- ------------------------- -----------------
------------------------- -------------------------
tor-1             swp1                      noc-
pr            swp4                      30m:21.735s
tor-1             swp2                      noc-
se            swp4                      30m:21.735s
tor-1             swp3                      spine-
1           swp7                      30m:21.735s
tor-1             swp4                      spine-
2           swp7                      30m:21.735s
tor-1             swp5                      spine-
3           swp7                      30m:21.735s
tor-1             swp6                      hosts-11          mac:00:
02:00:00:00:27     25m:42.653s
tor-1             swp7                      hosts-
12          swp1                      30m:21.734s
tor-1             swp8                      hosts-13          mac:00:
02:00:00:00:2d     25m:42.651s

cumulus@switch:~$ netq tor-1 show lldp changes
Matching lldp records:
Hostname          Interface                 Peer Hostname     Peer
Interface            DB State   Last Changed
----------------- ------------------------- -----------------
------------------------- ---------- -------------------------
tor-1             swp8                      hosts-13          mac:00:
02:00:00:00:2d     Add        25m:45.593s
tor-1             swp6                      hosts-11          mac:00:
02:00:00:00:27     Add        25m:45.595s
tor-1             swp8                      hosts-13          mac:00:
02:00:00:00:2d     Del        26m:17.954s
tor-1             swp6                      hosts-11          mac:00:
02:00:00:00:27     Del        26m:17.965s
tor-1             swp8                      hosts-13          mac:00:
02:00:00:00:2d     Add        26m:17.100s
tor-1             swp6                      hosts-11          mac:00:
02:00:00:00:27     Add        26m:17.101s
```

```
tor-1             swp6                      hosts-11          mac:00:
02:00:00:00:27     Add        27m:19.630s
tor-1             swp6                      hosts-11          mac:00:
02:00:00:00:27     Del        27m:49.517s
tor-1             swp6                      hosts-11          mac:00:
02:00:00:00:27     Add        27m:49.522s
tor-1             swp8                      hosts-13          mac:00:
02:00:00:00:2d     Add        30m:24.676s
tor-1             swp7                      hosts-
12          swp1                      Add        30m:24.677s
tor-1             swp6                      hosts-11          mac:00:
02:00:00:00:27     Add        30m:24.677s
tor-1             swp5                      spine-
3           swp7                      Add        30m:24.677s
tor-1             swp4                      spine-
2           swp7                      Add        30m:24.677s
tor-1             swp3                      spine-
1           swp7                      Add        30m:24.677s
tor-1             swp2                      noc-
se            swp4                      Add        30m:24.678s
tor-1             swp1                      noc-
pr            swp4                      Add        30m:24.678s
```

## Check for MTU Inconsistencies

The maximum transmission unit (MTU) determines the largest size packet or frame that can be transmitted across a given communication link. When the MTU is not configured to the same value on both ends of the link, communication problems can occur. With NetQ, you can verify that the MTU is correctly specified for each link using the netq check mtu command.

This example shows that four switches have inconsistently specified link MTUs. Now the network administrator or operator can reconfigure the switches and eliminate the communication issues associated with this misconfiguration.

```
cumulus@switch:~$ netq check mtu
Checked Nodes: 15, Checked Links: 215, Failed Nodes: 4, Failed Links:
8
MTU mismatch found on following links
Hostname          Interface                 MTU    Peer              P
eer Interface            Peer MTU Error
----------------- ------------------------- ------ -----------------
------------------------- -------- ---------------
spine01           swp30                     9216   exit01            s
wp51                     1500     MTU Mismatch
exit01            swp51                     1500   spine01           s
wp30                     9216     MTU Mismatch
spine01           swp29                     9216   exit02            s
wp51                     1500     MTU Mismatch
exit02            swp51                     1500   spine01           s
wp29                     9216     MTU Mismatch
```

```
exit01            swp52                     1500   spine02            s
wp30                     9216     MTU Mismatch
spine02           swp30                     9216   exit01             s
wp52                     1500     MTU Mismatch
spine02           swp29                     9216   exit02             s
wp52                     1500     MTU Mismatch
exit02            swp52                     1500   spine02            s
wp29                     9216     MTU Mismatch
```

## Monitor VLAN Configurations

A VLAN (Virtual Local Area Network) enables devices on one or more LANs to communicate as if they were on the same network, without being physically connected. The VLAN enables network administrators to partition a network for functional or security requirements without changing physical infrastructure. With NetQ, you can view the operation of VLANs for one or all devices. You can also view the information at an earlier point in time or view changes that have occurred to the information during a specified timeframe. NetQ enables you to view basic VLAN information for your devices using the `netq show vlan` command. Additional show commands enable you to view VLAN information associated with interfaces and MAC addresses. The syntax for these commands is:

```
netq [<hostname>] show vlan [<1-4096>] [around <text-time>] [json]
netq [<hostname>] show vlan [<1-4096>] changes [between <text-time>
and <text-endtime>] [json]
netq [<hostname>] show interfaces type (macvlan|vlan) [state <remote-
interface-state>] [around <text-time>] [count] [json]
netq [<hostname>] show interfaces type (macvlan|vlan) changes
[between <text-time> and <text-endtime>] [json]
netq [<hostname>] show macs [<mac>] [vlan <1-4096>] [origin] [around
<text-time>] [json]
netq [<hostname>] show macs [<mac>] [vlan <1-4096>] [around <text-
time>] count [json]
netq [<hostname>] show macs [<mac>] [vlan <1-4096>] [origin] changes
[between <text-time> and <text-endtime>] [json]
netq <hostname> show macs egress-port <egress-port> [<mac>] [vlan <1-
4096>] [origin] [around <text-time>] [json]
netq <hostname> show macs egress-port <egress-port> [<mac>] [vlan <1-
4096>] [origin] changes [between <text-time> and <text-endtime>]
[json]
```

ⓘ When entering a time value, you must include a numeric value *and* the unit of measure:

- d: day(s)
- h: hour(s)
- m: minute(s)
- s: second(s)
- now

For time ranges, the `<text-time>` is the most recent time and the `<text-endtime>` is the oldest time. The values do not have to have the same unit of measure.

## View VLAN Information for All Devices

This example shows the VLANs configured across your network.

```
cumulus@switch:~$ netq show vlan
Matching vlan records:
Hostname          VLANs                     SVIs
Last Changed
----------------- ------------------------- -------------------------
-------------------------
exit01            4001                      4001
19h:31m:35s
exit02            4001                      4001
19h:31m:11s
leaf01            1,13,24,4001              13 24 4001
20h:24m:35s
leaf02            1,13,24,4001              13 24 4001
19h:37m:56s
leaf03            1,13,24,4001              13 24 4001
19h:37m:6s
leaf04            1,13,24,4001              13 24 4001
19h:33m:46s
```

## View Changes to VLAN Information

If you are experiencing a connectivity issue with a particular device, using the `changes` keyword can help determine if a configuration change might be a cause. If no changes are found, a *No matching vlan records found* message is displayed.

When no timeframe is specified for the changes keyword, the default value, *between now and 1h*, is used.

This example shows all changes that have occurred for all VLANs in the last hour.

```
cumulus@switch:~$ netq show vlan changes
No matching vlan records found
```

This example shows all changes that have occurred for all VLANs on the network in the past two days.

```
cumulus@switch:~$ netq show vlan changes between now and 2d
Matching vlan records:
```

```
Hostname          VLANs                     SVIs
DB State   Last Changed
----------------- ------------------------- -------------------------
---------- -------------------------
exit02            4001                      4001
Add        19h:33m:10s
exit01            4001                      4001
Add        19h:33m:33s
leaf04            1,13,24,4001              13 24 4001
Add        19h:35m:45s
leaf03            1,13,24,4001              13 24 4001
Add        19h:39m:5s
leaf02            1,13,24,4001              13 24 4001
Add        19h:39m:54s
leaf01            1,13,24,4001              13 24 4001
Add        20h:26m:34s
```

## View VLAN Interface Information

You can view the current or past state of the interfaces associated with VLANs using the `netq show interfaces` command. This provides the status of the interface, its specified MTU, whether it is running over a VRF, and the last time it was changed.

```
cumulus@switch:~$ netq show interfaces type vlan
Matching link records:
Hostname          Interface                 Type             State
  VRF             Details                             Last Changed
----------------- ------------------------- ----------------
---------- --------------- -----------------------------------
-------------------------
exit01            vlan4001                  vlan             up
  vrf1            MTU:1500                            19h:35m:46s
exit02            vlan4001                  vlan             up
  vrf1            MTU:1500                            19h:35m:23s
leaf01            peerlink.
4094             vlan             up          default        MTU:
9000                            20h:28m:47s
leaf01            vlan13                    vlan             up
  vrf1            MTU:1500                            20h:28m:47s
leaf01            vlan24                    vlan             up
  vrf1            MTU:1500                            20h:28m:47s
leaf01            vlan4001                  vlan             up
  vrf1            MTU:1500                            20h:28m:47s
leaf02            peerlink.
4094             vlan             up          default        MTU:
9000                            19h:42m:7s
leaf02            vlan13                    vlan             up
  vrf1            MTU:1500                            19h:42m:7s
leaf02            vlan24                    vlan             up
  vrf1            MTU:1500                            19h:42m:7s
```

```
leaf02            vlan4001                  vlan             up
  vrf1            MTU:1500                            19h:42m:7s
leaf03            peerlink.
4094             vlan             up         default          MTU:
9000                            19h:41m:18s
leaf03            vlan13                    vlan             up
  vrf1            MTU:1500                            19h:41m:18s
leaf03            vlan24                    vlan             up
  vrf1            MTU:1500                            19h:41m:18s
leaf03            vlan4001                  vlan             up
  vrf1            MTU:1500                            19h:41m:18s
leaf04            peerlink.
4094             vlan             up         default          MTU:
9000                            19h:37m:58s
leaf04            vlan13                    vlan             up
  vrf1            MTU:1500                            19h:37m:58s
leaf04            vlan24                    vlan             up
  vrf1            MTU:1500                            19h:37m:58s
leaf04            vlan4001                  vlan             up
  vrf1            MTU:1500                            19h:37m:58s
```

## View MAC Addresses Associated with a VLAN

You can determine the MAC addresses associated with a given VLAN using the `netq show macs vlan` command. The command also provides the hostname of the devices, the egress port for the interface, whether the MAC address originated from the given device, whether it learns the MAC address from the peer (remote=yes), and the last time the configuration was changed.

This example shows the MAC addresses associated with *VLAN13*.

```
cumulus@switch:~$ netq show macs vlan 13
Matching mac records:
Origin MAC Address        VLAN   Hostname          Egress
Port          Remote Last Changed
------ ------------------ ------ -----------------
-------------------- ------ -------------------------
no     00:03:00:11:11:01  13     leaf01            bond01:
server01      no     20h:31m:23s
no     00:03:00:11:11:01  13     leaf02            bond01:
server01      no     19h:44m:44s
no     00:03:00:11:11:01  13     leaf03            vni13:
leaf01         yes    19h:43m:55s
no     00:03:00:11:11:01  13     leaf04            vni13:
leaf01         yes    19h:40m:34s
no     00:03:00:33:33:01  13     leaf01            vni13:
10.0.0.134     yes    20h:31m:23s
no     00:03:00:33:33:01  13     leaf02            vni13:
10.0.0.134     yes    19h:44m:44s
no     00:03:00:33:33:01  13     leaf03            bond03:
server03      no     19h:43m:55s
```

```
no     00:03:00:33:33:01  13      leaf04            bond03:
server03      no     19h:40m:34s
no     02:03:00:11:11:01  13      leaf01            bond01:
server01      no     20h:31m:23s
no     02:03:00:11:11:01  13      leaf02            bond01:
server01      no     19h:44m:44s
no     02:03:00:11:11:01  13      leaf03            vni13:
leaf01         yes    19h:43m:55s
no     02:03:00:11:11:01  13      leaf04            vni13:
leaf01         yes    19h:40m:34s
no     02:03:00:11:11:02  13      leaf01            bond01:
server01      no     20h:31m:23s
no     02:03:00:11:11:02  13      leaf02            bond01:
server01      no     19h:44m:44s
no     02:03:00:11:11:02  13      leaf03            vni13:
leaf01         yes    19h:43m:55s
no     02:03:00:11:11:02  13      leaf04            vni13:
leaf01         yes    19h:40m:34s
no     02:03:00:33:33:01  13      leaf01            vni13:
10.0.0.134     yes    20h:31m:23s
no     02:03:00:33:33:01  13      leaf02            vni13:
10.0.0.134     yes    19h:44m:44s
no     02:03:00:33:33:01  13      leaf03            bond03:
server03      no     19h:43m:55s
no     02:03:00:33:33:01  13      leaf04            bond03:
server03      no     19h:40m:34s
no     02:03:00:33:33:02  13      leaf01            vni13:
10.0.0.134     yes    20h:31m:23s
no     02:03:00:33:33:02  13      leaf02            vni13:
10.0.0.134     yes    19h:44m:44s
no     02:03:00:33:33:02  13      leaf03            bond03:
server03      no     19h:43m:55s
no     02:03:00:33:33:02  13      leaf04            bond03:
server03      no     19h:40m:34s
yes    44:38:39:00:00:
03  13     leaf01            bridge              no     20h:31m:23s
yes    44:38:39:00:00:
15  13     leaf02            bridge              no     19h:44m:44s
yes    44:38:39:00:00:
23  13     leaf03            bridge              no     19h:43m:55s
yes    44:38:39:00:00:
5c  13     leaf04            bridge              no     19h:40m:34s
yes    44:39:39:ff:00:
13  13     leaf01            bridge              no     20h:31m:23s
yes    44:39:39:ff:00:
13  13     leaf02            bridge              no     19h:44m:44s
yes    44:39:39:ff:00:
13  13     leaf03            bridge              no     19h:43m:55s
yes    44:39:39:ff:00:
13  13     leaf04            bridge              no     19h:40m:34s
```

## View MAC Addresses Associated with an Egress Port

You can filter that information down to just the MAC addresses that are associated with a given VLAN that use a particular egress port. This example shows MAC addresses associated with the *leaf03* switch and *VLAN 13* that use the *bridge* port.

```
cumulus@switch:~$ netq leaf03 show macs egress-port bridge vlan 13
Matching mac records:
Origin MAC Address        VLAN   Hostname          Egress
Port          Remote Last Changed
------ ------------------ ------ -----------------
-------------------- ------ -------------------------
yes    44:38:39:00:00:
23  13     leaf03           bridge               no     20h:46m:17s
yes    44:39:39:ff:00:
13  13     leaf03           bridge               no     20h:46m:17s
```

## View the MAC Addresses Associated with VRR Configurations

You can view all of the MAC addresses associated with your VRR (virtual router reflector) interface configuration using the **`netq show interfaces type macvlan`** command. This is useful for determining if the specified MAC address inside a VLAN is the same or different across your VRR configuration.

```
cumulus@switch:~$ netq show interfaces type macvlan
Matching link records:
Hostname          Interface                 Type             State
  VRF             Details                                 Last Changed
----------------- ------------------------- ----------------
---------- --------------- -----------------------------------
-------------------------
leaf01            vlan13-
v0                 macvlan          up        vrf1            MAC:
44:39:39:ff:00:13,             21h:37m:14s

                  Mode: Private
leaf01            vlan24-
v0                 macvlan          up        vrf1            MAC:
44:39:39:ff:00:24,             21h:37m:14s

                  Mode: Private
leaf02            vlan13-
v0                 macvlan          up        vrf1            MAC:
44:39:39:ff:00:13,             20h:50m:35s

                  Mode: Private
leaf02            vlan24-
v0                 macvlan          up        vrf1            MAC:
44:39:39:ff:00:24,             20h:50m:35s
```

```
                    Mode: Private
leaf03              vlan13-
v0                   macvlan          up         vrf1             MAC:
44:39:39:ff:00:13,            20h:49m:45s

                    Mode: Private
leaf03              vlan24-
v0                   macvlan          up         vrf1             MAC:
44:39:39:ff:00:24,            20h:49m:45s

                    Mode: Private
leaf04              vlan13-
v0                   macvlan          up         vrf1             MAC:
44:39:39:ff:00:13,            20h:46m:25s

                    Mode: Private
leaf04              vlan24-
v0                   macvlan          up         vrf1             MAC:
44:39:39:ff:00:24,            20h:46m:25s

                    Mode: Private
```

## Monitor MLAG Configurations

Multi-Chassis Link Aggregation (MLAG) is used to enable 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. Dual-connected devices can create LACP bonds that contain links to each physical switch. Therefore, active-active links from the dual-connected devices are supported even though they are connected to two different physical switches.

### 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 `clagd`, and other options in the code, such as `clag-id`, which exist for historical purposes. The Cumulus Linux implementation is truly a multi-chassis link aggregation protocol, so we call it MLAG.

For instructions on configuring MLAG, refer to the MLAG topic in the Cumulus Linux User Guide.

With NetQ, you can view the configuration and operation of devices using MLAG using the `netq show clag` command. You can view the current configuration and the configuration at a prior point in time, as well as view any changes that have been made within a timeframe. The syntax for the show command is:

```
netq [<hostname>] show clag [around <text-time>] [json]
netq [<hostname>] show clag changes [between <text-time> and <text-
endtime>] [json]
```

## View MLAG Configuration and Status for all Devices

This example shows the configuration and status of MLAG for all devices. In this case, three MLAG pairs are seen between torc-11 and torc-12 (which happens to be down), noc-pr(P) and noc-se, and torc-21(P) and torc-22.

```
cumulus@ts:~$ netq show clag
Matching CLAG session records are:
Hostname          Peer              SysMac             State
Backup #Bond #Dual Last Changed

s
----------------- ----------------- ------------------ ----------
------ ----- ----- -------------------------
torc-11                             44:38:39:ff:ff:01  down       n
/a    0     0     1m:43.468s
torc-12                             44:38:39:ff:ff:01  down
down   8     0     2m:11.967s
noc-pr(P)         noc-se            00:01:01:10:00:01  up
up     25    25    35m:29.324s
noc-se            noc-pr(P)         00:01:01:10:00:01  up
up     25    25    35m:26.551s
torc-21(P)        torc-22           44:38:39:ff:ff:02  up
up     8     8     35m:10.140s
torc-22           torc-21(P)        44:38:39:ff:ff:02  up
up     8     8     35m:7.342s
```

You can go back in time to see when this first MLAG pair went down. These results indicate that the pair became disconnected some time in the last five minutes.

```
cumulus@switch:~$ netq show clag around 5m
Matching clag records:
Hostname          Peer              SysMac             State      Back
up #Bond #Dual Last Changed

   s
----------------- ----------------- ------------------ ----------
------ ----- ----- -------------------------
noc-pr(P)         noc-se            00:01:01:10:00:
01  up         up     25    25    30m:40.399s
noc-se            noc-pr(P)         00:01:01:10:00:
01  up         up     25    25    30m:37.267s
torc-11(P)        torc-12           44:38:39:ff:ff:
01  up         up     8     8     30m:27.250s
torc-12           torc-11(P)        44:38:39:ff:ff:
01  up         up     8     8     30m:23.552s
torc-21(P)        torc-22           44:38:39:ff:ff:
02  up         up     8     8     30m:20.856s
```

```
torc-22           torc-21(P)        44:38:39:ff:ff:
02  up        up     8     8     30m:18.583s
```

## View MLAG Configuration and Status for Given Devices

This example shows that xxx device is up and MLAG properly configured with a peer connection to yyy and 8 bonds, all of which are dual bonded.

```
cumulus@switch:~$ netq tor-22 show clag
Matching CLAG session records are:
Hostname          Peer              SysMac             State
Backup #Bond #Dual Last Changed

s
----------------- ----------------- ------------------ ----------
------ ----- ----- -------------------------
torc-22           torc-21(P)        44:38:39:ff:ff:02  up
up     8     8     35m:7.342s
```

When you're directly on the switch, you can run `clagctl` to get the state:

```
cumulus@mlx-2700-03:/var/log# sudo clagctl

The peer is alive
Peer Priority, ID, and Role: 4096 00:02:00:00:00:4e primary
Our Priority, ID, and Role: 8192 44:38:39:00:a5:38 secondary
Peer Interface and IP: peerlink-3.4094 169.254.0.9
VxLAN Anycast IP: 36.0.0.20
Backup IP: 27.0.0.20 (active)
System MAC: 44:38:39:ff:ff:01

CLAG Interfaces
Our Interface    Peer Interface   CLAG Id Conflicts            Proto-
Down Reason
---------------- ---------------- ------- --------------------
-----------------
vx-38            vx-38            -       -                    -
vx-33            vx-33            -       -                    -
hostbond4        hostbond4        1       -                    -
hostbond5        hostbond5        2       -                    -
vx-37            vx-37            -       -                    -
vx-36            vx-36            -       -                    -
vx-35            vx-35            -       -                    -
vx-34            vx-34            -       -                    -
```

# Monitor Time Synchronization Status for Devices

It is important that the switches and hosts remain in time synchronization with the Telemetry Server to ensure collected data is properly captured and processed. You can use the `netq show ntp` command to view the time synchronization status for all devices or filter for devices that are either in synchronization or out of synchronization, currently or at a time in the past. The syntax for the show command is:

```
netq [<hostname>] show ntp [out-of-sync|in-sync] [json]
netq [<hostname>] show ntp [out-of-sync|in-sync] around <text-time>
[json]
```

This example shows the time synchronization status for all devices in the network.

```
cumulus@switch:~$ netq show ntp

Matching ntp records:
Hostname          NTP Sync Current Server    Stratum NTP App
----------------- -------- ----------------- -------
---------------------
edge01            yes      oob-mgmt-server   3       ntpq
exit01            yes      christensenplac   2       ntpq
exit02            yes      owners.kjsl.com   2       ntpq
internet          no       -                 16      ntpq
leaf01            yes      christensenplac   2       ntpq
leaf02            yes      owners.kjsl.com   2       ntpq
leaf03            yes      107.181.191.189   2       ntpq
leaf04            yes      grom.polpo.org    2       ntpq
oob-mgmt-server   yes      linode227395.st   2       ntpq
server01          yes      192.168.0.254     3       ntpq
server02          yes      192.168.0.254     3       ntpq
server03          yes      192.168.0.254     3       ntpq
server04          yes      192.168.0.254     3       ntpq
spine01           yes      107.181.191.189   2       ntpq
spine02           yes      t2.time.bf1.yah   2       ntpq
```

This example shows all devices in the network that are out of time synchronization, and consequently might need to be investigated.

```
cumulus@switch:~$ netq show ntp out-of-sync

Matching ntp records:
Hostname          NTP Sync Current Server    Stratum NTP App
----------------- -------- ----------------- -------
---------------------
internet          no       -                 16      ntpq
```

This example shows the time synchronization status for *leaf01*.

```
cumulus@switch:~$ netq leaf01 show ntp

Matching ntp records:
Hostname          NTP Sync Current Server    Stratum NTP App
----------------- -------- ----------------- -------
---------------------
leaf01            yes      kilimanjaro       2       ntpq
```

## Monitor Spanning Tree Protocol Configuration

The Spanning Tree Protocol (STP) is used in Ethernet-based networks to prevent communication loops when you have redundant paths on a bridge or switch. Loops cause excessive broadcast messages greatly impacting the network performance. With NetQ, you can view the STP topology on a bridge or switch to ensure no loops have been created using the `netq show stp topology` command. You can also view the topology information for a prior point in time to see if there have been changes from that point until now. The syntax for the show command is:

```
netq <hostname> show stp topology [json]
netq <hostname> show stp topology around <text-time> [json]
```

This example shows the STP topology as viewed from the *spine1* switch.

```
cumulus@switch:~$ netq spine1 show stp topology
Root(spine1) -- spine1:sw_clag200 -- leaf2:EdgeIntf(sng_hst2) --
hsleaf21
                                  -- leaf2:EdgeIntf(dual_host2) --
hdleaf2
                                  -- leaf2:EdgeIntf(dual_host1) --
hdleaf1
                                  -- leaf2:ClagIsl(peer-bond1) --
leaf1
                                  -- leaf1:EdgeIntf(sng_hst2) --
hsleaf11
                                  -- leaf1:EdgeIntf(dual_host2) --
hdleaf2
                                  -- leaf1:EdgeIntf(dual_host1) --
hdleaf1
                                  -- leaf1:ClagIsl(peer-bond1) --
leaf2
             -- spine1:ClagIsl(peer-bond1) -- spine2
             -- spine1:sw_clag300 -- edge1:EdgeIntf(sng_hst2) --
hsedge11
                                  -- edge1:EdgeIntf(dual_host2) --
hdedge2
```

```
                                  -- edge1:EdgeIntf(dual_host1) --
hdedge1
                                  -- edge1:ClagIsl(peer-bond1) --
edge2
                                  -- edge2:EdgeIntf(sng_hst2) --
hsedge21
                                  -- edge2:EdgeIntf(dual_host2) --
hdedge2
                                  -- edge2:EdgeIntf(dual_host1) --
hdedge1
                                  -- edge2:ClagIsl(peer-bond1) --
edge1
Root(spine2) -- spine2:sw_clag200 -- leaf2:EdgeIntf(sng_hst2) --
hsleaf21
                                  -- leaf2:EdgeIntf(dual_host2) --
hdleaf2
                                  -- leaf2:EdgeIntf(dual_host1) --
hdleaf1
                                  -- leaf2:ClagIsl(peer-bond1) --
leaf1
                                  -- leaf1:EdgeIntf(sng_hst2) --
hsleaf11
                                  -- leaf1:EdgeIntf(dual_host2) --
hdleaf2
                                  -- leaf1:EdgeIntf(dual_host1) --
hdleaf1
                                  -- leaf1:ClagIsl(peer-bond1) --
leaf2
             -- spine2:ClagIsl(peer-bond1) -- spine1
             -- spine2:sw_clag300 -- edge2:EdgeIntf(sng_hst2) --
hsedge21
                                  -- edge2:EdgeIntf(dual_host2) --
hdedge2
                                  -- edge2:EdgeIntf(dual_host1) --
hdedge1
                                  -- edge2:ClagIsl(peer-bond1) --
edge1
                                  -- edge1:EdgeIntf(sng_hst2) --
hsedge11
                                  -- edge1:EdgeIntf(dual_host2) --
hdedge2
                                  -- edge1:EdgeIntf(dual_host1) --
hdedge1
                                  -- edge1:ClagIsl(peer-bond1) --
edge2
```

# Validate Paths between Devices

If you have VLANs configured, you can view the available paths between two devices on the VLAN currently and at a time in the past using their MAC addresses . You can perform the trace in only one direction or both, and view the output in one of three formats ( *json, pretty,* and *detail* ). JSON output provides the output in a JSON file format for ease of importing to other applications or software. Pretty output lines up the paths in a pseudo-graphical manner to help visualize multiple paths. Detail output is useful for traces with higher hop counts where the pretty output wraps lines, making it harder to interpret the results. The detail output displays a table with a row for each path.

To view the paths:

1. Identify the MAC address and VLAN ID for the destination device
2. Identify the IP address or hostname for the source device
3. Use the `netq trace` command to see the available paths between those devices.

The trace command syntax is:

```
netq trace <mac> [vlan <1-4096>] from (<src-hostname>|<ip-src>) [vrf
<vrf>] [around <text-time>] [bidir] [json|detail|pretty] [debug]
```

> The syntax requires the destination device address first, *<mac>*, and then the source device address or hostname. Additionally, the *vlan* keyword-value pair is required for layer 2 traces even though the syntax indicates it is optional.
>
> The tracing function only knows about addresses that have already been learned. If you find that a path is invalid or incomplete, you may need to ping the identified device so that its address becomes known.

## View Paths between Two Switches with Pretty Output

This example shows the available paths between a top of rack switch, *tor-1*, and a server, *hostd-11*. The request is to go through VLAN *1001* from the VRF *vrf1*. The results include a summary of the trace, including the total number of paths available, those with errors and warnings, and the MTU of the paths. In this case, the results are displayed in pseudo-graphical output.

```
cumulus@switch:~$ netq trace  00:02:00:00:00:02 vlan 1001 from tor-1
vrf vrf1  pretty
Number of Paths: 4
Number of Paths with Errors: 0
Number of Paths with Warnings: 0
Path MTU: 9152
 tor-1 vni: 34 uplink-2 -- downlink-5 spine-2 downlink-2 -- uplink-2
vni: 34 torc-12 hostbond4 -- swp2 hostd-11
               uplink-2 -- downlink-5 spine-2 downlink-1 -- uplink-2
vni: 34 torc-11 hostbond4 -- swp1 hostd-11
```

```
 tor-1 vni: 34 uplink-1 -- downlink-5 spine-1 downlink-2 -- uplink-1
vni: 34 torc-12 hostbond4 -- swp2 hostd-11
               uplink-1 -- downlink-5 spine-1 downlink-1 -- uplink-1
vni: 34 torc-11 hostbond4 -- swp1 hostd-11
```

Alternately, you can use the IP address of the source device, as shown in this example.

```
cumulus@redis-1:~$  netq trace 00:02:00:00:00:02 vlan 1001 from
6.0.0.8 vrf vrf1  pretty
Number of Paths: 4
Number of Paths with Errors: 0
Number of Paths with Warnings: 0
Path MTU: 9152
 hosts-11 swp1 -- swp5 <vlan1000> tor-1 <vlan1001> vni: 34 uplink-2
-- downlink-5 spine-2 downlink-2 -- uplink-2 vni: 34 <vlan1001> torc-
12 hostbond4 -- swp2 hostd-11
                                                           uplink-2
-- downlink-5 spine-2 downlink-1 -- uplink-2 vni: 34 <vlan1001> torc-
11 hostbond4 -- swp1 hostd-11
          swp1 -- swp5 <vlan1000> tor-1 <vlan1001> vni: 34 uplink-1
-- downlink-5 spine-1 downlink-2 -- uplink-1 vni: 34 <vlan1001> torc-
12 hostbond4 -- swp2 hostd-11
                                                           uplink-1
-- downlink-5 spine-1 downlink-1 -- uplink-1 vni: 34 <vlan1001> torc-
11 hostbond4 -- swp1 hostd-11
```

## View Forward and Reverse Paths between Two Switches with Pretty Output

Like the previous example, this shows the paths between tor-1 and hostd-11, but by adding the *bidir* keyword both the forward and reverse paths are presented. Optionally, you can use the source device's hostname to achieve the same results.

```
cumulus@switch:~$ netq trace  00:02:00:00:00:02 vlan 1001 from tor-1
vrf vrf1 bidir pretty
Number of Paths: 4
Number of Paths with Errors: 0
Number of Paths with Warnings: 0
Path MTU: 9152
 tor-1 vni: 34 uplink-2 -- downlink-5 spine-2 downlink-2 -- uplink-2
vni: 34 torc-12 hostbond4 -- swp2 hostd-11
               uplink-2 -- downlink-5 spine-2 downlink-1 -- uplink-2
vni: 34 torc-11 hostbond4 -- swp1 hostd-11
 tor-1 vni: 34 uplink-1 -- downlink-5 spine-1 downlink-2 -- uplink-1
vni: 34 torc-12 hostbond4 -- swp2 hostd-11
               uplink-1 -- downlink-5 spine-1 downlink-1 -- uplink-1
vni: 34 torc-11 hostbond4 -- swp1 hostd-11
```

```
Number of Paths: 4
Number of Paths with Errors: 0
Number of Paths with Warnings: 0
Path MTU: 9152
 hostd-11 swp2 -- uplink-2 vni: 34 torc-12 hostbond4 -- downlink-2
spine-2 downlink-5 -- uplink-2 vni:34 tor-1
                  uplink-1 vni: 34 torc-12 hostbond4 -- downlink-2
spine-2 downlink-5 -- uplink-2 vni:34 tor-1
 hostd-11 swp2 -- uplink-2 vni: 34 torc-12 hostbond4 -- downlink-2
spine-2 downlink-5 -- uplink-1 vni:34 tor-1
                  uplink-1 vni: 34 torc-12 hostbond4 -- downlink-2
spine-2 downlink-5 -- uplink-1 vni:34 tor-1
```

## View Paths between Two Switches with Detailed Output

This example provides the same path information as the pretty output, but displays the information in a tabular output.

```
cumulus@switch:~$ netq trace  00:02:00:00:00:02 vlan 1001 from
6.0.0.8 vrf vrf1 bidir detail
Number of Paths: 4
Number of Paths with Errors: 0
Number of Paths with Warnings: 0
Path MTU: 9152
Id  Hop Hostname        InPort          InVlan InTunnel
InRtrIf         InVRF           OutRtrIf        OutVRF
OutTunnel             OutPort         OutVlan
--- --- --------------- --------------- ------ ---------------------
--------------- --------------- --------------- ---------------
--------------------- --------------- -------
1   1   hosts-
11
swp1            1000
    2   tor-1           swp5            1000
vlan1000        vrf1            vlan1001        vrf1            vni:
34               uplink-2
    3   spine-2         downlink-5
downlink-5      default         downlink-2
default                                 downlink-2
    4   torc-12         uplink-2               vni: 34
vlan1001
vrf1
hostbond4       1001
    5   hostd-11        swp2
--- --- --------------- --------------- ------ ---------------------
--------------- --------------- --------------- ---------------
--------------------- --------------- -------
2   1   hosts-
11
swp1            1000
```

```
    2   tor-1           swp5            1000
vlan1000        vrf1            vlan1001        vrf1            vni:
34               uplink-2
    3   spine-2         downlink-5
downlink-5      default         downlink-1
default                               downlink-1
    4   torc-11         uplink-2                vni: 34
vlan1001
vrf1
hostbond4       1001
    5   hostd-11        swp1
--- --- --------------- --------------- ------ ---------------------
--------------- --------------- --------------- ---------------
--------------------- --------------- -------
3   1   hosts-
11
swp1            1000
    2   tor-1           swp5            1000
vlan1000        vrf1            vlan1001        vrf1            vni:
34               uplink-1
    3   spine-1         downlink-5
downlink-5      default         downlink-2
default                               downlink-2
    4   torc-12         uplink-1                vni: 34
vlan1001
vrf1
hostbond4       1001
    5   hostd-11        swp2
--- --- --------------- --------------- ------ ---------------------
--------------- --------------- --------------- ---------------
--------------------- --------------- -------
4   1   hosts-
11
swp1            1000
    2   tor-1           swp5            1000
vlan1000        vrf1            vlan1001        vrf1            vni:
34               uplink-1
    3   spine-1         downlink-5
downlink-5      default         downlink-1
default                               downlink-1
    4   torc-11         uplink-1                vni: 34
vlan1001
vrf1
hostbond4       1001
    5   hostd-11        swp1
--- --- --------------- --------------- ------ ---------------------
--------------- --------------- --------------- ---------------
--------------------- --------------- -------
Number of Paths: 4
Number of Paths with Errors: 0
Number of Paths with Warnings: 0
Path MTU: 9152
```

```
Id  Hop Hostname        InPort          InVlan InTunnel
InRtrIf         InVRF           OutRtrIf        OutVRF
OutTunnel             OutPort         OutVlan
--- --- --------------- --------------- ------ ---------------------
--------------- --------------- --------------- ---------------
--------------------- --------------- -------
1   1   hostd-
11
swp2            1001
    2   torc-12         swp7            1001
vlan1001        vrf1            vlan1000        vrf1            vni:
33               uplink-2
    3   spine-2         downlink-2
downlink-2      default         downlink-5
default                               downlink-5
    4   tor-1           uplink-2               vni: 33
vlan1000
vrf1
hostbond3       1000
    5   hosts-11        swp1
--- --- --------------- --------------- ------ ---------------------
--------------- --------------- --------------- ---------------
--------------------- --------------- -------
2   1   hostd-
11
swp2            1001
    2   torc-12         swp7            1001
vlan1001        vrf1            vlan1000        vrf1            vni:
33               uplink-1
    3   spine-1         downlink-2
downlink-2      default         downlink-5
default                               downlink-5
    4   tor-1           uplink-1               vni: 33
vlan1000
vrf1
hostbond3       1000
    5   hosts-11        swp1
--- --- --------------- --------------- ------ ---------------------
--------------- --------------- --------------- ---------------
--------------------- --------------- -------
3   1   hostd-
11
swp1            1001
    2   torc-11         swp7            1001
vlan1001        vrf1            vlan1000        vrf1            vni:
33               uplink-2
    3   spine-2         downlink-1
downlink-1      default         downlink-5
default                               downlink-5
```

```
    4   tor-1           uplink-2               vni: 33
vlan1000
vrf1
hostbond3       1000
    5   hosts-11        swp1
--- --- --------------- --------------- ------ ---------------------
--------------- --------------- --------------- ---------------
--------------------- --------------- -------
4   1   hostd-
11
swp1            1001
    2   torc-11         swp7            1001
vlan1001        vrf1            vlan1000        vrf1            vni:
33               uplink-1
    3   spine-1         downlink-1
downlink-1      default         downlink-5
default                               downlink-5
    4   tor-1           uplink-1               vni: 33
vlan1000
vrf1
hostbond3       1000
    5   hosts-11        swp1
--- --- --------------- --------------- ------ ---------------------
--------------- --------------- --------------- ---------------
--------------------- --------------- -------
```

# Monitor Network Layer Protocols

With NetQ, a network administrator can monitor OSI Layer 3 network protocols running on Linux-based hosts, including IP (Internet Protocol), BGP (Border Gateway Protocol) and OSPF (Open Shortest Path First). NetQ provides the ability to:

- Validate protocol configurations
- Validate layer 3 communication paths

It helps answer questions such as:

- Who are the IP neighbors for a switch?
- How many IPv4 and IPv6 addresses am I using?
- When did changes occur to my IP configuration?
- Is BGP working as expected?
- Is OSPF working as expected?
- Can device A reach device B using IP addresses?

## Contents

This topic describes how to...

## Monitor IP Configuration

NetQ enables you to view the current status and the status an earlier point in time. From this information, you can:

- determine IP addresses of one or more interfaces

- determine IP neighbors for one or more devices
- determine IP routes owned by a device
- identify changes to the IP configuration

The `netq show ip` command is used to obtain the address, neighbor, and route information from the devices. Its syntax is:

```
netq [<hostname>] show ip addresses [<remote-interface>] [<ipv4>|<ipv4
/prefixlen>] [vrf <vrf>] [around <text-time>] [count] [json]
netq [<hostname>] show ip addresses [<remote-interface>] [<ipv4>|<ipv4
/prefixlen>] [vrf <vrf>] changes [between <text-time> and <text-
endtime>] [json]
netq [<hostname>] show ipv6 addresses [<remote-interface>]
[<ipv6>|<ipv6/prefixlen>] [vrf <vrf>] [around <text-time>] [count]
[json]
netq [<hostname>] show ipv6 addresses [<remote-interface>]
[<ipv6>|<ipv6/prefixlen>] [vrf <vrf>] changes [between <text-time>
and <text-endtime>] [json]

netq [<hostname>] show ip neighbors [<remote-interface>]
[<ipv4>|<ipv4> vrf <vrf>|vrf <vrf>] [<mac>] [around <text-time>]
[count] [json]
netq [<hostname>] show ip neighbors [<remote-interface>]
[<ipv4>|<ipv4> vrf <vrf>|vrf <vrf>] [<mac>] changes [between <text-
time> and <text-endtime>] [json]
netq [<hostname>] show ipv6 neighbors [<remote-interface>]
[<ipv6>|<ipv6> vrf <vrf>|vrf <vrf>] [<mac>] [around <text-time>]
[count] [json]
netq [<hostname>] show ipv6 neighbors [<remote-interface>]
[<ipv6>|<ipv6> vrf <vrf>|vrf <vrf>] [<mac>] changes [between <text-
time> and <text-endtime>] [json]

netq [<hostname>] show ip routes [<ipv4>|<ipv4/prefixlen>] [vrf
<vrf>] [origin] [around <text-time>] [count] [json]
netq [<hostname>] show ip routes [<ipv4>|<ipv4/prefixlen>] [vrf
<vrf>] [origin] changes [between <text-time> and <text-endtime>]
[json]
netq [<hostname>] show ipv6 routes [<ipv6>|<ipv6/prefixlen>] [vrf
<vrf>] [origin] [around <text-time>] [count] [json]
netq [<hostname>] show ipv6 routes [<ipv6>|<ipv6/prefixlen>] [vrf
<vrf>] [origin] changes [between <text-time> and <text-endtime>]
[json]
```

When entering a time value, you must include a numeric value *and* the unit of measure:

- d: day(s)
- h: hour(s)
- m: minute(s)

- s: second(s)
- now

For time ranges, the `<text-time>` is the most recent time and the `<text-endtime>` is the oldest time. The values do not have to have the same unit of measure.

## View IP Address Information

You can view the IPv4 and IPv6 address information for all of your devices, including the interface and VRF for each device. Additionally, you can:

- view the information at an earlier point in time
- view changes that have occurred over time
- filter against a particular device, interface or VRF assignment
- obtain a count of all of the addresses

Each of these provides information for troubleshooting potential configuration and communication issues at the layer 3 level.

**Example: View IPv4 address information for all devices**

```
cumulus@switch:~$ netq show ip addresses
Matching address records:
Address                   Hostname          Interface
VRF             Last Changed
------------------------- ----------------- -------------------------
--------------- -------------------------
10.0.0.11/32              leaf01            lo
default         36m:9.186s
10.0.0.12/32              leaf02            lo
default         36m:5.412s
10.0.0.13/32              leaf03            lo
default         35m:58.302s
10.0.0.14/32              leaf04            lo
default         35m:47.537s
10.0.0.21/32              spine01           lo
default         35m:53.615s
10.0.0.22/32              spine02           lo
default         35m:44.264s
10.0.0.254/32             oob-mgmt-server   eth0
default         22d:17h:40m:1s
172.16.1.1/24             leaf01            br0
default         36m:6.258s
172.16.1.101/24           server01          eth1
default         23m:19.110s
172.16.2.1/24             leaf02            br0
default         35m:57.423s
172.16.2.101/24           server02          eth2
default         21m:48.101s
172.16.3.1/24             leaf03            br0
default         35m:53.635s
```

```
172.16.3.101/24           server03          eth1
default         21m:21.209s
172.16.4.1/24             leaf04            br0
default         35m:45.120s
172.16.4.101/24           server04          eth2
default         29m:48.461s
172.17.0.1/16             oob-mgmt-server   docker0
default         22d:17h:40m:1s
192.168.0.11/24           leaf01            eth0
default         22d:17h:39m:56s
192.168.0.12/24           leaf02            eth0
default         22d:17h:40m:9s
192.168.0.13/24           leaf03            eth0
default         22d:17h:40m:4s
192.168.0.14/24           leaf04            eth0
default         22d:17h:40m:0s
192.168.0.21/24           spine01           eth0
default         22d:17h:40m:0s
192.168.0.22/24           spine02           eth0
default         22d:17h:40m:3s
192.168.0.254/24          oob-mgmt-server   eth1
default         22d:17h:40m:1s
192.168.0.31/24           server01          eth0
default         17h:43m:21s
192.168.0.32/24           server02          eth0
default         17h:41m:47s
192.168.0.33/24           server03          eth0
default         17h:41m:24s
192.168.0.34/24           server04          eth0
default         22d:17h:39m:59s
```

**Example: View IPv6 address information for all devices**

```
cumulus@switch:~$ netq show ipv6 addresses
Matching address records:
Address                   Hostname          Interface
VRF             Last Changed
------------------------- ----------------- -------------------------
--------------- -------------------------
fe80::203:ff:fe11:1101/64
server01          eth1                      default         47m:55.917
s
fe80::203:ff:fe22:2202/64
server02          eth2                      default         46m:24.908
s
fe80::203:ff:fe33:3301/64
server03          eth1                      default         45m:58.184
s
```

```
fe80::203:ff:fe44:4402/64
server04            eth2                       default          54m:25.264
s
fe80::4638:39ff:fe00:18/6
leaf02              br0                        default          1h:0m:31s
fe80::4638:39ff:fe00:1b/6
leaf03              swp52                      default          1h:0m:32s
fe80::4638:39ff:fe00:1c/6
spine02             swp3                       default          1h:0m:19s
fe80::4638:39ff:fe00:23/6
leaf03              br0                        default          1h:0m:26s
fe80::4638:39ff:fe00:24/6
leaf01              swp52                      default          1h:0m:44s
fe80::4638:39ff:fe00:25/6
spine02             swp1                       default          1h:0m:19s
fe80::4638:39ff:fe00:28/6
leaf02              swp51                      default          1h:0m:42s
fe80::4638:39ff:fe00:29/6
spine01             swp2                       default          1h:0m:29s
fe80::4638:39ff:fe00:2c/6
leaf04              br0                        default          1h:0m:18s
fe80::4638:39ff:fe00:3/64
leaf01              br0                        default          1h:0m:39s
fe80::4638:39ff:fe00:3b/6
leaf04              swp51                      default          1h:0m:23s
fe80::4638:39ff:fe00:3c/6
spine01             swp4                       default          1h:0m:27s
fe80::4638:39ff:fe00:46/6
leaf04              swp52                      default          1h:0m:22s
fe80::4638:39ff:fe00:47/6
spine02             swp4                       default          1h:0m:19s
fe80::4638:39ff:fe00:4f/6
leaf03              swp51                      default          1h:0m:36s
fe80::4638:39ff:fe00:50/6
spine01             swp3                       default          1h:0m:29s
fe80::4638:39ff:fe00:53/6
leaf01              swp51                      default          1h:0m:44s
fe80::4638:39ff:fe00:54/6
spine01             swp1                       default          1h:0m:29s
fe80::4638:39ff:fe00:57/6 oob-mgmt-
server   eth1                         default          22d:18h:4m:38s
fe80::4638:39ff:fe00:5d/6
leaf02              swp52                      default          1h:0m:40s
fe80::4638:39ff:fe00:5e/6
spine02             swp2                       default          1h:0m:19s
fe80::5054:ff:fe77:c277/6 oob-mgmt-
server   eth0                         default          22d:18h:4m:38s
fe80::a200:ff:fe00:11
/64  leaf01              eth0                       default          22d:
18h:4m:33s
```

```
fe80::a200:ff:fe00:12
/64  leaf02            eth0                      default         22d:
18h:4m:46s
fe80::a200:ff:fe00:13
/64  leaf03            eth0                      default         22d:
18h:4m:41s
fe80::a200:ff:fe00:14
/64  leaf04            eth0                      default         22d:
18h:4m:36s
fe80::a200:ff:fe00:21
/64  spine01           eth0                      default         22d:
18h:4m:37s
fe80::a200:ff:fe00:22
/64  spine02           eth0                      default         22d:
18h:4m:40s
fe80::a200:ff:fe00:31
/64  server01          eth0                      default         18h:
7m:58s
fe80::a200:ff:fe00:32
/64  server02          eth0                      default         18h:
6m:23s
fe80::a200:ff:fe00:33
/64  server03          eth0                      default         18h:
6m:1s
fe80::a200:ff:fe00:34
/64  server04          eth0                      default         22d:
18h:4m:36s
```

### Example: Filter IP Address Information for a Specific Interface

This example shows the IPv4 address information for the eth0 interface on all devices.

```
cumulus@switch:~$ netq show ip addresses eth0
Matching address records:
Address                   Hostname          Interface
VRF             Last Changed
------------------------- ----------------- -------------------------
--------------- -------------------------
10.0.0.254/32             oob-mgmt-server   eth0
default         22d:17h:40m:1s
192.168.0.11/24           leaf01            eth0
default         22d:17h:39m:56s
192.168.0.12/24           leaf02            eth0
default         22d:17h:40m:9s
192.168.0.13/24           leaf03            eth0
default         22d:17h:40m:4s
192.168.0.14/24           leaf04            eth0
default         22d:17h:40m:0s
192.168.0.21/24           spine01           eth0
default         22d:17h:40m:0s
```

```
192.168.0.22/24           spine02           eth0
default         22d:17h:40m:3s
192.168.0.31/24           server01          eth0
default         17h:43m:21s
192.168.0.32/24           server02          eth0
default         17h:41m:47s
192.168.0.33/24           server03          eth0
default         17h:41m:24s
192.168.0.34/24           server04          eth0
default         22d:17h:39m:59s
```

**Example: Filter IP Address Information for a Specific Device**

This example shows the IPv6 address information for the leaf01 switch.

```
cumulus@switch:~$ netq leaf01 show ipv6 addresses
Matching address records:
Address                   Hostname          Interface
VRF             Last Changed
------------------------- ----------------- -------------------------
--------------- -------------------------
fe80::4638:39ff:fe00:24/6
leaf01            swp52                     default         4h:18m:49s
fe80::4638:39ff:fe00:3/64
leaf01            br0                       default         4h:18m:45s
fe80::4638:39ff:fe00:53/6
leaf01            swp51                     default         4h:18m:50s
fe80::a200:ff:fe00:11
/64  leaf01            eth0                      default         22d:
21h:22m:39s
```

**Example: View Changes to IP Address Information**

This example shows the IPv4 address information that changed for all devices between 7 and 30 days ago.

```
cumulus@switch:~$ netq show ip addresses changes between 7d and 30d
Matching address records:
Address                   Hostname          Interface
VRF             DB State Last Changed
------------------------- ----------------- -------------------------
--------------- -------- -------------------------
192.168.0.11
/24           leaf01            eth0                      default
    Add      22d:20h:52m:30s
10.255.5.134
/24           leaf01            vagrant                   default
    Add      22d:20h:52m:30s
192.168.0.34
/24           server04          eth0                      default
    Add      22d:20h:52m:33s
```

```
192.168.0.14
/24            leaf04            eth0                      default
    Add      22d:20h:52m:34s
192.168.0.21
/24            spine01           eth0                      default
    Add      22d:20h:52m:35s
172.17.0.1/16              oob-mgmt-
server    docker0                   default         Add      22d:20h:
52m:35s
192.168.0.254/24           oob-mgmt-
server    eth1                      default         Add      22d:20h:
52m:35s
10.255.5.226/24            oob-mgmt-
server    eth0                      default         Add      22d:20h:
52m:35s
192.168.0.22
/24            spine02           eth0                      default
    Add      22d:20h:52m:37s
192.168.0.13
/24            leaf03            eth0                      default
    Add      22d:20h:52m:38s
10.255.5.191
/24            leaf03            vagrant                   default
    Add      22d:20h:52m:38s
192.168.0.12
/24            leaf02            eth0                      default
    Add      22d:20h:52m:43s
10.255.5.32
/24             leaf02            vagrant                   default
    Add      22d:20h:52m:43s
```

**Example: Obtain a Count of IP Addresses Used in Network**

This example shows the number of IPv4 and IPv6 addresses in the network.

```
cumulus@switch:~$ netq show ip addresses count
Count of matching address records: 33

cumulus@switch:~$ netq show ipv6 addresses count
Count of matching address records: 42
```

## View IP Neighbor Information

You can view the IPv4 and IPv6 neighbor information for all of your devices, including the interface port, MAC address, VRF assignment, and whether it learns the MAC address from the peer (remote=yes). Additionally, you can:

- view the information at an earlier point in time
- view changes that have occurred over time
- filter against a particular device, interface, address or VRF assignment

- obtain a count of all of the addresses

Each of these provides information for troubleshooting potential configuration and communication issues at the layer 3 level.

**Example: View IPv4 Neighbor Information for All Devices**

```
cumulus@switch:~$ netq show ip neighbors
Matching neighbor records:
IP Address                Hostname          Interface
MAC Address        VRF             Remote Last Changed
------------------------- ----------------- -------------------------
------------------ --------------- ------ -------------------------
10.255.5.1                oob-mgmt-server   eth0
52:54:00:0f:79:30  default         no     22d:21h:26m:33s
169.254.0.1               leaf01            swp51
44:38:39:00:00:54  default         no     4h:6m:17s
169.254.0.1               leaf01            swp52
44:38:39:00:00:25  default         no     4h:6m:18s
169.254.0.1               leaf02            swp51
44:38:39:00:00:29  default         no     4h:6m:16s
169.254.0.1               leaf02            swp52
44:38:39:00:00:5e  default         no     4h:6m:18s
169.254.0.1               leaf03            swp51
44:38:39:00:00:50  default         no     4h:6m:16s
169.254.0.1               leaf03            swp52
44:38:39:00:00:1c  default         no     4h:6m:17s
169.254.0.1               leaf04            swp51
44:38:39:00:00:3c  default         no     4h:6m:16s
169.254.0.1               leaf04            swp52
44:38:39:00:00:47  default         no     4h:6m:17s
169.254.0.1               spine01           swp1
44:38:39:00:00:53  default         no     4h:6m:17s
169.254.0.1               spine01           swp2
44:38:39:00:00:28  default         no     4h:6m:16s
169.254.0.1               spine01           swp3
44:38:39:00:00:4f  default         no     4h:6m:16s
169.254.0.1               spine01           swp4
44:38:39:00:00:3b  default         no     4h:6m:16s
169.254.0.1               spine02           swp1
44:38:39:00:00:24  default         no     4h:6m:8s
169.254.0.1               spine02           swp2
44:38:39:00:00:5d  default         no     4h:6m:8s
169.254.0.1               spine02           swp3
44:38:39:00:00:1b  default         no     4h:6m:7s
169.254.0.1               spine02           swp4
44:38:39:00:00:46  default         no     4h:6m:7s
192.168.0.11              oob-mgmt-server   eth1
a0:00:00:00:00:11  default         no     22d:21h:26m:33s
192.168.0.12              oob-mgmt-server   eth1
a0:00:00:00:00:12  default         no     22d:21h:26m:33s
```

```
192.168.0.13              oob-mgmt-server   eth1
a0:00:00:00:00:13  default         no     22d:21h:26m:33s
192.168.0.14              oob-mgmt-server   eth1
a0:00:00:00:00:14  default         no     22d:21h:26m:33s
192.168.0.21              oob-mgmt-server   eth1
a0:00:00:00:00:21  default         no     22d:21h:26m:33s
192.168.0.22              oob-mgmt-server   eth1
a0:00:00:00:00:22  default         no     22d:21h:26m:33s
192.168.0.253             oob-mgmt-server   eth1
a0:00:00:00:00:50  default         no     22d:21h:26m:33s
192.168.0.254             leaf01            eth0
44:38:39:00:00:57  default         no     22d:21h:26m:29s
192.168.0.254             leaf02            eth0
44:38:39:00:00:57  default         no     22d:21h:26m:41s
...
```

**Example: View IPv6 Neighbor Information for a Given Device.**

This example shows the IPv6 neighbors for leaf02 switch.

```
cumulus@switch$ netq leaf02 show ipv6 neighbors
Matching neighbor records:
IP Address                Hostname          Interface
MAC Address        VRF             Remote Last Changed
------------------------- ----------------- -------------------------
------------------ --------------- ------ -------------------------
fe80::203:ff:fe22:2202    leaf02            br0
00:03:00:22:22:02  default         no     4h:37m:59s
fe80::4638:39ff:fe00:29   leaf02            swp51
44:38:39:00:00:29  default         no     4h:41m:59s
fe80::4638:39ff:fe00:4    leaf02            eth0
44:38:39:00:00:04  default         no     22d:21h:46m:29s
fe80::4638:39ff:fe00:5e   leaf02            swp52
44:38:39:00:00:5e  default         no     4h:41m:58s
fe80::a200:ff:fe00:31     leaf02            eth0
a0:00:00:00:00:31  default         no     4h:37m:43s
fe80::a200:ff:fe00:32     leaf02            eth0
a0:00:00:00:00:32  default         no     4h:37m:56s
fe80::a200:ff:fe00:33     leaf02            eth0
a0:00:00:00:00:33  default         no     4h:37m:8s
fe80::a200:ff:fe00:34     leaf02            eth0
a0:00:00:00:00:34  default         no     4h:36m:40s
```

**Example: View Changes to IP Neighbors for All Devices**

This example shows changes to the IP neighbors for all devices in the last 5 days. If you want to see changes since the devices were added, remove the between keyword and values. If no changes are found, a *No matching neighbor records found* message shows as the result.

```
cumulus@switch:~$ netq show ip neighbors changes between now and 5d
```

```
Matching neighbor records:
IP Address                Hostname          Interface
MAC Address        VRF             Remote DB State   Last Changed
------------------------- ----------------- -------------------------
------------------ --------------- ------ ----------
-------------------------
169.254.0.1               exit02            swp51
44:38:39:00:00:22  default         no     Add        4d:20h:38m:6s
169.254.0.1               exit02            swp52
44:38:39:00:00:56  default         no     Add        4d:20h:38m:6s
169.254.0.1               exit02            swp44
44:38:39:00:00:3e  vrf1            no     Add        4d:20h:38m:6s
192.168.0.254             exit02            eth0
44:38:39:00:00:57  mgmt            no     Add        4d:20h:38m:6s
192.168.0.254             exit02            eth0
44:38:39:00:00:57  default         no     Del        4d:20h:38m:14s
10.255.0.1                exit02            vagrant
52:54:00:09:40:06  default         no     Del        4d:20h:38m:14s
169.254.0.1               exit01            swp44
44:38:39:00:00:07  vrf1            no     Add        4d:20h:38m:30s
192.168.0.254             exit01            eth0
44:38:39:00:00:57  mgmt            no     Add        4d:20h:38m:30s
169.254.0.1               exit01            swp52
44:38:39:00:00:5b  default         no     Add        4d:20h:38m:30s
169.254.0.1               exit01            swp51
44:38:39:00:00:0a  default         no     Add        4d:20h:38m:30s
192.168.0.254             exit01            eth0
44:38:39:00:00:57  default         no     Del        4d:20h:38m:38s
10.255.0.1                exit01            vagrant
52:54:00:09:40:06  default         no     Del        4d:20h:38m:38s
169.254.0.1               spine02           swp30
44:38:39:00:00:5a  default         no     Add        4d:20h:39m:30s
169.254.0.1               spine02           swp29
44:38:39:00:00:55  default         no     Add        4d:20h:39m:30s
192.168.0.254             spine02           eth0
44:38:39:00:00:57  mgmt            no     Add        4d:20h:39m:30s
192.168.0.254             spine02           eth0
44:38:39:00:00:57  default         no     Del        4d:20h:39m:38s
169.254.0.1               spine01           swp29
44:38:39:00:00:21  default         no     Add        4d:20h:39m:58s
192.168.0.254             spine01           eth0
44:38:39:00:00:57  mgmt            no     Add        4d:20h:39m:58s
169.254.0.1               spine01           swp30
44:38:39:00:00:09  default         no     Add        4d:20h:39m:58s
192.168.0.254             spine01           eth0
44:38:39:00:00:57  default         no     Del        4d:20h:40m:6s
192.168.0.254             leaf04            eth0
44:38:39:00:00:57  mgmt            no     Add        4d:20h:40m:41s
169.254.1.1               leaf04            peerlink.4094
44:38:39:00:00:2e  default         no     Add        4d:20h:40m:41s
192.168.0.254             leaf04            eth0
44:38:39:00:00:57  default         no     Del        4d:20h:40m:49s
```

```
192.168.0.11              leaf03            eth0
a0:00:00:00:00:11  mgmt             no     Add        4d:20h:44m:2s
169.254.1.2               leaf03            peerlink.4094
44:38:39:00:00:2f  default          no     Add        4d:20h:44m:2s
192.168.0.254             leaf03            eth0
44:38:39:00:00:57  mgmt             no     Add        4d:20h:44m:2s
192.168.0.254             leaf03            eth0
44:38:39:00:00:57  default          no     Del        4d:20h:44m:10s
192.168.0.254             leaf02            eth0
44:38:39:00:00:57  mgmt             no     Add        4d:20h:44m:51s
169.254.1.1               leaf02            peerlink.4094
44:38:39:00:00:10  default          no     Add        4d:20h:44m:51s
192.168.0.254             leaf02            eth0
44:38:39:00:00:57  default          no     Del        4d:20h:44m:59s
192.168.0.254             leaf01            eth0
44:38:39:00:00:57  mgmt             no     Add        4d:21h:31m:30s
169.254.1.2               leaf01            peerlink.4094
44:38:39:00:00:11  default          no     Add        4d:21h:31m:30s
192.168.0.13              leaf01            eth0
a0:00:00:00:00:13  mgmt             no     Add        4d:21h:31m:30s
192.168.0.254             leaf01            eth0
44:38:39:00:00:57  default          no     Del        4d:21h:31m:38s
```

## View IP Routes Information

You can view the IPv4 and IPv6 routes for all of your devices, including the IP address (with or without mask), the destination (by hostname) of the route, next hops available, VRF assignment, and whether a host is the owner of the route or MAC address. Additionally, you can:

- view the information at an earlier point in time
- view changes that have occurred over time
- filter against a particular address or VRF assignment
- obtain a count of all of the routes

Each of these provides information for troubleshooting potential configuration and communication issues at the layer 3 level.

**Example: View IP Routes for All Devices**

This example shows the IPv4 and IPv6 routes for all devices in the network.

```
cumulus@switch:~$ netq show ipv6 routes
Matching routes records:
Origin
VRF             Prefix                         Hostname          Nexth
ops                            Last Changed
------ --------------- ------------------------------
----------------- -----------------------------------
-------------------------
```

```
yes    default         ::
/0                           server04          lo
            6h:1m:52s
yes    default         ::
/0                           server03          lo
            6h:1m:53s
yes    default         ::
/0                           server01          lo
            6h:1m:54s
yes    default         ::
/0                           server02          lo
            6h:1m:53s

cumulus@switch:~$ netq show ip routes
Matching routes records:
Origin
VRF             Prefix                         Hostname          Nexth
ops                             Last Changed
------ --------------- ------------------------------
----------------- -----------------------------------
-------------------------
no     default         0.0.0.0
/0                      server02          192.168.0.254:
eth0                 6h:8m:55s
no     default         0.0.0.0
/0                      server04          192.168.0.254:
eth0                 6h:8m:54s
no     default         0.0.0.0
/0                      server01          192.168.0.254:
eth0                 6h:8m:55s
no     default         10.0.0.0
/8                     server03          172.16.3.1:
eth1                    6h:8m:54s
no     default         10.0.0.0
/8                     server02          172.16.2.1:
eth2                    6h:8m:55s
no     default         10.0.0.0
/8                     server04          172.16.4.1:
eth2                    6h:8m:54s
no     default         10.0.0.0
/8                     server01          172.16.1.1:
eth1                    6h:8m:55s
no     default         10.0.0.11
/32                   leaf04            169.254.0.1:
swp51,                 6h:15m:41s

  169.254.0.1: swp52
no     default         10.0.0.11
/32                   spine02           169.254.0.1:
swp1                   6h:15m:42s
```

```
no     default          10.0.0.11
/32                    spine01           169.254.0.1:
swp1                    6h:15m:48s
no     default          10.0.0.12
/32                    spine02           169.254.0.1:
swp2                    6h:15m:42s
no     default          10.0.0.12
/32                    leaf04            169.254.0.1:
swp51,                  6h:15m:41s

  169.254.0.1: swp52
no     default          10.0.0.12
/32                    spine01           169.254.0.1:
swp2                    6h:15m:48s
no     default          10.0.0.13
/32                    leaf04            169.254.0.1:
swp51,                  6h:15m:41s

  169.254.0.1: swp52
no     default          10.0.0.13
/32                    leaf01            169.254.0.1:
swp51,                  6h:15m:41s
...
```

**Example: View IP Routes for a Given IP Address**

This example shows the routes available for an IP address of 10.0.0.12/32.

```
cumulus@switch:~$ netq show ip routes 10.0.0.12
Matching routes records:
Origin
VRF             Prefix                         Hostname         Nexth
ops                            Last Changed
------ --------------- ------------------------------
----------------- -----------------------------------
-------------------------
no     default          10.0.0.12/32
leaf03            10.0.0.21: swp51, 10.0.0.22: swp52  5h:39m:57s
no     default          10.0.0.12/32
leaf01            10.0.0.21: swp51, 10.0.0.22: swp52  5h:39m:57s
no     default          10.0.0.12/32
leaf04            10.0.0.21: swp51, 10.0.0.22: swp52  5h:39m:57s
no     default          10.0.0.12/32
spine02           10.0.0.12: swp2                     5h:40m:1s
no     default          10.0.0.12/32
spine01           10.0.0.12: swp2                     5h:39m:56s
yes    default          10.0.0.12/32
leaf02            lo                                  5h:40m:21s
```

**Example: View IP Routes Owned by a Given Device**

This example shows the IPv4 routes that are owned by spine01 switch.

```
cumulus@switch:~$ netq spine01 show ip routes origin
Matching routes records:
Origin
VRF             Prefix                         Hostname          Nexth
ops                            Last Changed
------ --------------- ------------------------------
----------------- -----------------------------------
-------------------------
yes    default         10.0.0.21
/32                   spine01           lo
      23h:47m:23s
yes    default         192.168.0.0
/24                 spine01           eth0
    23d:16h:51m:28s
yes    default         192.168.0.21
/32                spine01           eth0
  23d:16h:51m:28s
```

### Example: View IP Routes for a Given Device at a Prior Time

This example show the IPv4 routes for spine01 switch about 24 hours ago.

```
cumulus@switch:~$ netq spine01 show ip routes around 24h
Matching routes records:
Origin
VRF             Prefix                         Hostname          Nexth
ops                            Last Changed
------ --------------- ------------------------------
----------------- -----------------------------------
-------------------------
no     default         10.0.0.11
/32                   spine01           169.254.0.1:
swp1                   3h:30m:12s
no     default         10.0.0.12
/32                   spine01           169.254.0.1:
swp2                   3h:30m:12s
no     default         10.0.0.13
/32                   spine01           169.254.0.1:
swp3                   3h:30m:11s
no     default         10.0.0.14
/32                   spine01           169.254.0.1:
swp4                   3h:30m:11s
no     default         172.16.1.0
/24                  spine01           169.254.0.1:
swp1                   3h:30m:13s
no     default         172.16.2.0
/24                  spine01           169.254.0.1:
swp2                   3h:30m:13s
```

```
no     default                172.16.3.0
/24                  spine01            169.254.0.1:
swp3                   3h:30m:13s
no     default                172.16.4.0
/24                  spine01            169.254.0.1:
swp4                   3h:30m:13s
yes    default                10.0.0.21
/32                   spine01            lo
      3h:46m:28s
yes    default                192.168.0.0
/24                 spine01            eth0
    22d:20h:50m:33s
yes    default                192.168.0.21
/32                spine01            eth0
  22d:20h:50m:33s
```

**Example: View the Number of IP Routes in Network**

This example shows the total number of IP routes for all devices in the network.

```
cumulus@switch:~$ netq show ip routes count
Count of matching routes records: 125

cumulus@switch:~$ netq show ipv6 routes count
Count of matching routes records: 5
```

## Monitor BGP Configuration

If you have BGP running on your switches and hosts, you can monitor its operation using the NetQ CLI. For each device, you can view its associated neighbors, ASN (autonomous system number), peer ASN, receive IP or EVPN address prefixes, and VRF assignment. Additionally, you can:

- view the information at an earlier point in time
- view changes that have occurred over time
- filter against a particular device, ASN, or VRF assignment
- validate it is operating correctly across the network

The `netq show bgp` command is used to obtain the BGP configuration information from the devices. The `netq check bgp` command is used to validate the configuration. The syntax of these commands is:

```
netq [<hostname>] show bgp [<bgp-session>|asn <number-asn>] [vrf
<vrf>] [around <text-time>] [json]
netq [<hostname>] show bgp [<bgp-session>|asn <number-asn>] [vrf
<vrf>] changes [between <text-time> and <text-endtime>] [json]

netq check bgp [vrf <vrf>] [around <text-time>] [json]
```

When entering a time value, you must include a numeric value *and* the unit of measure:

- d: day(s)
- h: hour(s)
- m: minute(s)
- s: second(s)
- now

For time ranges, the `<text-time>` is the most recent time and the `<text-endtime>` is the oldest time. The values do not have to have the same unit of measure.

## View BGP Configuration Information

NetQ enables you to view the BGP configuration of a single device or across all of your devices at once. You can filter the results based on an ASN, BGP session (IP address or interface name), or VRF assignment. You can view the configuration in the past and view changes made to the configuration within a given timeframe.

### Example: View BGP Configuration Information Across Network

This example shows the BGP configuration across all of your switches. In this scenario, BGP routing is configured between two spines and four leafs. Each leaf switch has a unique ASN and the spine switches share an ASN. The PfxRx column indicates that these devices have IPv4 address prefixes. The second and third values in this column indicate IPv6 and EVPN address prefixes when configured. This configuration was changed just over one day ago.

```
cumulus@switch:~$ netq show bgp
Matching bgp records:
Hostname          Neighbor                     VRF             ASN
    Peer ASN   PfxRx        Last Changed
----------------- ---------------------------- ---------------
---------- ---------- ------------ -------------------------
leaf01            swp51
(spine01)               default         65011      65020      7/-
/-        1d:3h:53m:22s
leaf01            swp52
(spine02)               default         65011      65020      7/-
/-        1d:3h:37m:20s
leaf02            swp51
(spine01)               default         65012      65020      7/-
/-        1d:3h:37m:30s
leaf02            swp52
(spine02)               default         65012      65020      7/-
/-        1d:3h:37m:21s
leaf03            swp51
(spine01)               default         65013      65020      7/-
/-        1d:3h:37m:30s
leaf03            swp52
(spine02)               default         65013      65020      7/-
/-        1d:3h:37m:21s
```

```
leaf04            swp51
(spine01)               default         65014      65020      7/-
/-        1d:3h:37m:29s
leaf04            swp52
(spine02)               default         65014      65020      7/-
/-        1d:3h:37m:20s
spine01           swp1
(leaf01)                 default         65020      65011      2/-
/-        1d:3h:53m:22s
spine01           swp2
(leaf02)                 default         65020      65012      2/-
/-        1d:3h:53m:22s
spine01           swp3
(leaf03)                 default         65020      65013      2/-
/-        1d:3h:53m:22s
spine01           swp4
(leaf04)                 default         65020      65014      2/-
/-        1d:3h:53m:22s
spine02           swp1
(leaf01)                 default         65020      65011      2/-
/-        1d:3h:37m:20s
spine02           swp2
(leaf02)                 default         65020      65012      2/-
/-        1d:3h:37m:20s
spine02           swp3
(leaf03)                 default         65020      65013      2/-
/-        1d:3h:37m:20s
spine02           swp4
(leaf04)                 default         65020      65014      2/-
/-        1d:3h:37m:19s
```

### Example: View BGP Configuration Information for a Given Device

This example shows the BGP configuration information for the spine02 switch. The switch is peered with swp1 on leaf01, swp2 on leaf02, and so on. Spine02 has an ASN of 65020 and each of the leafs have unique ASNs.

```
cumulus@switch:~$ netq spine02 show bgp
Matching bgp records:
Hostname          Neighbor                     VRF             ASN
    Peer ASN   PfxRx       Last Changed
----------------- ---------------------------- ---------------
---------- ---------- ------------ -------------------------
spine02           swp1
(leaf01)                 default         65020      65011      2/-
/-        1d:4h:55m:0s
spine02           swp2
(leaf02)                 default         65020      65012      2/-
/-        1d:4h:55m:0s
```

```
spine02           swp3
(leaf03)                 default         65020      65013      2/-
/-        1d:4h:55m:0s
spine02           swp4
(leaf04)                 default         65020      65014      2/-
/-        1d:4h:54m:59s
```

### Example: View BGP Configuration Information for a Given ASN

This example shows the BGP configuration information for ASN of *65013*. This ASN is associated with leaf03 and so the results show the BGP neighbors for that switch.

```
cumulus@switch:~$ netq show bgp asn 65013
Matching bgp records:
Hostname          Neighbor                     VRF             ASN
    Peer ASN   PfxRx        Last Changed
----------------- ---------------------------- ---------------
---------- ---------- ------------ -------------------------
leaf03            swp51
(spine01)               default         65013      65020      7/-
/-        1d:4h:54m:31s
leaf03            swp52
(spine02)               default         65013      65020      7/-
/-        1d:4h:54m:22s
```

### Example: View BGP Configuration Information for a Prior Time

This example shows the BGP configuration information as it was 12 hours earlier.

```
cumulus@switch:~$ netq show bgp around 12h
Matching bgp records:
Hostname          Neighbor                     VRF             ASN
    Peer ASN   PfxRx        Last Changed
----------------- ---------------------------- ---------------
---------- ---------- ------------ -------------------------
leaf01            swp51
(spine01)               default         65011      65020      7/-
/-        17h:29m:26s
leaf01            swp52
(spine02)               default         65011      65020      7/-
/-        17h:13m:24s
leaf02            swp51
(spine01)               default         65012      65020      7/-
/-        17h:13m:34s
leaf02            swp52
(spine02)               default         65012      65020      7/-
/-        17h:13m:25s
leaf03            swp51
(spine01)               default         65013      65020      7/-
/-        17h:13m:34s
```

```
leaf03            swp52
(spine02)               default         65013      65020      7/-
/-        17h:13m:25s
leaf04            swp51
(spine01)               default         65014      65020      7/-
/-        17h:13m:33s
leaf04            swp52
(spine02)               default         65014      65020      7/-
/-        17h:13m:24s
spine01           swp1
(leaf01)                 default         65020      65011      2/-
/-        17h:29m:26s
spine01           swp2
(leaf02)                 default         65020      65012      2/-
/-        17h:29m:26s
spine01           swp3
(leaf03)                 default         65020      65013      2/-
/-        17h:29m:26s
spine01           swp4
(leaf04)                 default         65020      65014      2/-
/-        17h:29m:26s
spine02           swp1
(leaf01)                 default         65020      65011      2/-
/-        17h:13m:24s
spine02           swp2
(leaf02)                 default         65020      65012      2/-
/-        17h:13m:24s
spine02           swp3
(leaf03)                 default         65020      65013      2/-
/-        17h:13m:24s
spine02           swp4
(leaf04)                 default         65020      65014      2/-
/-        17h:13m:23s
```

### Example: View BGP Configuration Changes

This example shows that BGP configuration changes were made about five days ago on this network.

```
cumulus@switch:~$ netq show bgp changes
Matching bgp records:
Hostname          Neighbor                     VRF
ASN        Peer ASN   PfxRx        DBState    Last Changed
----------------- ---------------------------- ---------------
---------- ---------- ------------ ----------
-------------------------
spine01           swp2(leaf02)                 default
65020      65012      2/-/10       Add        5d:1h:41m:31s
spine01           swp2(leaf02)                 default
65020      65012      2/-/10       Del        5d:1h:41m:31s
spine01           swp2(leaf02)                 default
65020      65012      2/-/10       Add        5d:1h:41m:31s
```

```
spine01           swp2(leaf02)                 default
65020      65012      2/-/10       Del        5d:1h:41m:31s
spine01           swp2(leaf02)                 default
65020      65012      2/-/10       Add        5d:1h:41m:31s
spine01           swp2(leaf02)                 default
65020      65012      2/-/10       Del        5d:1h:41m:31s
spine01           swp2(leaf02)                 default
65020      65012      2/-/10       Add        5d:1h:41m:31s
spine01           swp2(leaf02)                 default
65020      65012      2/-/10       Del        5d:1h:41m:31s
spine01           swp2(leaf02)                 default
65020      65012      2/-/10       Add        5d:1h:41m:31s
spine01           swp2(leaf02)                 default
65020      65012      2/-/10       Del        5d:1h:41m:31s
spine01           swp2(leaf02)                 default
65020      65012      2/-/10       Add        5d:1h:41m:31s
spine01           swp2(leaf02)                 default
65020      65012      2/-/10       Del        5d:1h:41m:31s
spine01           swp2(leaf02)                 default
65020      65012      2/-/10       Add        5d:1h:41m:31s
...
```

## Validate BGP Operation

A single command enables you to validate that all configured route peering is established across the network. The command checks for duplicate router IDs and sessions that are in an unestablished state. Either of these conditions trigger a configuration check failure. When a failure is found, the reason is identified in the output along with the time the issue occurred.

This example shows a check on the BGP operations that found no failed sessions.

```
cumulus@switch:~/$ netq check bgp
Total Nodes: 15, Failed Nodes: 0, Total Sessions: 16, Failed
Sessions: 0
```

This example shows a check on the BGP operations that found two failed sessions. The results indicate that BGP peering on leaf03 that connects to spine01 failed four minutes ago. The failure was caused by an interface failure on leaf03 which has lead to BGP hold timer expiration on spine01.

```
cumulus@switch:~$ netq check bgp
Total Sessions: 8 , Failed Sessions: 2
Node     Neighbor    Peer ID    Reason              Time
-------  ----------  ---------  ------------------  -------
leaf03   swp51       spine01    Interface down      4m ago
spine01  swp3        leaf03     Hold Timer Expired  4m ago
```

This example shows two failed BGP sessions because an interface is down and possibly because an RA was not configured.

```
cumulus@switch:~$ netq check bgp
Total Nodes: 10, Failed Nodes: 2, Total Sessions: 28 , Failed
Sessions: 2,
Hostname          VRF             Peer Name         Peer
Hostname     Reason                                            Last
Changed
----------------- --------------- ----------------- -----------------
---------------------------------------------
-------------------------
mlx-2700-03       default         uplink-1          spine-
1           RA not configured(?)                             0.116739s
spine-1           default         downlink-5        mlx-2700-
03       Interface down                                 0.116793s
```

This example shows four failed BGP sessions because peers are not configured and possibly because an RA was not configured.

```
cumulus@switch:~$ netq check bgp
Total Nodes: 10, Failed Nodes: 3, Total Sessions: 28 , Failed
Sessions: 4,
Hostname          VRF             Peer Name         Peer
Hostname     Reason                                            Last
Changed
----------------- --------------- ----------------- -----------------
---------------------------------------------
-------------------------
mlx-2700-03       default         uplink-1          spine-
1           Peer not configured                              4.59256s
mlx-2700-03       default         uplink-
2          unknown           RA not configured
(?)                          4.63093s
spine-1           default         downlink-5        mlx-2700-
03       Peer not configured                            0.155377s
spine-2           default         downlink-5        mlx-2700-
03       Peer not configured                            0.155410s
```

## Monitor OSPF Configuration

If you have OSPF running on your switches and hosts, you can monitor its operation using the NetQ CLI. For each device, you can view its associated interfaces, areas, peers, state, and type of OSPF running (numbered or unnumbered). Additionally, you can:

- view the information at an earlier point in time
- view changes that have occurred over time
- filter against a particular device, interface, or area
- validate it is operating correctly across the network

The `netq show ospf` command is used to obtain the OSPF configuration information from the devices. The `netq check ospf` command is used to validate the configuration. The syntax of these commands is:

```
netq [<hostname>] show ospf [<remote-interface>] [area <area-id>]
[around <text-time>] [json]
netq [<hostname>] show ospf [<remote-interface>] [area <area-id>]
changes [between <text-time> and <text-endtime>] [json]

netq check ospf [around <text-time>] [json]
```

> When entering a time value, you must include a numeric value *and* the unit of measure:
>
> - d: day(s)
> - h: hour(s)
> - m: minute(s)
> - s: second(s)
> - now
>
> For time ranges, the `<text-time>` is the most recent time and the `<text-endtime>` is the oldest time. The values do not have to have the same unit of measure.

## View OSPF Configuration Information

NetQ enables you to view the OSPF configuration of a single device or across all of your devices at once. You can filter the results based on a device, interface, or area. You can view the configuration in the past and view changes made to the configuration within a given timeframe.

**Example: View OSPF Configuration Information Across the Network**

This example shows all devices included in OSPF unnumbered routing, the assigned areas, state, peer and interface, and the last time this information was changed.

```
cumulus@switch:~$ netq show ospf

Matching ospf records:
Hostname          Interface                 Area         Type
    State      Peer Hostname     Peer Interface            Last
Changed
----------------- ------------------------- ------------
---------------- ---------- -----------------
------------------------- -------------------------
leaf01            swp51                     0.0.0.0      Unnumbered
    Full       spine01           swp1                      27.914477s
leaf01            swp52                     0.0.0.0      Unnumbered
    Full       spine02           swp1                      27.910094s
leaf02            swp51                     0.0.0.0      Unnumbered
    Full       spine01           swp2                      36.816204s
```

```
leaf02            swp52                     0.0.0.0      Unnumbered
    Full       spine02          swp2                       36.815804s
leaf03            swp51                     0.0.0.0      Unnumbered
    Full       spine01          swp3                       34.547961s
leaf03            swp52                     0.0.0.0      Unnumbered
    Full       spine02          swp3                       34.547727s
leaf04            swp51                     0.0.0.0      Unnumbered
    Full       spine01          swp4                       27.332121s
leaf04            swp52                     0.0.0.0      Unnumbered
    Full       spine02          swp4                       27.331475s
spine01           swp1                      0.0.0.0      Unnumbered
    Full       leaf01           swp51                      37.647986s
spine01           swp2                      0.0.0.0      Unnumbered
    Full       leaf02           swp51                      37.647565s
spine01           swp3                      0.0.0.0      Unnumbered
    Full       leaf03           swp51                      37.647786s
spine01           swp4                      0.0.0.0      Unnumbered
    Full       leaf04           swp51                      37.648211s
spine02           swp1                      0.0.0.0      Unnumbered
    Full       leaf01           swp52                      37.840344s
spine02           swp2                      0.0.0.0      Unnumbered
    Full       leaf02           swp52                      37.839967s
spine02           swp3                      0.0.0.0      Unnumbered
    Full       leaf03           swp52                      37.840188s
spine02           swp4                      0.0.0.0      Unnumbered
    Full       leaf04           swp52                      37.840626s
```

**Example: View OSPF Configuration Information for a Given Device**

This example show the OSPF configuration information for leaf01.

```
cumulus@switch:~$ netq leaf01 show ospf

Matching ospf records:
Hostname          Interface                 Area         Type
    State      Peer Hostname    Peer Interface             Last
Changed
----------------- ------------------------- ------------
---------------- ---------- -----------------
------------------------- -------------------------
leaf01            swp51                     0.0.0.0      Unnumbered
    Full       spine01          swp1                       8m:58.461s
leaf01            swp52                     0.0.0.0      Unnumbered
    Full       spine02          swp1                       8m:58.457s
```

**Example: View OSPF Configuration Information for a Given Interface**

This example shows the OSPF configuration for all devices with the swp51 interface.

```
cumulus@switch:~$ netq show ospf swp51
```

```
Matching ospf records:
Hostname          Interface                 Area         Type
    State      Peer Hostname    Peer Interface            Last
Changed
----------------- ------------------------- ------------
---------------- ---------- -----------------
------------------------- -------------------------
leaf01            swp51                     0.0.0.0      Unnumbered
    Full       spine01          swp1                      11m:10.639s
leaf02            swp51                     0.0.0.0      Unnumbered
    Full       spine01          swp2                      11m:19.540s
leaf03            swp51                     0.0.0.0      Unnumbered
    Full       spine01          swp3                      11m:17.272s
leaf04            swp51                     0.0.0.0      Unnumbered
    Full       spine01          swp4                      11m:10.567s
```

**Example: View OSPF Configuration Information at a Prior Time**

This example shows the OSPF configuration for all leaf switches about five minutes ago.

```
cumulus@switch:~$ netq leaf* show ospf around 5m

Matching ospf records:
Hostname          Interface                 Area         Type
    State      Peer Hostname    Peer Interface            Last
Changed
----------------- ------------------------- ------------
---------------- ---------- -----------------
------------------------- -------------------------
leaf01            swp51                     0.0.0.0      Unnumbered
    Full       spine01          swp1                      9m:10.128s
leaf01            swp52                     0.0.0.0      Unnumbered
    Full       spine02          swp1                      9m:10.124s
leaf02            swp51                     0.0.0.0      Unnumbered
    Full       spine01          swp2                      9m:19.305s
leaf02            swp52                     0.0.0.0      Unnumbered
    Full       spine02          swp2                      9m:19.301s
leaf03            swp51                     0.0.0.0      Unnumbered
    Full       spine01          swp3                      9m:16.762s
leaf03            swp52                     0.0.0.0      Unnumbered
    Full       spine02          swp3                      9m:16.762s
leaf04            swp51                     0.0.0.0      Unnumbered
    Full       spine01          swp4                      9m:9.546s
leaf04            swp52                     0.0.0.0      Unnumbered
    Full       spine02          swp4                      9m:9.545s
```

## Validate OSPF Operation

A single command, `netq check ospf`, enables you to validate that all configured route peering is established across the network. The command checks for:

- router ID conflicts, such as duplicate IDs
- links that are down, or have mismatched MTUs
- mismatched session parameters (hello timer, dead timer, area ids, and network type)

When peer information is not available, the command verifies whether OSPF is configured on the peer and if so, whether the service is disabled, shutdown, or not functioning.

All of these conditions trigger a configuration check failure. When a failure is found, the reason is identified in the output along with the time the issue occurred.

This example shows a check on the OSPF operations that found no failed sessions.

```
cumulus@switch:~$ netq check ospf
Total Sessions: 16, Failed Sessions: 0
```

This example shows a check on the OSPF operations that found two failed sessions. The results indicate the reason for the failure is a mismatched MTU for two links .

```
cumulus@switch:~$ netq check ospf
Total Nodes: 21, Failed Nodes: 2, Total Sessions: 40 , Failed
Sessions: 2,
Hostname          Interface                 PeerID
Peer IP
Reason                                        Last Changed
----------------- ------------------------- -------------------------
-------------------------
---------------------------------------------
-------------------------
spine-3           swp6                      0.0.0.23
27.0.0.23                 mtu mismatch, mtu
mismatch                    4.915650s
torc-22           swp5                      0.0.0.17
27.0.0.17                 mtu mismatch, mtu
mismatch                    11.452045s
```

## View Paths between Devices

You can view the available paths between two devices on the network currently and at a time in the past using their IPv4 or IPv6 addresses . You can perform the trace in only one direction or both, and view the output in one of three formats ( *json, pretty,* and *detail* ). JSON output provides the output in a JSON file format for ease of importing to other applications or software. Pretty output lines up the paths in a pseudo-graphical manner to help visualize multiple paths. Detail output is the default when not specified, and is useful for traces with higher hop counts where the pretty output wraps lines, making it harder to interpret the results. The detail output displays a table with a row per hop and a set of rows per path.

To view the paths, first identify the addresses for the source and destination devices using the `netq show ip addresses` command (see syntax above), and then use the `netq trace` command to see the available paths between those devices. The trace command syntax is:

```
netq trace <ip> from (<src-hostname>|<ip-src>) [vrf <vrf>] [around
<text-time>] [bidir] [json|detail|pretty] [debug]
```

> The syntax requires the destination device address first, *<ip>*, and then the source device address or hostname.
>
> The tracing function only knows about addresses that have already been learned. If you find that a path is invalid or incomplete, you may need to ping the identified device so that its address becomes known.

## View Paths between Two Switches with Pretty Output

This example first determines the IP addresses of the leaf01 and leaf03 switches, then shows the available paths between them. The results include a summary of the trace, including the total number of paths available, those with errors and warnings, and the MTU of the paths. In this case, the results are displayed in pseudo-graphical output.

```
cumulus@switch:~$ netq leaf01 show ip addresses
Matching address records:
Address                   Hostname          Interface
VRF             Last Changed
------------------------- ----------------- -------------------------
--------------- -------------------------
10.0.0.11/32              leaf01            lo
default         27m:3.870s
10.0.0.11/32              leaf01            swp51
default         27m:3.898s
10.0.0.11/32              leaf01            swp52
default         27m:3.877s
172.16.1.1/24             leaf01            br0
default         27m:3.653s
192.168.0.11/24           leaf01            eth0
default         33m:59.368s

cumulus@switch:~$ netq leaf03 show ip addresses
Matching address records:
Address                   Hostname          Interface
VRF             Last Changed
------------------------- ----------------- -------------------------
--------------- -------------------------
10.0.0.13/32              leaf03            lo
default         55m:43.224s
```

```
10.0.0.13/32              leaf03            swp51
default         55m:43.250s
10.0.0.13/32              leaf03            swp52
default         55m:43.230s
172.16.3.1/24             leaf03            br0
default         55m:43.754s
192.168.0.13/24           leaf03            eth0
default         1h:2m:47s

cumulus@switch:~$ netq trace 10.0.0.13 from 10.0.0.11 pretty
Number of Paths: 2
Number of Paths with Errors: 0
Number of Paths with Warnings: 0
Path MTU: 1500

 leaf01 swp52 -- swp1 spine02 swp3 -- swp52 leaf03 <lo>
        swp51 -- swp1 spine01 swp3 -- swp51 leaf03 <lo>
```

## View Forward and Reverse Paths between Two Switches with Pretty Output

Like the previous example, this shows the paths between leaf01 and leaf03 switches, but by adding the *bidir* keyword both the forward and reverse paths are presented. Optionally, you can use the source device's hostname to achieve the same results.

```
cumulus@switch:~$ netq trace 10.0.0.13 from 10.0.0.11 bidir pretty
Number of Paths: 2
Number of Paths with Errors: 0
Number of Paths with Warnings: 0
Path MTU: 1500

 leaf01 swp52 -- swp1 spine02 swp3 -- swp52 leaf03 <lo>
        swp51 -- swp1 spine01 swp3 -- swp51 leaf03 <lo>

Number of Paths: 2
Number of Paths with Errors: 0
Number of Paths with Warnings: 0
Path MTU: 1500

 leaf03 swp52 -- swp3 spine02 swp1 -- swp52 leaf01 <lo>
        swp51 -- swp3 spine01 swp1 -- swp51 leaf01 <lo>
```

## View Paths between Two Switches with Detailed Output

This example provides the same path information as the pretty output, but displays the information in a tabular output. In this case there, no VLAN is configured, so the related fields are left blank.

```
cumulus@switch:~$ netq trace 10.0.0.13 from 10.0.0.11 detail
Number of Paths: 2
Number of Paths with Errors: 0
Number of Paths with Warnings: 0
Path MTU: 1500

Id  Hop Hostname        InPort          InVlan
InTunnel              InRtrIf         InVRF           OutRtrIf
OutVRF          OutTunnel             OutPort         OutVlan
--- --- --------------- --------------- ------ ---------------------
--------------- --------------- --------------- ---------------
--------------------- --------------- -------
1   1   leaf01
                              swp52           default
              swp52
    2   spine02         swp1                                          s
wp1            default         swp3            default
              swp3
    3   leaf03          swp52                                         s
wp52           default         lo
--- --- --------------- --------------- ------ ---------------------
--------------- --------------- --------------- ---------------
--------------------- --------------- -------
2   1   leaf01
                              swp51           default
              swp51
    2   spine01         swp1                                          s
wp1            default         swp3            default
              swp3
    3   leaf03          swp51                                         s
wp51           default         lo
--- --- --------------- --------------- ------ ---------------------
--------------- --------------- --------------- ---------------
--------------------- --------------- -------
```

# Monitor Virtual Network Overlays

With NetQ, a network administrator can monitor virtual network components in the data center, including VXLAN, EVPN, and LNV software constructs. NetQ provides the ability to:

- Manage virtual constructs: view the performance and status of VXLANs, EVPN, and LNV
- Validate overlay communication paths

It helps answer questions such as:

- Is my overlay configured and operating correctly?
- Is my control plane configured correctly?
- Can device A reach device B?

## Contents

This topic describes how to...

## Monitor Virtual Extensible LANs

Virtual Extensible LANs (VXLANs) provide a way to create a virtual network on top of layer 2 and layer 3 technologies. It is intended for organizations, such as data centers, that require larger scale without additional infrastructure and more flexibility than is available with existing infrastructure equipment. With NetQ, you can monitor the current and historical configuration and status of your VXLANs using the following command:

```
netq [<hostname>] show vxlan [vni <text-vni>] [around <text-time>]
[json]
netq [<hostname>] show vxlan [vni <text-vni>] changes [between <text-
time> and <text-endtime>] [json]
netq [<hostname>] show interfaces type vxlan changes [between <text-
time> and <text-endtime>] [json]
```

When entering a time value, you must include a numeric value *and* the unit of measure:

- d: day(s)
- h: hour(s)
- m: minute(s)
- s: second(s)
- now

For time ranges, the `<text-time>` is the most recent time and the `<text-endtime>` is the oldest time. The values do not have to have the same unit of measure.

## View All VXLANs in Your Network

You can view a list of configured VXLANs for all devices, including the VNI (VXLAN network identifier), protocol, address of associated VTEPs (VXLAN tunnel endpoint), (replication list–what is this?), and the last time it was changed. You can also view VXLAN information for a given device by adding a hostname to the `show` command. You can filter the results by VNI.

This example shows all configured VXLANs across the network. In this network, there are three VNIs (13, 24, and 104001) associated with three VLANs (13, 24, 4001), EVPN is the virtual protocol deployed, and the configuration was last changed around 23 hours ago.

```
cumulus@switch:~$ netq show vxlan
Matching vxlan records:
Hostname          VNI        Protoc VTEP IP          VLAN
Replication List                    Last Changed
                             ol
----------------- ---------- ------ ---------------- ------
----------------------------------- -------------------------
exit01            104001     EVPN   10.0.0.41
4001                                       22h:50m:1s
exit02            104001     EVPN   10.0.0.42
4001                                       22h:49m:38s
leaf01            13         EVPN   10.0.0.112       13     10.0.0.134
(leaf04, leaf03)          23h:43m:1s
leaf01            24         EVPN   10.0.0.112       24     10.0.0.134
(leaf04, leaf03)          23h:43m:1s
leaf01            104001     EVPN   10.0.0.112
4001                                       23h:43m:1s
leaf02            13         EVPN   10.0.0.112       13     10.0.0.134
(leaf04, leaf03)          22h:56m:22s
leaf02            24         EVPN   10.0.0.112       24     10.0.0.134
(leaf04, leaf03)          22h:56m:22s
leaf02            104001     EVPN   10.0.0.112
4001                                       22h:56m:22s
leaf03            13         EVPN   10.0.0.134       13     10.0.0.112
(leaf02, leaf01)          22h:55m:33s
```

```
leaf03            24         EVPN   10.0.0.134       24     10.0.0.112
(leaf02, leaf01)          22h:55m:33s
leaf03            104001     EVPN   10.0.0.134
4001                                       22h:55m:33s
leaf04            13         EVPN   10.0.0.134       13     10.0.0.112
(leaf02, leaf01)          22h:52m:12s
leaf04            24         EVPN   10.0.0.134       24     10.0.0.112
(leaf02, leaf01)          22h:52m:12s
leaf04            104001     EVPN   10.0.0.134
4001                                       22h:52m:12s
```

This example shows the changes that have been made to VXLANs in your network in the last 24 hours. In this case, the EVPN configuration was added to each of the devices in the last 24 hours.

```
cumulus@switch:~$ netq show vxlan changes between now and 24h
Matching vxlan records:
Hostname          VNI        Protoc VTEP IP          VLAN
Replication List                    DB State   Last Changed
                             ol
----------------- ---------- ------ ---------------- ------
----------------------------------- ----------
-------------------------
exit02            104001     EVPN   10.0.0.42
4001                                       Add        23h:3m:8s
exit02            104001     EVPN   10.0.0.42
4001                                       Add        23h:3m:8s
exit02            104001     EVPN   10.0.0.42
4001                                       Add        23h:3m:8s
exit02            104001     EVPN   10.0.0.42
4001                                       Add        23h:3m:8s
exit02            104001     EVPN   10.0.0.42
4001                                       Add        23h:3m:8s
exit02            104001     EVPN   10.0.0.42
4001                                       Add        23h:3m:8s
exit02            104001     EVPN   10.0.0.42
4001                                       Add        23h:3m:8s
exit01            104001     EVPN   10.0.0.41
4001                                       Add        23h:3m:32s
exit01            104001     EVPN   10.0.0.41
4001                                       Add        23h:3m:32s
exit01            104001     EVPN   10.0.0.41
4001                                       Add        23h:3m:32s
exit01            104001     EVPN   10.0.0.41
4001                                       Add        23h:3m:32s
exit01            104001     EVPN   10.0.0.41
4001                                       Add        23h:3m:32s
exit01            104001     EVPN   10.0.0.41
4001                                       Add        23h:3m:32s
exit01            104001     EVPN   10.0.0.41
4001                                       Add        23h:3m:32s
```

```
exit01            104001     EVPN   10.0.0.41
4001                                       Add        23h:3m:32s
leaf04            104001     EVPN   10.0.0.134
4001                                       Add        23h:5m:43s
leaf04            104001     EVPN   10.0.0.134
4001                                       Add        23h:5m:43s
leaf04            104001     EVPN   10.0.0.134
4001                                       Add        23h:5m:43s
leaf04            104001     EVPN   10.0.0.134
4001                                       Add        23h:5m:43s
leaf04            104001     EVPN   10.0.0.134
4001                                       Add        23h:5m:43s
leaf04            104001     EVPN   10.0.0.134
4001                                       Add        23h:5m:43s
leaf04            104001     EVPN   10.0.0.134
4001                                       Add        23h:5m:43s
leaf04            13         EVPN   10.0.0.134       13
10.0.0.112()                        Add        23h:5m:43s
leaf04            13         EVPN   10.0.0.134       13
10.0.0.112()                        Add        23h:5m:43s
leaf04            13         EVPN   10.0.0.134       13
10.0.0.112()                        Add        23h:5m:43s
leaf04            13         EVPN   10.0.0.134       13
10.0.0.112()                        Add        23h:5m:43s
leaf04            13         EVPN   10.0.0.134       13
10.0.0.112()                        Add        23h:5m:43s
leaf04            13         EVPN   10.0.0.134       13
10.0.0.112()                        Add        23h:5m:43s
leaf04            13         EVPN   10.0.0.134       13
10.0.0.112()                        Add        23h:5m:43s
...
```

Consequently, if you looked for the VXLAN configuration and status for last week, you would find either another configuration or no configuration. This example shows that no VXLAN configuration was present.

```
cumulus@switch:~$ netq show vxlan around 7d

No matching vxlan records found
```

You can filter the list of VXLANs to view only those associated with a particular VNI. This example shows the configured VXLANs for *VNI 24*.

```
cumulus@switch:~$ netq show vxlan vni 24
Matching vxlan records:
Hostname          VNI        Protoc VTEP IP          VLAN
Replication List                    Last Changed
                             ol
----------------- ---------- ------ ---------------- ------
----------------------------------- -------------------------
```

```
leaf01            24        EVPN   10.0.0.112       24      10.0.0.134
(leaf04, leaf03)         1d:0h:4m:12s
leaf02            24        EVPN   10.0.0.112       24      10.0.0.134
(leaf04, leaf03)         23h:17m:33s
leaf03            24        EVPN   10.0.0.134       24      10.0.0.112
(leaf02, leaf01)         23h:16m:44s
leaf04            24        EVPN   10.0.0.134       24      10.0.0.112
(leaf02, leaf01)         23h:13m:23s
```

## View the Interfaces Associated with VXLANs

You can view detailed information about the VXLAN interfaces using the netq show interface command. You can also view this information for a given device by adding a hostname to the `show` command. This example shows the detailed VXLAN interface information for the leaf02 switch.

```
cumulus@switch:~$ netq leaf02 show interfaces type vxlan
Matching link records:
Hostname          Interface                 Type
State      VRF             Details                            Last
Changed
----------------- ------------------------- ----------------
---------- --------------- -----------------------------------
-------------------------
leaf02            vni13                     vxlan
up         default         VNI: 13, PVID: 13, Master: bridge,  23h:
23m:11s

VTEP: 10.0.0.112, MTU: 9000
leaf02            vni24                     vxlan
up         default         VNI: 24, PVID: 24, Master: bridge,  23h:
23m:11s

VTEP: 10.0.0.112, MTU: 9000
leaf02            vxlan4001                 vxlan
up         default         VNI: 104001, PVID: 4001,            23h:
23m:11s

Master: bridge, VTEP: 10.0.0.112,

MTU: 1500
```

# Monitor EVPN

EVPN (Ethernet Virtual Private Network) enables network administrators in the data center to deploy a virtual layer 2 bridge overlay on top of layer 3 IP networks creating access, or tunnel, between two locations. This connects devices in different layer 2 domains or sites running VXLANs and their associated underlays.

With NetQ, you can monitor the configuration and status of the EVPN setup using the **`netq show evpn`** command. You can filter the EVPN information by a VNI (VXLAN network identifier), and view the current information or for a time in the past. The command also enables visibility into changes that have occurred in the configuration during a specific timeframe. The syntax for the command is:

```
netq [<hostname>] show evpn [vni <text-vni>] changes [between <text-
time> and <text-endtime>] [json]
netq [<hostname>] show evpn [vni <text-vni>] [around <text-time>]
[json]
```

ⓘ When entering a time value, you must include a numeric value *and* the unit of measure:

- d: day(s)
- h: hour(s)
- m: minute(s)
- s: second(s)
- now

For time ranges, the `<text-time>` is the most recent time and the `<text-endtime>` is the oldest time. The values do not have to have the same unit of measure.

For more information about and configuration of EVPN in your data center, refer to the Cumulus Linux EVPN topic.

## View the Status of EVPN

You can view the configuration and status of your EVPN overlay across your network or for a particular device. This example shows the configuration and status for all devices, including the associated VNI, VTEP address, the import and export route (showing the BGP ASN and VNI path), and the last time a change was made for each device running EVPN. Use the *hostname* variable to view the configuration and status for a single device.

```
cumulus@switch:~$ netq show evpn
Matching evpn records:
Hostname          VNI        VTEP IP          In Kernel Export
RT        Import RT        Last Changed
----------------- ---------- ---------------- ---------
---------------- ---------------- -------------------------
exit01            104001     10.0.0.41        yes       65041:
104001     65041:104001     3d:17h:20m:10s
exit02            104001     10.0.0.42        yes       65042:
104001     65042:104001     3d:17h:19m:48s
leaf01            13         10.0.0.112       yes       65011:
13         65011:13         3d:18h:13m:12s
leaf01            24         10.0.0.112       yes       65011:
24         65011:24         3d:18h:13m:12s
```

```
leaf01            104001     10.0.0.112       yes       65011:
104001     65011:104001     3d:18h:13m:12s
leaf02            13         10.0.0.112       yes       65012:
13         65012:13         3d:17h:26m:31s
leaf02            24         10.0.0.112       yes       65012:
24         65012:24         3d:17h:26m:31s
leaf02            104001     10.0.0.112       yes       65012:
104001     65012:104001     3d:17h:26m:31s
leaf03            13         10.0.0.134       yes       65013:
13         65013:13         3d:17h:25m:42s
leaf03            24         10.0.0.134       yes       65013:
24         65013:24         3d:17h:25m:42s
leaf03            104001     10.0.0.134       yes       65013:
104001     65013:104001     3d:17h:25m:42s
leaf04            13         10.0.0.134       yes       65014:
13         65014:13         3d:17h:22m:22s
leaf04            24         10.0.0.134       yes       65014:
24         65014:24         3d:17h:22m:22s
leaf04            104001     10.0.0.134       yes       65014:
104001     65014:104001     3d:17h:22m:22s
```

## View the Status of EVPN for a Given VNI

You can filter the full device view to focus on a single VNI. This example only shows the EVPN configuration and status for VNI 24.

```
cumulus@switch:~$ netq show evpn vni 24
Matching evpn records:
Hostname          VNI        VTEP IP          In Kernel Export
RT        Import RT        Last Changed
----------------- ---------- ---------------- ---------
---------------- ---------------- -------------------------
leaf01            24         10.0.0.112       yes       65011:
24         65011:24         3d:18h:37m:23s
leaf02            24         10.0.0.112       yes       65012:
24         65012:24         3d:17h:50m:43s
leaf03            24         10.0.0.134       yes       65013:
24         65013:24         3d:17h:49m:53s
leaf04            24         10.0.0.134       yes       65014:
24         65014:24         3d:17h:46m:34s
```

## View Changes to the EVPN Configuration

You can view the changes that have been made to your EVPN configuration within the last hour or within a given timeframe. Perhaps you are seeing errors related to EVPN and you suspect a configuration change may be the cause. You can find out if any changes were made and when using the *changes* keyword. This example shows the changes made in the last hour (none) and the changes made in the last 7 days (the addition of EVPN on the leaf switches and exit switches).

```
cumulus@switch:~$ netq show evpn changes between now and 7d
Matching evpn records:
Hostname          VNI        VTEP IP          In Kernel Export
RT        Import RT        DB State   Last Changed
----------------- ---------- ---------------- ---------
---------------- ---------------- ---------- -------------------------
exit02            104001     10.0.0.42        yes       65042:
104001     65042:104001     Add        3d:17h:46m:59s
exit01            104001     10.0.0.41        yes       65041:
104001     65041:104001     Add        3d:17h:47m:21s
leaf04            104001     10.0.0.134       yes       65014:
104001     65014:104001     Add        3d:17h:49m:33s
leaf04            13         10.0.0.134       yes       65014:
13         65014:13         Add        3d:17h:49m:33s
leaf04            24         10.0.0.134       yes       65014:
24         65014:24         Add        3d:17h:49m:33s
leaf03            104001     10.0.0.134       yes       65013:
104001     65013:104001     Add        3d:17h:52m:53s
leaf03            13         10.0.0.134       yes       65013:
13         65013:13         Add        3d:17h:52m:53s
leaf03            24         10.0.0.134       yes       65013:
24         65013:24         Add        3d:17h:52m:53s
leaf02            104001     10.0.0.112       yes       65012:
104001     65012:104001     Add        3d:17h:53m:42s
leaf02            13         10.0.0.112       yes       65012:
13         65012:13         Add        3d:17h:53m:42s
leaf02            24         10.0.0.112       yes       65012:
24         65012:24         Add        3d:17h:53m:42s
leaf01            104001     10.0.0.112       yes       65011:
104001     65011:104001     Add        3d:18h:40m:23s
leaf01            13         10.0.0.112       yes       65011:
13         65011:13         Add        3d:18h:40m:23s
leaf01            24         10.0.0.112       yes       65011:
24         65011:24         Add        3d:18h:40m:23s
```

## Monitor LNV

Lightweight Network Virtualization (LNV) is a technique for deploying VXLANs without a central controller on bare metal switches. LNV enables data center network administrators and operators to create a data path between bridges on top of a layer 3 fabric. With NetQ, you can monitor the configuration and status of the LNV setup using the `netq show lnv` command. You can view the current information or for a time in the past. The command also enables visibility into changes that have occurred in the configuration during a specific timeframe. The syntax for the command is:

```
netq [<hostname>] show lnv [around <text-time>] [json]
netq [<hostname>] show lnv changes [between <text-time> and <text-
endtime>] [json]
```

## View LNV Status

You can view the configuration and status of your LNV overlay across your network or for a particular device. This example shows the configuration and status of LNV across the network, including the role each node plays, replication mode, number of peers and VNIs, and the last time the configuration was changed.

```
cumulus@switch:~$ netq show lnv
Matching LNV session records are:
Hostname          Role       ReplMode State      #Peers #VNIs  Last
Changed
----------------- ---------- -------- ---------- ------ ------
-------------------------
spine-1           SND        HER      up         3      6      45m:
26.865s
spine-2           SND        HER      up         3      6      45m:
23.299s
spine-3           SND        HER      up         3      6      45m:
21.847s
tor-1             RD         HER      up         4      6      45m:
25.335s
tor-2             RD         HER      up         4      6      45m:
6.495s
torc-
11           RD         HER      up         0      0      17.258785s
torc-12           RD         HER      up         4      6      45m:
16.800s
torc-21           RD         HER      up         4      6      45m:
29.437s
torc-22           RD         HER      up         4      6      45m:
11.440s
```

## View LNV Status in the Past

You can view the status in the past using either the around or changes keywords. This example shows the status of LNV about 30 minutes ago.

```
cumulus@switch:~$ netq show lnv around 30m
Matching LNV session records are:
Hostname          Role       ReplMode State      #Peers #VNIs  Last
Changed
----------------- ---------- -------- ---------- ------ ------
-------------------------
spine-1           SND        HER      up         3      6      45m:
37.973s
spine-2           SND        HER      up         3      6      45m:
34.407s
spine-3           SND        HER      up         3      6      45m:
32.955s
```

```
tor-1             RD         HER      up         4      6      45m:
36.443s
tor-2             RD         HER      up         4      6      45m:
17.603s
torc-11           RD         HER      up         4      6      45m:
46.696s
torc-12           RD         HER      up         4      6      45m:
27.908s
torc-21           RD         HER      up         4      6      45m:
40.546s
torc-22           RD         HER      up         4      6      45m:
22.548s
```

For more information about and configuration of LNV, refer to the Cumulus Linux LNV Overview topic.

## View Communication Paths between Devices

You can view the available paths between devices that communicate over virtual constructs using the netq trace command. The syntax of this command is:

```
netq trace <mac> [vlan <1-4096>] from (<src-hostname>|<ip-src>) [vrf
<vrf>] [around <text-time>] [bidir] [json|detail|pretty] [debug]
netq trace <ip> from (<src-hostname>|<ip-src>) [vrf <vrf>] [around
<text-time>] [bidir] [json|detail|pretty] [debug]
```

This example shows the available paths between leaf01 and leaf03 which are connected through an EVPN overlay. This example uses the default presentation of *detail* output.

```
cumulus@switch:~$ netq trace 10.0.0.13 from leaf01
Number of Paths: 2
Number of Paths with Errors: 0
Number of Paths with Warnings: 0
Path MTU: 9216

Id  Hop Hostname    InPort          InTun, RtrIf    OutRtrIf,
Tun   OutPort
--- --- ----------- --------------- --------------- ---------------
---------------
1   1   leaf01                                      swp52           sw
p52
    2   spine02     swp1            swp1            swp3            sw
p3
    3   leaf03      swp52           swp52                           lo
--- --- ----------- --------------- --------------- ---------------
---------------
2   1   leaf01                                      swp51           sw
p51
```

```
    2   spine01     swp1            swp1            swp3            sw
p3
    3   leaf03      swp51           swp51                           lo
--- --- ----------- --------------- --------------- ---------------
---------------
```

You can also view the paths in both directions, to and from the two devices, as shown in this example using the *pretty* output option.

```
cumulus@switch:~$ netq trace 10.0.0.13 from leaf01 bidir pretty
Number of Paths: 2
Number of Paths with Errors: 0
Number of Paths with Warnings: 0
Path MTU: 9216

 leaf01 swp52 -- swp1 spine02 swp3 -- swp52 leaf03 lo
 leaf01 swp51 -- swp1 spine01 swp3 -- swp51 leaf03 lo

Number of Paths: 2
Number of Paths with Errors: 0
Number of Paths with Warnings: 0
Path MTU: 9216

 leaf03 swp52 -- swp3 spine02 swp1 -- swp52 leaf01
 leaf03 swp51 -- swp3 spine01 swp1 -- swp51 leaf01
```

For more information about the trace command, run the `netq example trace` command.

```
cumulus@switch:~$ netq example trace

Control Path Trace
==================

Commands
========
   netq trace <mac> [vlan <1-4096>] from (<src-hostname>|<ip-src>)
[vrf <vrf>] [around <text-time>] [bidir] [json|detail|pretty] [debug]
   netq trace <ip> from (<src-hostname>|<ip-src>) [vrf <vrf>] [around
<text-time>] [bidir] [json|detail|pretty] [debug]

Usage
=====
netq trace provides control path tracing (no real packets are sent)
from
a specified source to a specified destination. The trace covers
complete
end-to-end path tracing including bridged, routed and Vxlan overlay
paths.
```

```
ECMP is supported as well as checking for forwarding loops, MTU
consistency
across all paths, and VLAN consistency across all paths.  Reverse path
trace is also available as an option.

Trace output can be generated in multiple formats.
...
```

# Monitor Linux Hosts

Running NetQ on Linux hosts provides unprecedented network visibility, giving the network operator a complete view of the entire infastrucutre's network connectivity instead of just from the network devices.

The NetQ Agent is supported on the following Linux hosts:

- CentOS 7
- Red Hat Enterprise Linux 7.1
- Ubuntu 16.04

You need to install the OS-specific NetQ metapack on every host you want to monitor with NetQ.

The NetQ Agent monitors the following on Linux hosts:

- netlink
- Layer 2: LLDP and VLAN-aware bridge
- Layer 3: IPv4, IPv6
- Routing on the Host: BGP, OSPF
- systemctl for services
- Docker containers — refer to the Monitor Container Environments (see page 145) topic

Using NetQ on a Linux host is the same as using it on a Cumulus Linux switch. For example, if you want to check LLDP neighbor information about a given host, run:

```
cumulus@switch:~$ netq server01 show lldp
Matching lldp records:
Hostname          Interface                 Peer Hostname     Peer
Interface            Last Changed
----------------- ------------------------- -----------------
------------------------- -------------------------
server01          eth0                      oob-mgmt-switch
swp2                      5d:22h:44m:44s
server01          eth1                      leaf01
swp1                      3d:23h:30m:37s
server01          eth2                      leaf02
swp1                      3d:23h:28m:50s
```

Then, to see LLDP from the switch's perspective:

```
cumulus@switch:~$ netq leaf01 show lldp
Matching lldp records:
Hostname          Interface                 Peer Hostname     Peer
Interface            Last Changed
----------------- ------------------------- -----------------
------------------------- -------------------------
leaf01            eth0                      oob-mgmt-switch
swp6                      5d:22h:45m:35s
```

```
leaf01            swp1                      server01
eth1                      5d:22h:39m:53s
leaf01            swp2                      server02
eth1                      3d:19h:23m:9s
leaf01            swp49                     leaf02
swp49                     4d:0h:30m:34s
leaf01            swp50                     leaf02
swp50                     4d:0h:30m:34s
leaf01            swp51                     spine01
swp1                      5d:22h:40m:24s
leaf01            swp52                     spine02
swp1                      5d:22h:40m:24s
```

To get the routing table for a server:

```
cumulus@server01:~$ netq server01 show ip route
Matching routes records:
Origin VRF             Prefix
Hostname          Nexthops                             Last Changed
------ --------------- ------------------------------
----------------- -----------------------------------
-------------------------
no     default         10.2.4.0/24
server01          10.1.3.1: uplink                     3d:23h:31m:8s
no     default         172.16.1.0/24
server01          10.1.3.1: uplink                     3d:23h:31m:8s
yes    default         10.1.3.0/24
server01          uplink                               3d:23h:31m:8s
yes    default         10.1.3.101/32
server01          uplink                               3d:23h:31m:8s
yes    default         192.168.0.0/24
server01          eth0                                 3d:23h:31m:8s
yes    default         192.168.0.31/32
server01          eth0                                 3d:23h:31m:8s
```

# Monitor Container Environments

The NetQ Agent monitors container environments the same way it monitors physical servers. There is no special implementation. The NetQ Agent pulls data from the container as it would pull data from a Cumulus Linux switch or Linux host. It can be installed on a Linux server or in a Linux VM. NetQ Agent integrates with container orchestrators including Kubernetes and Docker Swarm.

NetQ monitors many aspects of containers on your network, including their:

- **Identity**: The NetQ agent tracks every container's IP and MAC address, name, image, and more. NetQ can locate containers across the fabric based on a container's name, image, IP or MAC address, and protocol and port pair.
- **Port mapping on a network**: The NetQ agent tracks protocol and ports exposed by a container. NetQ can identify containers exposing a specific protocol and port pair on a network.
- **Connectivity**: NetQ can provide information on network connectivity for a container, including adjacency, and can identify containers that can be affected by a top of rack switch.

NetQ helps answer questions such as:

- Where is this container located?
- Open ports? What image is being used?
- Which containers are part of this service? How are they connected?

## Contents

This topic describes how to...

- View All Containers in a Network (see page 180)
- View Container Adjacency (see page 181)
- Show Container-Specific Information (see page 184)
- Show Containers with a Specific Image (see page 184)
- Show Container Connectivity (see page 187)
- Check Network Traffic over a Given Protocol (see page 190)
- Show Docker Swarm Clusters and Networks (see page 191)
- Show Docker Service Connectivity and Impact (see page 193)
- View Docker Configuration in the Past (see page 195)

# Use NetQ with Kubernetes Clusters

The NetQ Agent interfaces with a Kubernetes API server and listens to Kubernetes events. The NetQ Agent monitors network identity and physical network connectivity of Kubernetes resources like Pods, Daemon sets, Service, and so forth. NetQ works with any container network interface (CNI), such as Calico or Flannel.

The NetQ Kubernetes integration enables network administrators to:

- Identify and locate pods, deployment, replica-set and services deployed within the network using IP, name, label, and so forth.
- Track network connectivity of all pods of a service, deployment and replica set.
- Locate what pods have been deployed adjacent to a top of rack (ToR) switch.
- Check what pod, services, replica set or deployment can be impacted by a specific ToR switch.

NetQ also helps network administrators identify changes within a Kubernetes cluster and determine if such changes had an adverse effect on the network performance (caused by a noisy neighbor for example). Additionally, NetQ helps the infrastructure administrator determine how Kubernetes workloads are distributed within a network.

## Requirements

The NetQ Agent supports Kubernetes version 1.9.2 or later.

Due to the higher memory requirements to run containers, Cumulus Networks recommends you run the NetQ Telemetry Server (TS) on a host with at least 32G RAM. For more information, refer to the How Far Back in Time Can You Travel? (see page 230) topic.

## Command Summary

There is a large set of commands available to monitor Kubernetes configurations, including the ability to monitor clusters, nodes, daemon-set, deployment, pods, replication, and services:

```
netq [<hostname>] show kubernetes cluster [name <kube-cluster-name>]
[around <text-time>] [json]
netq [<hostname>] show kubernetes cluster [name <kube-cluster-name>]
changes [between <text-time> and <text-endtime>] [json]
netq [<hostname>] show kubernetes node [components] [name <kube-node-
name>] [cluster <kube-cluster-name> ] [label <kube-node-label>]
[around <text-time>] [json]
```

```
netq [<hostname>] show kubernetes node [components] [name <kube-node-
name>] [cluster <kube-cluster-name> ] [label <kube-node-label>]
changes [between <text-time> and <text-endtime>] [json]

netq [<hostname>] show kubernetes daemon-set [name <kube-ds-name>]
[cluster <kube-cluster-name>] [namespace <namespace>] [label <kube-ds-
label>] [around <text-time>] [json]
netq [<hostname>] show kubernetes daemon-set [name <kube-ds-name>]
[cluster <kube-cluster-name>] [namespace <namespace>] [label <kube-ds-
label>] connectivity [around <text-time>] [json]
netq [<hostname>] show kubernetes daemon-set [name <kube-ds-name>]
[cluster <kube-cluster-name>] [namespace <namespace>] [label <kube-ds-
label>] changes [between <text-time> and <text-endtime>] [json]

netq [<hostname>] show kubernetes deployment [name <kube-deployment-
name>] [cluster <kube-cluster-name>] [namespace <namespace>] [label
<kube-deployment-label>] [around <text-time>] [json]
netq [<hostname>] show kubernetes deployment [name <kube-deployment-
name>] [cluster <kube-cluster-name>] [namespace <namespace>] [label
<kube-deployment-label>] connectivity [around <text-time>] [json]
netq [<hostname>] show kubernetes deployment [name <kube-deployment-
name>] [cluster <kube-cluster-name>] [namespace <namespace>] [label
<kube-deployment-label>] changes [between <text-time> and <text-
endtime>] [json]
netq <hostname> show impact kubernetes deployment [master <kube-
master-node>] [name <kube-deployment-name>] [cluster <kube-cluster-
name>] [namespace <namespace>] [label <kube-deployment-label>]
[around <text-time>] [json]Requirements

netq [<hostname>] show kubernetes pod [name <kube-pod-name>] [cluster
<kube-cluster-name> ] [namespace <namespace>] [label <kube-pod-
label>] [pod-ip <kube-pod-ipaddress>] [node <kube-node-name>] [around
<text-time>] [json]
netq [<hostname>] show kubernetes pod [name <kube-pod-name>] [cluster
<kube-cluster-name> ] [namespace <namespace>] [label <kube-pod-
label>] [pod-ip <kube-pod-ipaddress>] [node <kube-node-name>] changes
[between <text-time> and <text-endtime>] [json]

netq [<hostname>] show kubernetes replication-controller [name <kube-
rc-name>] [cluster <kube-cluster-name>] [namespace <namespace>]
[label <kube-rc-label>] [around <text-time>] [json]
netq [<hostname>] show kubernetes replication-controller [name <kube-
rc-name>] [cluster <kube-cluster-name>] [namespace <namespace>]
[label <kube-rc-label>] changes [between <text-time> and <text-
endtime>] [json]
netq [<hostname>] show kubernetes replica-set [name <kube-rs-name>]
[cluster <kube-cluster-name>] [namespace <namespace>] [label <kube-rs-
label>] [around <text-time>] [json]
netq [<hostname>] show kubernetes replica-set [name <kube-rs-name>]
[cluster <kube-cluster-name>] [namespace <namespace>] [label <kube-rs-
label>] connectivity [around <text-time>] [json]
```

```
netq [<hostname>] show kubernetes replica-set [name <kube-rs-name>]
[cluster <kube-cluster-name>] [namespace <namespace>] [label <kube-rs-
label>] changes [between <text-time> and <text-endtime>] [json]
netq <hostname> show impact kubernetes replica-set [master <kube-
master-node>] [name <kube-rs-name>] [cluster <kube-cluster-name>]
[namespace <namespace>] [label <kube-rs-label>] [around <text-time>]
[json]

netq [<hostname>] show kubernetes service [name <kube-service-name>]
[cluster <kube-cluster-name>] [namespace <namespace>] [label <kube-
service-label>] [service-cluster-ip <kube-service-cluster-ip>]
[service-external-ip <kube-service-external-ip>] [around <text-time>]
[json]
netq [<hostname>] show kubernetes service [name <kube-service-name>]
[cluster <kube-cluster-name>] [namespace <namespace>] [label <kube-
service-label>] [service-cluster-ip <kube-service-cluster-ip>]
[service-external-ip <kube-service-external-ip>] connectivity [around
<text-time>] [json]
netq [<hostname>] show kubernetes service [name <kube-service-name>]
[cluster <kube-cluster-name>] [namespace <namespace>] [label <kube-
service-label>] [service-cluster-ip <kube-service-cluster-ip>]
[service-external-ip <kube-service-external-ip>] changes [between
<text-time> and <text-endtime>] [json]
netq <hostname>  show impact kubernetes service [master <kube-master-
node>] [name <kube-service-name>] [cluster <kube-cluster-name>]
[namespace <namespace>] [label <kube-service-label>] [service-cluster-
ip <kube-service-cluster-ip>] [service-external-ip <kube-service-
external-ip>] [around <text-time>] [json]
```

## Enable Kubernetes Monitoring

For NetQ to monitor the containers on a host, you must configure the following on the Kubernetes master node:

1. Configure the host to point to the TS by its IP address. See the Install NetQ topic for details.
2. Enable Kubernetes monitoring by NetQ. You can specify a polling period between 10 and 120 seconds; 15 seconds is the default.

   ```
   cumulus@host:~$ netq config add agent kubernetes-monitor poll-
   period 20
   Successfully added kubernetes monitor. Please restart netq-agent.
   ```

3. Restart the NetQ agent:

   ```
   cumulus@server01:~$ netq config restart agent
   ```

Next, you must enable the NetQ Agent on all the worker nodes, as described in the Install NetQ topic, for complete insight into your container network.

## View Status of Kubernetes Clusters

You can get the status of all Kubernetes clusters in the fabric using the `netq show kubernetes cluster` command:

```
cumulus@hostd-11:~$ netq show kubernetes cluster
Matching kube_cluster records:
Master                   Cluster Name     Controller Status
Scheduler Status Nodes
------------------------ ---------------- --------------------
---------------- --------------------
hostd-11:3.0.0.68        default          Healthy
Healthy          hostd-11 hostd-13 ho

std-22 hosts-11 host

s-12 hosts-23 hosts-

24
hostd-12:3.0.0.69        default          Healthy
Healthy          hostd-12 hostd-21 ho

std-23 hosts-13 host

s-14 hosts-21 hosts-

22
```

To filter the list, you can specify the hostname of the master before the `show` command:

```
cumulus@hostd-11:~$ netq hostd-11 show kubernetes cluster
Matching kube_cluster records:
Master                   Cluster Name     Controller Status
Scheduler Status Nodes
------------------------ ---------------- --------------------
---------------- --------------------
hostd-11:3.0.0.68        default          Healthy
Healthy          hostd-11 hostd-13 ho

std-22 hosts-11 host

s-12 hosts-23 hosts-

24
```

Optionally, you can output the results in JSON format:

```
cumulus@hostd-11:~$ netq show kubernetes cluster json
{
    "kube_cluster":[
        {
            "clusterName":"default",
            "schedulerStatus":"Healthy",
            "master":"hostd-12:3.0.0.69",
            "nodes":"hostd-12 hostd-21 hostd-23 hosts-13 hosts-14
hosts-21 hosts-22",
            "controllerStatus":"Healthy"
        },
        {
            "clusterName":"default",
            "schedulerStatus":"Healthy",
            "master":"hostd-11:3.0.0.68",
            "nodes":"hostd-11 hostd-13 hostd-22 hosts-11 hosts-12
hosts-23 hosts-24",
            "controllerStatus":"Healthy"
    }
    ],
    "truncatedResult":false
}
```

## View Changes to a Cluster

If data collection from the NetQ Agents is not occurring as it once was, you can verify that no changes have been made to the Kubernetes cluster configuration using the *changes* keyword. This example shows the changes that have been made in the last hour. If you want to view a larger timeframe, specify that with the *between* option.

```
cumulus@hostd-11:~$ netq show kubernetes cluster changes
Matching kube_cluster records:
Master                   Cluster Name     Controller Status
Scheduler Status Nodes                                    DBState
Last changed
------------------------ ---------------- --------------------
---------------- ---------------------------------------- --------
----------------
hostd-11:3.0.0.68        default          Healthy
Healthy          hostd-11 hostd-13 hostd-22 hosts-11 host Add      2d:
13h:54m:26s

s-12 hosts-23 hosts-24
hostd-12:3.0.0.69        default          Healthy
Healthy          hostd-12 hostd-21 hostd-23 hosts-13 host Add      2d:
13h:54m:35s

s-14 hosts-21 hosts-22
```

```
hostd-12:3.0.0.69        default          Healthy
Healthy          hostd-12 hostd-21 hostd-23 hosts-13      Add      2d:
13h:54m:50s
hostd-11:3.0.0.68        default          Healthy
Healthy          hostd-11                                 Add      2d:
13h:54m:57s
hostd-12:3.0.0.69        default          Healthy
Healthy          hostd-12                                 Add      2d:
13h:55m:50s
```

### View Kubernetes Pod Information

You can show configuration and status of the pods in a cluster, including the names, labels, addresses, associated cluster and containers, and whether the pod is running. This example shows pods for FRR, Nginx, Calico, various Kubernetes components sorted by master node.

```
cumulus@hostd-11:~$ netq show kubernetes pod
Matching kube_pod records:
Master                   Namespace    Name
IP               Node         Labels               Status
Containers               Last Changed
------------------------ ------------ --------------------
---------------- ------------ -------------------- --------
------------------------ ----------------
hostd-11:3.0.0.68        default      cumulus-frr-8vssx
3.0.0.70         hostd-13     pod-template-generat Running  cumulus-
frr:f8cac70bb217 2d:13h:54m:1s

ion:1 name:cumulus-f

rr controller-revisi

on-hash:3710533951
hostd-11:3.0.0.68        default      cumulus-frr-dkkgp
3.0.5.135        hosts-24     pod-template-generat Running  cumulus-
frr:577a60d5f40c 2d:13h:54m:1s

ion:1 name:cumulus-f

rr controller-revisi

on-hash:3710533951
hostd-11:3.0.0.68        default      cumulus-frr-f4bgx
3.0.3.196        hosts-11     pod-template-generat Running  cumulus-
frr:1bc73154a9f5 2d:13h:54m:1s

ion:1 name:cumulus-f

rr controller-revisi
```

```
on-hash:3710533951
hostd-11:3.0.0.68        default      cumulus-frr-gqqxn
3.0.2.5          hostd-22     pod-template-generat Running  cumulus-
frr:3ee0396d126a 2d:13h:54m:1s

ion:1 name:cumulus-f

rr controller-revisi

on-hash:3710533951
hostd-11:3.0.0.68        default      cumulus-frr-kdh9f
3.0.3.197        hosts-12     pod-template-generat Running  cumulus-
frr:94b6329ecb50 2d:13h:54m:1s

ion:1 name:cumulus-f

rr controller-revisi

on-hash:3710533951
hostd-11:3.0.0.68        default      cumulus-frr-mvv8m
3.0.5.134        hosts-23     pod-template-generat Running  cumulus-
frr:b5845299ce3c 2d:13h:54m:1s

ion:1 name:cumulus-f

rr controller-revisi

on-hash:3710533951
hostd-11:3.0.0.68        default      httpd-5456469bfd-bq9
10.244.49.65     hostd-22     app:httpd            Running  httpd:
79b7f532be2d       2d:13h:48m:18s
                                      zm
hostd-11:3.0.0.68        default      influxdb-6cdb566dd-8
10.244.162.128   hostd-13     app:influx           Running  influxdb:
15dce703cdec    2d:13h:48m:18s
                                      9lwn
hostd-11:3.0.0.68        default      nginx-8586cf59-26pj5
10.244.9.193     hosts-24     run:nginx            Running  nginx:
6e2b65070c86       2d:13h:53m:29s
hostd-11:3.0.0.68        default      nginx-8586cf59-c82ns
10.244.40.128    hosts-12     run:nginx            Running  nginx:
01b017c26725       2d:13h:53m:29s
hostd-11:3.0.0.68        default      nginx-8586cf59-wjwgp
10.244.49.64     hostd-22     run:nginx            Running  nginx:
ed2b4254e328       2d:13h:53m:29s
hostd-11:3.0.0.68        kube-system  calico-etcd-pfg9r
3.0.0.68         hostd-11     k8s-app:calico-etcd  Running  calico-
etcd:f95f44b745a7 2d:13h:55m:59s

pod-template-generat
```

```
ion:1 controller-rev

ision-hash:142071906

5
hostd-11:3.0.0.68        kube-system  calico-kube-controll
3.0.2.5          hostd-22     k8s-app:calico-kube- Running  calico-
kube-controllers: 2d:13h:54m:56s
                                      ers-d669cc78f-
4r5t2                                controllers
3688b0c5e9c5
hostd-11:3.0.0.68        kube-system  calico-node-4px69
3.0.2.5          hostd-22     k8s-app:calico-node  Running  calico-
node:1d01648ebba4 2d:13h:55m:41s

pod-template-generat          install-cni:da350802a3d2

ion:1 controller-rev

ision-hash:324404111

9
hostd-11:3.0.0.68        kube-system  calico-node-bt8w6
3.0.3.196        hosts-11     k8s-app:calico-node  Running  calico-
node:9b3358a07e5e 2d:13h:55m:38s

pod-template-generat          install-cni:d38713e6fdd8

ion:1 controller-rev

ision-hash:324404111

9
hostd-11:3.0.0.68        kube-system  calico-node-gtmkv
3.0.3.197        hosts-12     k8s-app:calico-node  Running  calico-
node:48fcc6c40a6b 2d:13h:55m:34s

pod-template-generat          install-cni:f0838a313eff

ion:1 controller-rev

ision-hash:324404111

9
hostd-11:3.0.0.68        kube-system  calico-node-mvslq
3.0.5.134        hosts-23     k8s-app:calico-node  Running  calico-
node:7b361aece76c 2d:13h:55m:33s

pod-template-generat          install-cni:f2da6bc36bf8

ion:1 controller-rev
```

```
ision-hash:324404111

9
hostd-11:3.0.0.68        kube-system  calico-node-sjj2s
3.0.5.135        hosts-24     k8s-app:calico-node  Running  calico-
node:6e13b2b73031 2d:13h:55m:29s

pod-template-generat          install-cni:fa4b2b17fba9

ion:1 controller-rev

ision-hash:324404111

9
hostd-11:3.0.0.68        kube-system  calico-node-vdkk5
3.0.0.70         hostd-13     k8s-app:calico-node  Running  calico-
node:fb3ec9429281 2d:13h:55m:36s

pod-template-generat          install-cni:b56980da7294

ion:1 controller-rev

ision-hash:324404111

9
hostd-11:3.0.0.68        kube-system  calico-node-zzfkr
3.0.0.68         hostd-11     k8s-app:calico-node  Running  calico-
node:c1ac399dd862 2d:13h:55m:59s

pod-template-generat          install-cni:60a779fdc47a

ion:1 controller-rev

ision-hash:324404111

9
hostd-11:3.0.0.68        kube-system  etcd-hostd-11
3.0.0.68         hostd-11     tier:control-plane c Running  etcd:
dde63d44a2f5        2d:13h:56m:44s

omponent:etcd
hostd-11:3.0.0.68        kube-system  kube-apiserver-hostd
3.0.0.68         hostd-11     tier:control-plane c Running  kube-
apiserver:0cd557bbf 2d:13h:56m:44s

-11                                                omponent:kube-
apiser          2fe

ver
```

```
hostd-11:3.0.0.68        kube-system  kube-controller-mana
3.0.0.68         hostd-11     tier:control-plane c Running   kube-
controller-manager: 2d:13h:56m:44s
                                      ger-hostd-
11                                       omponent:kube-
contro          89b2323d09b2

lller-manager
hostd-11:3.0.0.68        kube-system  kube-dns-6f4fd4bdf-p
10.244.34.64     hosts-23     k8s-app:kube-dns     Running   dnsmasq:
284d9d363999 kub 2d:13h:54m:56s

lv7p
edns:bd8bdc49b950 sideca

r:fe10820ffb19
hostd-11:3.0.0.68        kube-system  kube-proxy-4cx2t
3.0.3.197        hosts-12     k8s-app:kube-proxy p Running   kube-
proxy:49b0936a4212  2d:13h:55m:34s

od-template-generati

on:1 controller-revi

sion-hash:3953509896
hostd-11:3.0.0.68        kube-system  kube-proxy-7674k
3.0.3.196        hosts-11     k8s-app:kube-proxy p Running   kube-
proxy:5dc2f5fe0fad  2d:13h:55m:38s

od-template-generati

on:1 controller-revi

sion-hash:3953509896
hostd-11:3.0.0.68        kube-system  kube-proxy-ck5cn
3.0.2.5          hostd-22     k8s-app:kube-proxy p Running   kube-
proxy:6944f7ff8c18  2d:13h:55m:41s

od-template-generati

on:1 controller-revi

sion-hash:3953509896
hostd-11:3.0.0.68        kube-system  kube-proxy-f9dt8
3.0.0.68         hostd-11     k8s-app:kube-proxy p Running   kube-
proxy:032cc82ef3f8  2d:13h:55m:59s

od-template-generati

on:1 controller-revi

sion-hash:3953509896
```

```
hostd-11:3.0.0.68        kube-system  kube-proxy-j6qw6
3.0.5.135        hosts-24     k8s-app:kube-proxy p Running  kube-
proxy:10544e43212e  2d:13h:55m:29s

od-template-generati

on:1 controller-revi

sion-hash:3953509896
hostd-11:3.0.0.68        kube-system  kube-proxy-lq8zz
3.0.5.134        hosts-23     k8s-app:kube-proxy p Running  kube-
proxy:1bcfa09bb186  2d:13h:55m:33s

od-template-generati

on:1 controller-revi

sion-hash:3953509896
hostd-11:3.0.0.68        kube-system  kube-proxy-vg7kj
3.0.0.70         hostd-13     k8s-app:kube-proxy p Running  kube-
proxy:8fed384b68e5  2d:13h:55m:36s

od-template-generati

on:1 controller-revi

sion-hash:3953509896
hostd-11:3.0.0.68        kube-system  kube-scheduler-hostd
3.0.0.68         hostd-11     tier:control-plane c Running  kube-
scheduler:c262a8071 2d:13h:56m:44s

-11                                                omponent:kube-
schedu          3cb

ler
hostd-12:3.0.0.69        default      cumulus-frr-2gkdv
3.0.2.4          hostd-21     pod-template-generat Running  cumulus-
frr:25d1109f8898 2d:13h:54m:39s

ion:1 name:cumulus-f

rr controller-revisi

on-hash:3710533951
hostd-12:3.0.0.69        default      cumulus-frr-b9dm5
3.0.3.199        hosts-14     pod-template-generat Running  cumulus-
frr:45063f9a095f 2d:13h:54m:39s

ion:1 name:cumulus-f

rr controller-revisi
```

```
on-hash:3710533951
hostd-12:3.0.0.69        default      cumulus-frr-rtqhv
3.0.2.6          hostd-23     pod-template-generat Running   cumulus-
frr:63e802a52ea2 2d:13h:54m:39s

ion:1 name:cumulus-f

rr controller-revisi

on-hash:3710533951
hostd-12:3.0.0.69        default      cumulus-frr-tddrg
3.0.5.133        hosts-22     pod-template-generat Running   cumulus-
frr:52dd54e4ac9f 2d:13h:54m:39s

ion:1 name:cumulus-f

rr controller-revisi

on-hash:3710533951
hostd-12:3.0.0.69        default      cumulus-frr-vx7jp
3.0.5.132        hosts-21     pod-template-generat Running   cumulus-
frr:1c20addfcbd3 2d:13h:54m:39s

ion:1 name:cumulus-f

rr controller-revisi

on-hash:3710533951
hostd-12:3.0.0.69        default      cumulus-frr-x7ft5
3.0.3.198        hosts-13     pod-template-generat Running   cumulus-
frr:b0f63792732e 2d:13h:54m:39s

ion:1 name:cumulus-f

rr controller-revisi

on-hash:3710533951
hostd-12:3.0.0.69        kube-system  calico-etcd-btqgt
3.0.0.69         hostd-12     k8s-app:calico-etcd  Running   calico-
etcd:72b1a16968fb 2d:13h:56m:52s

pod-template-generat

ion:1 controller-rev

ision-hash:142071906

5
hostd-12:3.0.0.69        kube-system  calico-kube-controll
3.0.5.132        hosts-21     k8s-app:calico-kube- Running   calico-
kube-controllers: 2d:13h:54m:49s
```

```
                                      ers-d669cc78f-
bdnzk                                controllers
6821bf04696f
hostd-12:3.0.0.69        kube-system  calico-node-4g6vd
3.0.3.198        hosts-13     k8s-app:calico-node  Running  calico-
node:1046b559a50c 2d:13h:55m:53s

pod-template-generat          install-cni:0a136851da17

ion:1 controller-rev

ision-hash:490828062
hostd-12:3.0.0.69        kube-system  calico-node-4hg6l
3.0.0.69         hostd-12     k8s-app:calico-node  Running  calico-
node:4e7acc83f8e8 2d:13h:56m:52s

pod-template-generat          install-cni:a26e76de289e

ion:1 controller-rev

ision-hash:490828062
hostd-12:3.0.0.69        kube-system  calico-node-4p66v
3.0.2.6          hostd-23     k8s-app:calico-node  Running  calico-
node:a7a44072e4e2 2d:13h:56m:0s

pod-template-generat          install-cni:9a19da2b2308

ion:1 controller-rev

ision-hash:490828062
hostd-12:3.0.0.69        kube-system  calico-node-5z7k4
3.0.5.133        hosts-22     k8s-app:calico-node  Running  calico-
node:9878b0606158 2d:13h:55m:45s

pod-template-generat          install-cni:489f8f326cf9

ion:1 controller-rev

ision-hash:490828062
hostd-12:3.0.0.69        kube-system  calico-node-885s6
3.0.5.132        hosts-21     k8s-app:calico-node  Running  calico-
node:24a696f0406c 2d:13h:55m:48s

pod-template-generat          install-cni:15f626e44a6d

ion:1 controller-rev

ision-hash:490828062
hostd-12:3.0.0.69        kube-system  calico-node-c8wjf
3.0.3.199        hosts-14     k8s-app:calico-node  Running  calico-
node:597c8b2053f4 2d:13h:55m:50s
```

```
pod-template-generat          install-cni:646e8df27be8

ion:1 controller-rev

ision-hash:490828062
hostd-12:3.0.0.69        kube-system  calico-node-gkkgk
3.0.2.4          hostd-21     k8s-app:calico-node  Running  calico-
node:73806361f929 2d:13h:56m:5s

pod-template-generat          install-cni:2f9fedf26968

ion:1 controller-rev

ision-hash:490828062
hostd-12:3.0.0.69        kube-system  etcd-hostd-12
3.0.0.69         hostd-12     tier:control-plane c Running  etcd:
cba8d4559e7f        2d:13h:57m:52s

omponent:etcd
hostd-12:3.0.0.69        kube-system  kube-apiserver-hostd
3.0.0.69         hostd-12     tier:control-plane c Running  kube-
apiserver:bbb852aed 2d:13h:57m:52s

-12                                                omponent:kube-
apiser          a1e

ver
hostd-12:3.0.0.69        kube-system  kube-controller-mana
3.0.0.69         hostd-12     tier:control-plane c Running  kube-
controller-manager: 2d:13h:57m:52s
                                      ger-hostd-
12                                       omponent:kube-
contro          f3d5501adbf3

ller-manager
hostd-12:3.0.0.69        kube-system  kube-dns-6f4fd4bdf-5
10.245.104.128   hosts-22     k8s-app:kube-dns     Running  dnsmasq:
b9149784c5d0 kub 2d:13h:54m:49s

psn4
edns:370104ad260c sideca

r:2dc9ac7eb34b
hostd-12:3.0.0.69        kube-system  kube-proxy-56dq8
3.0.5.132        hosts-21     k8s-app:kube-proxy p Running  kube-
proxy:c3f9944efcac  2d:13h:55m:48s

od-template-generati

on:1 controller-revi
```

```
sion-hash:3953509896
hostd-12:3.0.0.69        kube-system  kube-proxy-5c9rx
3.0.2.4          hostd-21     k8s-app:kube-proxy p Running  kube-
proxy:7266de023ad9  2d:13h:56m:5s

od-template-generati

on:1 controller-revi

sion-hash:3953509896
hostd-12:3.0.0.69        kube-system  kube-proxy-5pznh
3.0.3.198        hosts-13     k8s-app:kube-proxy p Running  kube-
proxy:846a571b6fd2  2d:13h:55m:53s

od-template-generati

on:1 controller-revi

sion-hash:3953509896
hostd-12:3.0.0.69        kube-system  kube-proxy-8mt6w
3.0.2.6          hostd-23     k8s-app:kube-proxy p Running  kube-
proxy:9de8b5c76565  2d:13h:56m:0s

od-template-generati

on:1 controller-revi

sion-hash:3953509896
hostd-12:3.0.0.69        kube-system  kube-proxy-9qngl
3.0.3.199        hosts-14     k8s-app:kube-proxy p Running  kube-
proxy:638ffdb9ed51  2d:13h:55m:50s

od-template-generati

on:1 controller-revi

sion-hash:3953509896
hostd-12:3.0.0.69        kube-system  kube-proxy-k568l
3.0.0.69         hostd-12     k8s-app:kube-proxy p Running  kube-
proxy:a0e081e5a141  2d:13h:56m:52s

od-template-generati

on:1 controller-revi

sion-hash:3953509896
hostd-12:3.0.0.69        kube-system  kube-proxy-mwf6s
3.0.5.133        hosts-22     k8s-app:kube-proxy p Running  kube-
proxy:55d80158e5fc  2d:13h:55m:45s

od-template-generati
```

```
on:1 controller-revi

sion-hash:3953509896
hostd-12:3.0.0.69        kube-system  kube-scheduler-hostd
3.0.0.69         hostd-12     tier:control-plane c Running   kube-
scheduler:d941808cd 2d:13h:57m:52s

-12                                                omponent:kube-
schedu          f2a

ler
```

You can filter this information to focus on a particular pod:

```
cumulus@hostd-11:~$ netq show kubernetes pod node hostd-11
Matching kube_pod records:
Master                   Namespace    Name
IP               Node         Labels               Status
Containers               Last Changed
------------------------ ------------ --------------------
---------------- ------------ -------------------- --------
------------------------ ----------------
hostd-11:3.0.0.68        kube-system  calico-etcd-pfg9r
3.0.0.68         hostd-11     k8s-app:calico-etcd  Running   calico-
etcd:f95f44b745a7 2d:14h:0m:59s

pod-template-generat

ion:1 controller-rev

ision-hash:142071906

5
hostd-11:3.0.0.68        kube-system  calico-node-zzfkr
3.0.0.68         hostd-11     k8s-app:calico-node  Running   calico-
node:c1ac399dd862 2d:14h:0m:59s

pod-template-generat          install-cni:60a779fdc47a

ion:1 controller-rev

ision-hash:324404111

9
hostd-11:3.0.0.68        kube-system  etcd-hostd-11
3.0.0.68         hostd-11     tier:control-plane c Running   etcd:
dde63d44a2f5        2d:14h:1m:44s

omponent:etcd
```

```
hostd-11:3.0.0.68        kube-system  kube-apiserver-hostd
3.0.0.68         hostd-11     tier:control-plane c Running  kube-
apiserver:0cd557bbf 2d:14h:1m:44s

-11                                                omponent:kube-
apiser          2fe

ver
hostd-11:3.0.0.68        kube-system  kube-controller-mana
3.0.0.68         hostd-11     tier:control-plane c Running  kube-
controller-manager: 2d:14h:1m:44s
                                      ger-hostd-
11                                       omponent:kube-
contro          89b2323d09b2

ller-manager
hostd-11:3.0.0.68        kube-system  kube-proxy-f9dt8
3.0.0.68         hostd-11     k8s-app:kube-proxy p Running  kube-
proxy:032cc82ef3f8  2d:14h:0m:59s

od-template-generati

on:1 controller-revi

sion-hash:3953509896
hostd-11:3.0.0.68        kube-system  kube-scheduler-hostd
3.0.0.68         hostd-11     tier:control-plane c Running  kube-
scheduler:c262a8071 2d:14h:1m:44s

-11                                                omponent:kube-
schedu          3cb

ler
```

## View Kubernetes Node Information

You can view a lot of information about a node, including the pod CIDR and kubelet status.

```
cumulus@host:~$ netq hostd-11 show kubernetes node
Matching kube_cluster records:
Master                   Cluster Name     Node Name
Role       Status           Labels               Pod
CIDR                 Last Changed
------------------------ ---------------- --------------------
---------- ---------------- --------------------
------------------------ ----------------
hostd-11:3.0.0.68        default          hostd-11
master     KubeletReady     node-role.kubernetes 10.224.0.0
/24              14h:23m:46s
```

```
.io/master: kubernet

es.io/hostname:hostd

-11 beta.kubernetes.

io/arch:amd64 beta.k

ubernetes.io/os:linu

x
hostd-11:3.0.0.68        default         hostd-13
worker     KubeletReady     kubernetes.io/hostna 10.224.3.0
/24            14h:19m:56s

me:hostd-13 beta.kub

ernetes.io/arch:amd6

4 beta.kubernetes.io

/os:linux
hostd-11:3.0.0.68        default         hostd-22
worker     KubeletReady     kubernetes.io/hostna 10.224.1.0
/24            14h:24m:31s

me:hostd-22 beta.kub

ernetes.io/arch:amd6

4 beta.kubernetes.io

/os:linux
hostd-11:3.0.0.68        default         hosts-11
worker     KubeletReady     kubernetes.io/hostna 10.224.2.0
/24            14h:24m:16s

me:hosts-11 beta.kub

ernetes.io/arch:amd6

4 beta.kubernetes.io

/os:linux
hostd-11:3.0.0.68        default         hosts-12
worker     KubeletReady     kubernetes.io/hostna 10.224.4.0
/24            14h:24m:16s

me:hosts-12 beta.kub

ernetes.io/arch:amd6
```

```
4 beta.kubernetes.io

/os:linux
hostd-11:3.0.0.68        default          hosts-23
worker     KubeletReady     kubernetes.io/hostna 10.224.5.0
/24            14h:24m:16s

me:hosts-23 beta.kub

ernetes.io/arch:amd6

4 beta.kubernetes.io

/os:linux
hostd-11:3.0.0.68        default          hosts-24
worker     KubeletReady     kubernetes.io/hostna 10.224.6.0
/24            14h:24m:1s

me:hosts-24 beta.kub

ernetes.io/arch:amd6

4 beta.kubernetes.io

/os:linux
```

To display the kubelet or Docker version, append **`components`** to the above command. This example lists all the details of all master and worker nodes because the master's hostname — *hostd-11* in this case — was included in the query.

```
cumulus@hostd-11:~$ netq hostd-11 show kubernetes node components
Matching kube_cluster records:
                         Master           Cluster Name         Node
Name    Kubelet      KubeProxy         Container Runt

ime
------------------------ ---------------- --------------------
------------ ------------ ----------------- --------------
hostd-11:3.0.0.68        default          hostd-11             v1.
9.2       v1.9.2       docker://17.3.2   KubeletReady
hostd-11:3.0.0.68        default          hostd-13             v1.
9.2       v1.9.2       docker://17.3.2   KubeletReady
hostd-11:3.0.0.68        default          hostd-22             v1.
9.2       v1.9.2       docker://17.3.2   KubeletReady
hostd-11:3.0.0.68        default          hosts-11             v1.
9.2       v1.9.2       docker://17.3.2   KubeletReady
hostd-11:3.0.0.68        default          hosts-12             v1.
9.2       v1.9.2       docker://17.3.2   KubeletReady
```

```
hostd-11:3.0.0.68        default          hosts-23             v1.
9.2       v1.9.2       docker://17.3.2   KubeletReady
hostd-11:3.0.0.68        default          hosts-24             v1.
9.2       v1.9.2       docker://17.3.2   KubeletReady
```

To view only the details for a worker node, specify the hostname at the end of the command after the **name** command:

```
cumulus@hostd-11:~$ netq hostd-11 show kubernetes node components
name hostd-13
Matching kube_cluster records:
                         Master           Cluster Name         Node
Name    Kubelet      KubeProxy        Container Runt

ime
------------------------ ---------------- --------------------
------------ ------------ ----------------- --------------
hostd-11:3.0.0.68        default          hostd-13             v1.
9.2       v1.9.2       docker://17.3.2   KubeletReady
```

You can view information about the replica set:

```
cumulus@hostd-11:~$ netq hostd-11 show kubernetes replica-set
Matching kube_replica records:
Master                   Cluster Name Namespace        Replication
Name               Labels
Replicas                           Ready Replicas Last Changed
------------------------ ------------ ----------------
------------------------------ --------------------
---------------------------------- -------------- ----------------
hostd-11:3.0.0.68        default      default          influxdb-
6cdb566dd            app:influx
1                                  1              14h:19m:28s
hostd-11:3.0.0.68        default      default          nginx-
8586cf59                 run:nginx
3                                  3              14h:24m:39s
hostd-11:3.0.0.68        default      default          httpd-
5456469bfd               app:httpd
1                                  1              14h:19m:28s
hostd-11:3.0.0.68        default      kube-system      kube-dns-
6f4fd4bdf            k8s-app:kube-dns
1                                  1              14h:27m:9s
hostd-11:3.0.0.68        default      kube-system      calico-kube-
controllers-d669cc k8s-app:calico-kube-
1                                  1              14h:27m:9s

78f                            controllers
```

You can view information about the daemon set:

```
cumulus@hostd-11:~$ netq hostd-11 show kubernetes daemon-set
namespace default
Matching kube_daemonset records:
Master                   Cluster Name Namespace        Daemon Set
Name                Labels               Desired Count Ready Count
Last Changed
------------------------ ------------ ----------------
------------------------------ -------------------- -------------
----------- ----------------
hostd-11:3.0.0.68        default      default          cumulus-
frr                    k8s-app:cumulus-frr  6              6
14h:25m:37s
```

You can view information about the pod:

```
cumulus@hostd-11:~$ netq hostd-11 show kubernetes pod namespace
default label nginx
Matching kube_pod records:
Master                   Namespace    Name
IP               Node         Labels               Status
Containers               Last Changed
------------------------ ------------ --------------------
---------------- ------------ -------------------- --------
------------------------ ----------------
hostd-11:3.0.0.68        default      nginx-8586cf59-26pj5
10.244.9.193     hosts-24     run:nginx            Running  nginx:
6e2b65070c86       14h:25m:24s
hostd-11:3.0.0.68        default      nginx-8586cf59-c82ns
10.244.40.128    hosts-12     run:nginx            Running  nginx:
01b017c26725       14h:25m:24s
hostd-11:3.0.0.68        default      nginx-8586cf59-wjwgp
10.244.49.64     hostd-22     run:nginx            Running  nginx:
ed2b4254e328       14h:25m:24s

cumulus@hostd-11:~$ netq hostd-11 show kubernetes pod namespace
default label app
Matching kube_pod records:
Master                   Namespace    Name
IP               Node         Labels               Status
Containers               Last Changed
------------------------ ------------ --------------------
---------------- ------------ -------------------- --------
------------------------ ----------------
hostd-11:3.0.0.68        default      httpd-5456469bfd-bq9
10.244.49.65     hostd-22     app:httpd            Running  httpd:
79b7f532be2d       14h:20m:34s
                                      zm
```

```
hostd-11:3.0.0.68        default      influxdb-6cdb566dd-8
10.244.162.128   hostd-13     app:influx           Running  influxdb:
15dce703cdec    14h:20m:34s
                                      9lwn
```

You can view information about the replication controller:

```
cumulus@hostd-11:~$ netq hostd-11 show kubernetes replication-
controller
No matching kube_replica records found
```

You can view information about a deployment:

```
cumulus@hostd-11:~$ netq hostd-11 show kubernetes deployment name
nginx
Matching kube_deployment records:
Master                   Namespace       Name
Replicas                           Ready Replicas
Labels                         Last Changed
------------------------ --------------- --------------------
---------------------------------- --------------
------------------------------ ----------------
hostd-11:3.0.0.68        default         nginx
3                                  3              run:
nginx                      14h:27m:20s
```

You can search for information using labels as well. The label search is similar to a "contains" regular expression search. In the following example, we are looking for all nodes that contain *kube* in the replication set name or label:

```
cumulus@hostd-11:~$ netq hostd-11 show kubernetes replica-set label
kube
Matching kube_replica records:
Master                   Cluster Name Namespace        Replication
Name                   Labels
Replicas                           Ready Replicas Last Changed
------------------------ ------------ ----------------
------------------------------ --------------------
---------------------------------- -------------- ----------------
hostd-11:3.0.0.68        default      kube-system      kube-dns-
6f4fd4bdf             k8s-app:kube-dns
1                                  1              14h:30m:41s
hostd-11:3.0.0.68        default      kube-system      calico-kube-
controllers-d669cc k8s-app:calico-kube-
1                                  1              14h:30m:41s

78f                            controllers
```

## View Container Connectivity

You can view the connectivity graph of a Kubernetes pod, seeing its replica set, deployment or service level. The impact/connectivity graph starts with the server where the pod is deployed, and shows the peer for each server interface.

```
cumulus@hostd-11:~$ netq hostd-11 show kubernetes deployment name
nginx connectivity
nginx -- nginx-8586cf59-wjwgp -- hostd-22:swp1:torbond1 -- swp7:
hostbond3:torc-21
                              -- hostd-22:swp2:torbond1 -- swp7:
hostbond3:torc-22
                              -- hostd-22:swp3:NetQBond-2 -- swp20:
NetQBond-20:noc-pr
                              -- hostd-22:swp4:NetQBond-2 -- swp20:
NetQBond-20:noc-se
      -- nginx-8586cf59-c82ns -- hosts-12:swp2:NetQBond-1 -- swp23:
NetQBond-23:noc-pr
                              -- hosts-12:swp3:NetQBond-1 -- swp23:
NetQBond-23:noc-se
                              -- hosts-12:swp1:swp1 -- swp6:VlanA-1:
tor-1
      -- nginx-8586cf59-26pj5 -- hosts-24:swp2:NetQBond-1 -- swp29:
NetQBond-29:noc-pr
                              -- hosts-24:swp3:NetQBond-1 -- swp29:
NetQBond-29:noc-se
                              -- hosts-24:swp1:swp1 -- swp8:VlanA-1:
tor-2
```

## View Kubernetes Service Connectivity and Impact

You can show the Kubernetes services in a cluster:

```
cumulus@hostd-11:~$ netq show kubernetes service
Matching kube_service records:
Master                   Namespace        Service Name
Labels       Type       Cluster IP       External IP
Ports                               Last Changed
------------------------ ---------------- --------------------
------------ ---------- ---------------- ----------------
----------------------------------- ----------------
hostd-11:3.0.0.68        default
kubernetes                        ClusterIP
10.96.0.1                         TCP:443
2d:13h:45m:30s
hostd-11:3.0.0.68        kube-system      calico-etcd          k8s-
app:cali ClusterIP  10.96.232.136                     TCP:
6666                            2d:13h:45m:27s
```

```
                                                               co-etcd
hostd-11:3.0.0.68        kube-system      kube-dns             k8s-
app:kube ClusterIP  10.96.0.10                        UDP:53 TCP:
53                       2d:13h:45m:28s
                                                               -dns
hostd-12:3.0.0.69        default
kubernetes                        ClusterIP
10.96.0.1                         TCP:443
2d:13h:46m:24s
hostd-12:3.0.0.69        kube-system      calico-etcd          k8s-
app:cali ClusterIP  10.96.232.136                     TCP:
6666                            2d:13h:46m:20s
                                                               co-etcd
hostd-12:3.0.0.69        kube-system      kube-dns             k8s-
app:kube ClusterIP  10.96.0.10                        UDP:53 TCP:
53                       2d:13h:46m:20s
                                                               -dns
```

And get detailed information about a Kubernetes service:

```
cumulus@hostd-11:~$ netq show kubernetes service name calico-etcd
Matching kube_service records:
Master                   Namespace        Service Name
Labels       Type       Cluster IP       External IP
Ports                              Last Changed
------------------------ ---------------- --------------------
------------ ---------- ---------------- ----------------
----------------------------------- ----------------
hostd-11:3.0.0.68        kube-system      calico-etcd          k8s-
app:cali ClusterIP  10.96.232.136                     TCP:
6666                            2d:13h:48m:10s
                                                               co-etcd
hostd-12:3.0.0.69        kube-system      calico-etcd          k8s-
app:cali ClusterIP  10.96.232.136                     TCP:
6666                            2d:13h:49m:3s
                                                               co-etcd
```

To see the connectivity of a given Kubernetes service, run:

```
cumulus@hostd-11:~$ netq show kubernetes service name calico-etcd
connectivity
calico-etcd -- calico-etcd-pfg9r -- hostd-11:swp1:torbond1 -- swp6:
hostbond2:torc-11
                                 -- hostd-11:swp2:torbond1 -- swp6:
hostbond2:torc-12
                                 -- hostd-11:swp3:NetQBond-2 -- swp16:
NetQBond-16:noc-pr
                                 -- hostd-11:swp4:NetQBond-2 -- swp16:
NetQBond-16:noc-se
```

```
calico-etcd -- calico-etcd-btqgt -- hostd-12:swp1:torbond1 -- swp7:
hostbond3:torc-11
                                 -- hostd-12:swp2:torbond1 -- swp7:
hostbond3:torc-12
                                 -- hostd-12:swp3:NetQBond-2 -- swp17:
NetQBond-17:noc-pr
                                 -- hostd-12:swp4:NetQBond-2 -- swp17:
NetQBond-17:noc-se
```

To see the impact of a given Kubernetes service, run:

```
cumulus@hostd-11:~$ netq hostd-11 show impact kubernetes service name
calico-etcd
calico-etcd -- calico-etcd-pfg9r -- hostd-11:swp1:torbond1 -- swp6:
hostbond2:torc-11
                                 -- hostd-11:swp2:torbond1 -- swp6:
hostbond2:torc-12
                                 -- hostd-11:swp3:NetQBond-2 -- swp16:
NetQBond-16:noc-pr
                                 -- hostd-11:swp4:NetQBond-2 -- swp16:
NetQBond-16:noc-se
```

## View Kubernetes Cluster Configuration in the Past

You can use the "time machine" features (see page 230) of NetQ on a Kubernetes cluster, using the `around` and `changes` commands to go back in time to check the network status and identify any changes that occurred on the network.

This example shows the current state of the network. Notice there is a node named *hosts-23*. hosts-23 is there because the node *hostd-22* went down and Kubernetes spun up a third replica on a different host to satisfy the deployment requirement.

```
cumulus@redis-1:~$ netq hostd-11 show kubernetes deployment name
nginx connectivity
nginx -- nginx-8586cf59-fqtnj -- hosts-12:swp2:NetQBond-1 -- swp23:
NetQBond-23:noc-pr
                              -- hosts-12:swp3:NetQBond-1 -- swp23:
NetQBond-23:noc-se
                              -- hosts-12:swp1:swp1 -- swp6:VlanA-1:
tor-1
      -- nginx-8586cf59-8g487 -- hosts-24:swp2:NetQBond-1 -- swp29:
NetQBond-29:noc-pr
                              -- hosts-24:swp3:NetQBond-1 -- swp29:
NetQBond-29:noc-se
                              -- hosts-24:swp1:swp1 -- swp8:VlanA-1:
tor-2
      -- nginx-8586cf59-2hb8t -- hosts-23:swp1:swp1 -- swp7:VlanA-1:
tor-2
```

```
                              -- hosts-23:swp2:NetQBond-1 -- swp28:
NetQBond-28:noc-pr
                              -- hosts-23:swp3:NetQBond-1 -- swp28:
NetQBond-28:noc-se
```

You can see this by going back in time 10 minutes. *hosts-23* was not present, whereas *hostd-22* **was** present:

```
cumulus@redis-1:~$ netq hostd-11 show kubernetes deployment name
nginx connectivity around 10m
nginx -- nginx-8586cf59-fqtnj -- hosts-12:swp2:NetQBond-1 -- swp23:
NetQBond-23:noc-pr
                              -- hosts-12:swp3:NetQBond-1 -- swp23:
NetQBond-23:noc-se
                              -- hosts-12:swp1:swp1 -- swp6:VlanA-1:
tor-1
      -- nginx-8586cf59-2xxs4 -- hostd-22:swp1:torbond1 -- swp7:
hostbond3:torc-21
                              -- hostd-22:swp2:torbond1 -- swp7:
hostbond3:torc-22
                              -- hostd-22:swp3:NetQBond-2 -- swp20:
NetQBond-20:noc-pr
                              -- hostd-22:swp4:NetQBond-2 -- swp20:
NetQBond-20:noc-se
      -- nginx-8586cf59-8g487 -- hosts-24:swp2:NetQBond-1 -- swp29:
NetQBond-29:noc-pr
                              -- hosts-24:swp3:NetQBond-1 -- swp29:
NetQBond-29:noc-se
                              -- hosts-24:swp1:swp1 -- swp8:VlanA-1:
tor-2
```

You can determine the impact on the Kubernetes deployment in the event a host or switch goes down. The output is color coded (not shown in the example below) so you can clearly see the impact: green shows no impact, yellow shows partial impact, and red shows full impact.

```
cumulus@hostd-11:~$ netq torc-21 show impact kubernetes deployment
name nginx
nginx -- nginx-8586cf59-wjwgp -- hostd-22:swp1:torbond1 -- swp7:
hostbond3:torc-21
                              -- hostd-22:swp2:torbond1 -- swp7:
hostbond3:torc-22
                              -- hostd-22:swp3:NetQBond-2 -- swp20:
NetQBond-20:noc-pr
                              -- hostd-22:swp4:NetQBond-2 -- swp20:
NetQBond-20:noc-se
      -- nginx-8586cf59-c82ns -- hosts-12:swp2:NetQBond-1 -- swp23:
NetQBond-23:noc-pr
```

```
                              -- hosts-12:swp3:NetQBond-1 -- swp23:
NetQBond-23:noc-se
                              -- hosts-12:swp1:swp1 -- swp6:VlanA-1:
tor-1
      -- nginx-8586cf59-26pj5 -- hosts-24:swp2:NetQBond-1 -- swp29:
NetQBond-29:noc-pr
                              -- hosts-24:swp3:NetQBond-1 -- swp29:
NetQBond-29:noc-se
                              -- hosts-24:swp1:swp1 -- swp8:VlanA-1:
tor-2
cumulus@hostd-11:~$ netq hosts-12 show impact kubernetes deployment
name nginx
nginx -- nginx-8586cf59-wjwgp -- hostd-22:swp1:torbond1 -- swp7:
hostbond3:torc-21
                              -- hostd-22:swp2:torbond1 -- swp7:
hostbond3:torc-22
                              -- hostd-22:swp3:NetQBond-2 -- swp20:
NetQBond-20:noc-pr
                              -- hostd-22:swp4:NetQBond-2 -- swp20:
NetQBond-20:noc-se
      -- nginx-8586cf59-c82ns -- hosts-12:swp2:NetQBond-1 -- swp23:
NetQBond-23:noc-pr
                              -- hosts-12:swp3:NetQBond-1 -- swp23:
NetQBond-23:noc-se
                              -- hosts-12:swp1:swp1 -- swp6:VlanA-1:
tor-1
      -- nginx-8586cf59-26pj5 -- hosts-24:swp2:NetQBond-1 -- swp29:
NetQBond-29:noc-pr
                              -- hosts-24:swp3:NetQBond-1 -- swp29:
NetQBond-29:noc-se
```

## Use NetQ with Docker Swarm

The NetQ Agent parses the following Docker events:

- Image: pull and delete
- Container: run, stop, start, restart, attach and detach
- Network: create, connect, disconnect and destroy

Currently, the NetQ Agent does not support:

- Monitoring Docker volume mount and unmount events
- Plugin install and deletes
- Third party network configuration through plugins like Calico

### Requirements

The NetQ Agent supports Docker version 1.13 (Jan 2017), 17.03 or later, including Docker Swarm.

Due to the higher memory requirements to run containers, Cumulus Networks recommends you run the NetQ Telemetry Server on a host with at least 32G RAM. For more information, read the How Far Back in Time Can You Travel? (see page 230) topic.

## Command Summary

NetQ provides a set of commands to monitor Docker configurations, including the ability to monitor network, service, Swarm cluster, network, and nodes:

```
netq <hostname> show docker container adjacent [interfaces <remote-
physical-interface>] [around <text-time>] [json]

netq [<hostname>] show docker summary [<docker-version>] [around
<text-time>] [json]
netq [<hostname>] show docker summary [<docker-version>] changes
[between <text-time> and <text-endtime>] [json]
netq [<hostname>] show docker network [name <network-name> | <ipv4
/prefixlen>] [brief] [around <text-time>] [json]
netq [<hostname>] show docker network [name <network-name> | <ipv4
/prefixlen>] [brief] changes [between <text-time> and <text-endtime>]
[json]
netq [<hostname>] show docker network driver <network-driver> [brief]
[around <text-time>] [json]
netq [<hostname>] show docker network driver <network-driver> [brief]
changes [between <text-time> and <text-endtime>] [json]
netq [<hostname>] show docker service [name <swarm-service-name> |
mode <mode>] [around <text-time>] [json]
netq [<hostname>] show docker service [name <swarm-service-name> |
mode <mode>] connectivity [vrf <vrf>] [around <text-time>] [json]
netq <hostname> show impact docker service [<swarm-service-name>]
[vrf <vrf>] [around <text-time>] [json]

netq [<hostname>] show docker swarm cluster [node-name <cluster-
node>] [around <text-time>] [json]
netq [<hostname>] show docker swarm cluster [<cluster-name>] [node-
name <cluster-node>] [around <text-time>] [json]
netq <hostname> show docker swarm cluster changes [between <text-
time> and <text-endtime>] [json]
netq <hostname> show docker swarm cluster [<cluster-name>] changes
[between <text-time> and <text-endtime>] [json]
netq [<hostname>] show docker swarm network [<swarm-service-name>]
[around <text-time>] [json]
netq <hostname> show docker swarm network [<swarm-service-name>]
changes [between <text-time> and <text-endtime>] [json]
netq [<hostname>] show docker swarm node [<node-name> | role <role>]
[cluster <cluster-name>] [around <text-time>] [json]
netq <hostname> show docker swarm node [<node-name> | role <role>]
[cluster <cluster-name>] changes [between <text-time> and <text-
endtime>] [json]
```

## Enable Docker Container Monitoring

For NetQ to monitor the Docker containers on a host, you must configure the following on the host:

1. Configure the host to point to the telemetry server by its IP address. Refer to the Install NetQ topic for details.
2. Enable Docker monitoring by NetQ. You can specify a polling period between 10 and 120 seconds; 15 seconds is the default.

```
cumulus@hostd-11:~$ netq config add agent docker-monitor poll-
period 20
Successfully added docker monitor. Please restart netq-agent.
```

3. Restart the NetQ agent:

```
cumulus@server01:~$ netq config restart agent
```

## View Container Summary Information

To see a high level view of the network, including the number of containers installed and running on the network, run the `netq show docker summary` command:

```
cumulus@host:~$ netq show docker summary
Hostname    Version     Installed    Running    Images    Swarm
Cluster    Networks
----------  ----------  -----------  ---------  --------
---------------  ----------
exit01      17.06.0-ce          26         26
1                            3
exit02      17.06.0-ce           1          0
3                            3
server01    17.06.0-ce          14         14         4
default                   5
server02    17.06.0-ce           0          0
0                            3
server03    17.06.0-ce           0          0
0                            3
server04    17.06.0-ce           0          0
0                            3
server01    17.06.0-ce          13         13         1
default                   3
server02    17.06.0-ce           0          0
0                            3
```

## Identify Containers on the Network

To view the different container networks and the containers in them, run `netq show docker network`:

```
cumulus@host:~$ netq show docker network
Network Name     Hostname   subnet          gateway         ipv6
ip masq.
---------------- ---------- --------------- --------------- ---------
--------
bridge           exit01     172.17.0.0/16                   Disabled
True
bridge           exit02     172.17.0.0/16                   Disabled
True
bridge           server01   172.17.0.0/16                   Disabled
True
bridge           server02   172.17.0.0/16                   Disabled
True
bridge           server03   172.17.0.0/16                   Disabled
True
bridge           server04   172.17.0.0/16                   Disabled
True
bridge           server01   172.17.0.0/16                   Disabled
True
bridge           server02   172.17.0.0/16                   Disabled
True
bridge           server03   172.17.0.0/16                   Disabled
True
bridge           server04   172.17.0.0/16                   Disabled
True
host             exit01                                     Disabled
False
host             exit02                                     Disabled
False
host             server01                                   Disabled
False
host             server02                                   Disabled
False
host             server03                                   Disabled
False
host             server04                                   Disabled
False
host             server01                                   Disabled
False
host             server02                                   Disabled
False
host             server03                                   Disabled
False
host             server04                                   Disabled
False
```

```
none             exit01                                     Disabled
False
none             exit02                                     Disabled
False
none             server01                                   Disabled
False
none             server02                                   Disabled
False
none             server03                                   Disabled
False
none             server04                                   Disabled
False
none             server01                                   Disabled
False
none             server02                                   Disabled
False
none             server03                                   Disabled
False
none             server04                                   Disabled
False
```

## View Deployed Container Network Drivers

To view all the hosts using a specific container network driver, use `netq show docker network driver NAME`. Use the `brief` keyword for a shorter summary. Docker supports many network drivers.

```
cumulus@host:~$ netq show docker network driver bridge brief
Network Name     Hostname   Driver     subnet          gateway
IP Masq  Containers
---------------- ---------- --------- --------------- ---------------
-------- -------------------------
bridge           exit01     bridge    172.17.0.0/16
True     Name:netcat-8085 IPv4:172

.17.0.7/16,

Name:netcat-8082 IPv4:172

.17.0.4/16,

Name:netcat-8083 IPv4:172

.17.0.5/16,

Name:netcat-8089 IPv4:172

.17.0.11/16,

Name:netcat-8081 IPv4:172
```

```
.17.0.3/16,

Name:netcat-8084 IPv4:172

.17.0.6/16,

Name:netcat-8090 IPv4:172

.17.0.12/16,

Name:netcat-8080 IPv4:172

.17.0.2/16,

Name:netcat-8091 IPv4:172

.17.0.13/16,

Name:netcat-8092 IPv4:172

.17.0.14/16,

Name:netcat-8088 IPv4:172

.17.0.10/16,

Name:netcat-8087 IPv4:172

.17.0.9/16,

Name:netcat-8086 IPv4:172

.17.0.8/16
bridge           exit02      bridge     172.17.0.0/16
True
bridge           server01    bridge     172.17.0.0/16
True
bridge           server02    bridge     172.17.0.0/16
True
bridge           server03    bridge     172.17.0.0/16
True
bridge           server04    bridge     172.17.0.0/16
True
bridge           server01    bridge     172.17.0.0/16
True     Name:netcat-8082 IPv4:172

.17.0.4/16,

Name:netcat-8085 IPv4:172

.17.0.7/16,
Name:netcat-8083 IPv4:172

.17.0.5/16,

Name:netcat-8086 IPv4:172

.17.0.8/16,

Name:netcat-8089 IPv4:172

.17.0.11/16,

Name:netcat-8084 IPv4:172

.17.0.6/16,

Name:netcat-8092 IPv4:172

.17.0.14/16,

Name:netcat-8087 IPv4:172

.17.0.9/16,

Name:netcat-8080 IPv4:172

.17.0.2/16,

Name:netcat-8081 IPv4:172

.17.0.3/16,

Name:netcat-8090 IPv4:172

.17.0.12/16,

Name:netcat-8091 IPv4:172

.17.0.13/16,

Name:netcat-8088 IPv4:172

.17.0.10/16
bridge           server02   bridge    172.17.0.0/16
True
bridge           server03   bridge    172.17.0.0/16
True
bridge           server04   bridge    172.17.0.0/16
True
```

## View All Containers in a Network

To see all the containers on a given container network, run the following command, where the container network is named *host*:

```
cumulus@host:~$ netq show docker container network host
Container Name       Hostname   Container IP      IP Masq  Network
Name   Service Name    UpTime
-------------------- ---------- ----------------- --------
-------------- --------------- ---------------
netcat-9080          exit01     45.0.0.17/26,     False
host                           0:29:42
                                27.0.0.3/32,
                                192.168.0.15/24
netcat-9081          exit01     45.0.0.17/26,     False
host                           0:29:41
                                27.0.0.3/32,
                                192.168.0.15/24
netcat-9082          exit01     45.0.0.17/26,     False
host                           0:29:42
                                27.0.0.3/32,
                                192.168.0.15/24
netcat-9083          exit01     45.0.0.17/26,     False
host                           0:29:39
                                27.0.0.3/32,
                                192.168.0.15/24
netcat-9084          exit01     45.0.0.17/26,     False
host                           0:29:40
                                27.0.0.3/32,
                                192.168.0.15/24
netcat-9085          exit01     45.0.0.17/26,     False
host                           0:29:40
                                27.0.0.3/32,
                                192.168.0.15/24
netcat-9086          exit01     45.0.0.17/26,     False
host                           0:29:39
                                27.0.0.3/32,
                                192.168.0.15/24
netcat-9087          exit01     45.0.0.17/26,     False
host                           0:29:38
                                27.0.0.3/32,
                                192.168.0.15/24
netcat-9088          exit01     45.0.0.17/26,     False
host                           0:29:37
                                27.0.0.3/32,
                                192.168.0.15/24
netcat-9089          exit01     45.0.0.17/26,     False
host                           0:29:38
                                27.0.0.3/32,
                                192.168.0.15/24
```

```
netcat-9090          exit01     45.0.0.17/26,     False
host                           0:29:36
                                27.0.0.3/32,
                                192.168.0.15/24
netcat-9091          exit01     45.0.0.17/26,     False
host                           0:29:37
                                27.0.0.3/32,
                                192.168.0.15/24
netcat-9092          exit01     45.0.0.17/26,     False
host                           0:29:38
                                27.0.0.3/32,
                                192.168.0.15/24
```

The Service Name column is populated when a container is created by Docker Swarm for a service:

```
cumulus@leaf01:mgmt-vrf:~$ netq show docker container
Matching container records are:
Container Name       Hostname   Container IP      IP Masq  Network
Name    Service Name    UpTime
-------------------- ---------- ----------------- --------
-------------- --------------- ---------------
Web.3.xm2jjbe1l60eje hostd-11   10.255.0.9        False
ingress        Web             16:30:47
sgpx8rlq5pf
redis2.nh7ouztl2ap79 hostd-21   172.17.0.2        True
bridge         redis2          16:36:52
2iyycl5bukfh.lznwsxh
8jepg65hr16kccxeau
redis2.rx8uywzrkm9pj hostd-11   172.17.0.2        True
bridge         redis2          16:36:52
e9a81613gfpm.s6fhc09
1xwoqmkjdi3y1kxm7z
Web.1.m72ghox4y2bfeg hosts-21   10.255.0.7        False
ingress        Web             16:30:47
f1ukeocjhgn
Web.2.9t9yuv9za28taz hosts-11   10.255.0.8        False
ingress        Web             16:30:46
3mee6pr8d11
Web.3.kv0icnnh7fxb45 hosts-21   10.255.0.11       False
ingress        Web             14:31:58
```

## View Container Adjacency

NetQ can list all the containers running on hosts adjacent to a top of rack switch. This helps in analyzing what impact the ToR switch can have on an application. Run `netq NODE show docker container adjacent` to identify all the containers that may have been launched on hosts adjacent to a given node:

```
cumulus@leaf01:~$ netq leaf01 show docker container adjacent
```

```
Interface            Peer Node  Peer Interface        Container
Name       IP                   Network    Service Name
-------------------- ---------- ---------------------
-------------------- -------------------- ---------- ---------------
swp6:VlanA-1         server01   mac:00:02:00:00:00:27 netcat-
9090                               host
swp6:VlanA-1         server01   mac:00:02:00:00:00:27 netcat-
9082                               host
swp6:VlanA-1         server01   mac:00:02:00:00:00:27 netcat-
9091                               host
swp6:VlanA-1         server01   mac:00:02:00:00:00:27 netcat-
9086                               host
swp6:VlanA-1         server01   mac:00:02:00:00:00:27 netcat-
9081                               host
swp6:VlanA-1         server01   mac:00:02:00:00:00:27 netcat-
9083                               host
swp6:VlanA-1         server01   mac:00:02:00:00:00:27 netcat-
9087                               host
swp6:VlanA-1         server01   mac:00:02:00:00:00:27 netcat-
9088                               host
swp6:VlanA-1         server01   mac:00:02:00:00:00:27 netcat-
9085                               host
swp6:VlanA-1         server01   mac:00:02:00:00:00:27 netcat-
9080                               host
swp6:VlanA-1         server01   mac:00:02:00:00:00:27 netcat-
9084                               host
swp6:VlanA-1         server01   mac:00:02:00:00:00:27 netcat-
9089                               host
swp6:VlanA-1         server01   mac:00:02:00:00:00:27 netcat-
9092                               host
swp7:VlanA-1         server02   mac:00:02:00:00:00:2a netcat-
8089          172.17.0.11          bridge
swp7:VlanA-1         server02   mac:00:02:00:00:00:2a netcat-
8084          172.17.0.6           bridge
swp7:VlanA-1         server02   mac:00:02:00:00:00:2a netcat-
8092          172.17.0.14          bridge
swp7:VlanA-1         server02   mac:00:02:00:00:00:2a netcat-
8083          172.17.0.5           bridge
swp7:VlanA-1         server02   mac:00:02:00:00:00:2a netcat-
8085          172.17.0.7           bridge
swp7:VlanA-1         server02   mac:00:02:00:00:00:2a netcat-
8081          172.17.0.3           bridge
swp7:VlanA-1         server02   mac:00:02:00:00:00:2a netcat-
8080          172.17.0.2           bridge
swp7:VlanA-1         server02   mac:00:02:00:00:00:2a netcat-
8086          172.17.0.8           bridge
swp7:VlanA-1         server02   mac:00:02:00:00:00:2a netcat-
8088          172.17.0.10          bridge
swp7:VlanA-1         server02   mac:00:02:00:00:00:2a netcat-
8082          172.17.0.4           bridge
swp7:VlanA-1         server02   mac:00:02:00:00:00:2a netcat-
8091          172.17.0.13          bridge
```

```
swp7:VlanA-1         server02   mac:00:02:00:00:00:2a netcat-
8090          172.17.0.12          bridge
swp7:VlanA-1         server02   mac:00:02:00:00:00:2a netcat-
8087          172.17.0.9           bridge
swp8:VlanA-1         server03   mac:00:02:00:00:00:2d netcat-
8091          172.17.0.13          bridge
swp8:VlanA-1         server03   mac:00:02:00:00:00:2d netcat-
8083          172.17.0.5           bridge
swp8:VlanA-1         server03   mac:00:02:00:00:00:2d netcat-
8087          172.17.0.9           bridge
swp8:VlanA-1         server03   mac:00:02:00:00:00:2d netcat-
8082          172.17.0.4           bridge
swp8:VlanA-1         server03   mac:00:02:00:00:00:2d netcat-
8080          172.17.0.2           bridge
swp8:VlanA-1         server03   mac:00:02:00:00:00:2d netcat-
8092          172.17.0.14          bridge
swp8:VlanA-1         server03   mac:00:02:00:00:00:2d netcat-
8086          172.17.0.8           bridge
swp8:VlanA-1         server03   mac:00:02:00:00:00:2d netcat-
8084          172.17.0.6           bridge
swp8:VlanA-1         server03   mac:00:02:00:00:00:2d netcat-
8088          172.17.0.10          bridge
swp8:VlanA-1         server03   mac:00:02:00:00:00:2d netcat-
8090          172.17.0.12          bridge
swp8:VlanA-1         server03   mac:00:02:00:00:00:2d netcat-
8085          172.17.0.7           bridge
swp8:VlanA-1         server03   mac:00:02:00:00:00:2d netcat-
8089          172.17.0.11          bridge
swp8:VlanA-1         server03   mac:00:02:00:00:00:2d netcat-
8081          172.17.0.3           bridge
```

You can filter this output for a given interface:

```
cumulus@switch:~$ netq leaf01 show docker container adjacent
interfaces swp6
Interface            Peer Node  Peer Interface        Container
Name       IP                   Network    Service Name
-------------------- ---------- ---------------------
-------------------- -------------------- ---------- ---------------
swp6:VlanA-1         server01   mac:00:02:00:00:00:27 netcat-
9090                               host
swp6:VlanA-1         server01   mac:00:02:00:00:00:27 netcat-
9082                               host
swp6:VlanA-1         server01   mac:00:02:00:00:00:27 netcat-
9091                               host
swp6:VlanA-1         server01   mac:00:02:00:00:00:27 netcat-
9086                               host
swp6:VlanA-1         server01   mac:00:02:00:00:00:27 netcat-
9081                               host
```

```
swp6:VlanA-1         server01   mac:00:02:00:00:00:27 netcat-
9083                               host
swp6:VlanA-1         server01   mac:00:02:00:00:00:27 netcat-
9087                               host
swp6:VlanA-1         server01   mac:00:02:00:00:00:27 netcat-
9088                               host
swp6:VlanA-1         server01   mac:00:02:00:00:00:27 netcat-
9085                               host
swp6:VlanA-1         server01   mac:00:02:00:00:00:27 netcat-
9080                               host
swp6:VlanA-1         server01   mac:00:02:00:00:00:27 netcat-
9084                               host
swp6:VlanA-1         server01   mac:00:02:00:00:00:27 netcat-
9089
host
swp6:VlanA-1         server01   mac:00:02:00:00:00:27 netcat-
9092                               host
```

## Show Container-Specific Information

You can see information about a given container by running `netq show docker container name NAME`:

```
cumulus@host:~$ netq show docker container name netcat-9092
Name                 Node       IP                IP Masq.
Network        Service Name    Up time
-------------------- ---------- ----------------- --------
-------------- --------------- ---------------
netcat-9092          exit01     45.0.0.17/26,     False
host                           0:34:15
                                27.0.0.3/32,
                                192.168.0.15/24
```

## Show Containers with a Specific Image

To search for all the containers on the network with a specific Docker image, run `netq show docker container image IMAGE_NAME`:

```
cumulus@host:~$ netq show docker container image chilcano/netcat:
jessie
Name                 Node       IP                IP Masq.
Network        Service Name    Up time
-------------------- ---------- ----------------- --------
-------------- --------------- ---------------
netcat-8080          exit01     172.17.0.2        True
bridge                         0:32:09
```

```
netcat-8080          server01   172.17.0.2        True
bridge                         0:23:11
netcat-8081          exit01     172.17.0.3        True
bridge                         0:32:07
netcat-8081          server01   172.17.0.3        True
bridge                         0:23:10
netcat-8082          exit01     172.17.0.4        True
bridge                         0:32:08
netcat-8082          server01   172.17.0.4        True
bridge                         0:23:08
netcat-8083          exit01     172.17.0.5        True
bridge                         0:32:07
netcat-8083          server01   172.17.0.5        True
bridge                         0:23:07
netcat-8084          exit01     172.17.0.6        True
bridge                         0:32:07
netcat-8084          server01   172.17.0.6        True
bridge                         0:23:09
netcat-8085          exit01     172.17.0.7        True
bridge                         0:32:05
netcat-8085          server01   172.17.0.7        True
bridge                         0:23:06
netcat-8086          exit01     172.17.0.8        True
bridge                         0:32:06
netcat-8086          server01   172.17.0.8        True
bridge                         0:23:06
netcat-8087          exit01     172.17.0.9        True
bridge                         0:32:05
netcat-8087          server01   172.17.0.9        True
bridge                         0:23:06
netcat-8088          exit01     172.17.0.10       True
bridge                         0:32:04
netcat-8088          server01   172.17.0.10       True
bridge                         0:23:06
netcat-8089          exit01     172.17.0.11       True
bridge                         0:32:02
netcat-8089          server01   172.17.0.11       True
bridge                         0:23:03
netcat-8090          exit01     172.17.0.12       True
bridge                         0:32:01
netcat-8090          server01   172.17.0.12       True
bridge                         0:23:05
netcat-8091          exit01     172.17.0.13       True
bridge                         0:32:03
netcat-8091          server01   172.17.0.13       True
bridge                         0:23:04
netcat-8092          exit01     172.17.0.14       True
bridge                         0:31:59
netcat-8092          server01   172.17.0.14       True
bridge                         0:23:03
netcat-9080          exit01     45.0.0.17/26,     False
host                           0:31:51
```

```
                                27.0.0.3/32,
                                192.168.0.15/24
netcat-9081          exit01     45.0.0.17/26,     False
host                           0:31:51
                                27.0.0.3/32,
                                192.168.0.15/24
netcat-9082          exit01     45.0.0.17/26,     False
host                           0:31:52
                                27.0.0.3/32,
                                192.168.0.15/24
netcat-9083          exit01     45.0.0.17/26,     False
host                           0:31:49
                                27.0.0.3/32,
                                192.168.0.15/24
netcat-9084          exit01     45.0.0.17/26,     False
host                           0:31:50
                                27.0.0.3/32,
                                192.168.0.15/24
netcat-9085          exit01     45.0.0.17/26,     False
host                           0:31:50
                                27.0.0.3/32,
                                192.168.0.15/24
netcat-9086          exit01     45.0.0.17/26,     False
host                           0:31:48
                                27.0.0.3/32,
                                192.168.0.15/24
netcat-9087          exit01     45.0.0.17/26,     False
host                           0:31:48
                                27.0.0.3/32,
                                192.168.0.15/24
netcat-9088          exit01     45.0.0.17/26,     False
host                           0:31:47
                                27.0.0.3/32,
                                192.168.0.15/24
netcat-9089          exit01     45.0.0.17/26,     False
host                           0:31:48
                                27.0.0.3/32,
                                192.168.0.15/24
netcat-9090          exit01     45.0.0.17/26,     False
host                           0:31:46
                                27.0.0.3/32,
                                192.168.0.15/24
netcat-9091          exit01     45.0.0.17/26,     False
host                           0:31:47
                                27.0.0.3/32,
                                192.168.0.15/24
netcat-9092          exit01     45.0.0.17/26,     False
host                           0:31:47
                                27.0.0.3/32,
                                192.168.0.15/24
```

## Show Container Connectivity

Run `netq HOST show docker container network NAME connectivity` to determine how a particular container is attached to a network. The output tells you the host where the container was launched, adjacent nodes, and adjacent ports.

```
cumulus@leaf01:~$ netq server01 show docker container network host
connectivity
Name            Swarm Service Cont IP        Network    Node
Port                 Peer Node  Peer Port
--------------- ------------- --------------- ---------- ----------
-------------------- ---------- --------------------
netcat-9080                                  host       server01
swp2:NetQBond-1      noc-pr     swp21:NetQBond-19
netcat-9080                                  host       server01
swp3:NetQBond-1      noc-se     swp21:NetQBond-19
netcat-9080                                  host       server01
swp1:swp1            tor-1      Local Node tor-1 and

Ports swp6 <==> Remo

te  Node/s hosts-11

and Ports swp1
netcat-9081                                  host       server01
swp2:NetQBond-1      noc-pr     swp21:NetQBond-19
netcat-9081                                  host       server01
swp3:NetQBond-1      noc-se     swp21:NetQBond-19
netcat-9081                                  host       server01
swp1:swp1            tor-1      Local Node tor-1 and

Ports swp6 <==> Remo

te  Node/s hosts-11

and Ports swp1
netcat-9082                                  host       server01
swp2:NetQBond-1      noc-pr     swp21:NetQBond-19
netcat-9082                                  host       server01
swp3:NetQBond-1      noc-se     swp21:NetQBond-19
netcat-9082                                  host       server01
swp1:swp1            tor-1      Local Node tor-1 and

Ports swp6 <==> Remo

te  Node/s hosts-11

and Ports swp1
netcat-9083                                  host       server01
swp2:NetQBond-1      noc-pr     swp21:NetQBond-19
```

```
netcat-9083                                   host       server01
swp3:NetQBond-1      noc-se     swp21:NetQBond-19
netcat-9083                                   host       server01
swp1:swp1            tor-1      Local Node tor-1 and

Ports swp6 <==> Remo

te  Node/s hosts-11

and Ports swp1
netcat-9084                                   host       server01
swp2:NetQBond-1      noc-pr     swp21:NetQBond-19
netcat-9084                                   host       server01
swp3:NetQBond-1      noc-se     swp21:NetQBond-19
netcat-9084                                   host       server01
swp1:swp1            tor-1      Local Node tor-1 and

Ports swp6 <==> Remo

te  Node/s hosts-11

and Ports swp1
netcat-9085                                   host       server01
swp2:NetQBond-1      noc-pr     swp21:NetQBond-19
netcat-9085                                   host       server01
swp3:NetQBond-1      noc-se     swp21:NetQBond-19
netcat-9085                                   host       server01
swp1:swp1            tor-1      Local Node tor-1 and

Ports swp6 <==> Remo

te  Node/s hosts-11

and Ports swp1
netcat-9086                                   host       server01
swp2:NetQBond-1      noc-pr     swp21:NetQBond-19
netcat-9086                                   host       server01
swp3:NetQBond-1      noc-se     swp21:NetQBond-19
netcat-9086                                   host       server01
swp1:swp1            tor-1      Local Node tor-1 and

Ports swp6 <==> Remo

te  Node/s hosts-11

and Ports swp1
netcat-9087                                   host       server01
swp2:NetQBond-1      noc-pr     swp21:NetQBond-19
netcat-9087                                   host       server01
swp3:NetQBond-1      noc-se     swp21:NetQBond-19
netcat-9087                                   host       server01
swp1:swp1            tor-1      Local Node tor-1 and
```

```
Ports swp6 <==> Remo

te  Node/s hosts-11

and Ports swp1
netcat-9088                                   host       server01
swp2:NetQBond-1      noc-pr     swp21:NetQBond-19
netcat-9088                                   host       server01
swp3:NetQBond-1      noc-se     swp21:NetQBond-19
netcat-9088                                   host       server01
swp1:swp1            tor-1      Local Node tor-1 and

Ports swp6 <==> Remo

te  Node/s hosts-11

and Ports swp1
netcat-9089                                   host       server01
swp2:NetQBond-1      noc-pr     swp21:NetQBond-19
netcat-9089                                   host       server01
swp3:NetQBond-1      noc-se     swp21:NetQBond-19
netcat-9089                                   host       server01
swp1:swp1            tor-1      Local Node tor-1 and

Ports swp6 <==> Remo

te  Node/s hosts-11

and Ports swp1
netcat-9090                                   host       server01
swp2:NetQBond-1      noc-pr     swp21:NetQBond-19
netcat-9090                                   host       server01
swp3:NetQBond-1      noc-se     swp21:NetQBond-19
netcat-9090                                   host       server01
swp1:swp1            tor-1      Local Node tor-1 and

Ports swp6 <==> Remo

te  Node/s hosts-11

and Ports swp1
netcat-9091                                   host       server01
swp2:NetQBond-1      noc-pr     swp21:NetQBond-19
netcat-9091                                   host       server01
swp3:NetQBond-1      noc-se     swp21:NetQBond-19
netcat-9091                                   host       server01
swp1:swp1            tor-1      Local Node tor-1 and

Ports swp6 <==> Remo

te  Node/s hosts-11
```

```
and Ports swp1
netcat-9092                                   host       server01
swp2:NetQBond-1      noc-pr     swp21:NetQBond-19
netcat-9092                                   host       server01
swp3:NetQBond-1      noc-se     swp21:NetQBond-19
netcat-9092                                   host       server01
swp1:swp1            tor-1      Local Node tor-1 and

Ports swp6 <==> Remo

te  Node/s hosts-11

and Ports swp1
```

## Check Network Traffic over a Given Protocol

You can specify either the TCP or UDP protocol when you observe a given flow of traffic on the network and want to identify which container sent or received traffic using that protocol from a given port.

```
cumulus@switch:mgmt-vrf:~$ netq hosts-11 show docker container
6.0.1.5 tcp
Container Name       Node       Proto  Port     Cont IP
Network        Host IP              Host Port
-------------------- ---------- ------ -------- -----------------
-------------- --------------------- ------------
netcat-9080          server01   tcp    9192
host           6.0.1.5/26:swp1.1004  9192
netcat-9080          server01   tcp    8182
host           6.0.1.5/26:swp1.1004  8182
netcat-9081          server01   tcp    9192
host           6.0.1.5/26:swp1.1004  9192
netcat-9081          server01   tcp    8182
host           6.0.1.5/26:swp1.1004  8182
netcat-9082          server01   tcp    8182
host           6.0.1.5/26:swp1.1004  8182
netcat-9082          server01   tcp    9192
host           6.0.1.5/26:swp1.1004  9192
netcat-9083          server01   tcp    8182
host           6.0.1.5/26:swp1.1004  8182
netcat-9083          server01   tcp    9192
host           6.0.1.5/26:swp1.1004  9192
netcat-9084          server01   tcp    9192
host           6.0.1.5/26:swp1.1004  9192
netcat-9084          server01   tcp    8182
host           6.0.1.5/26:swp1.1004  8182
netcat-9085          server01   tcp    8182
host           6.0.1.5/26:swp1.1004  8182
netcat-9085          server01   tcp    9192
host           6.0.1.5/26:swp1.1004  9192
```

```
netcat-9086          server01   tcp    9192
host           6.0.1.5/26:swp1.1004  9192
netcat-9086          server01   tcp    8182
host           6.0.1.5/26:swp1.1004  8182
netcat-9087          server01   tcp    9192
host           6.0.1.5/26:swp1.1004  9192
netcat-9087          server01   tcp    8182
host           6.0.1.5/26:swp1.1004  8182
netcat-9088          server01   tcp    9192
host           6.0.1.5/26:swp1.1004  9192
netcat-9088          server01   tcp    8182
host           6.0.1.5/26:swp1.1004  8182
netcat-9089          server01   tcp    8182
host           6.0.1.5/26:swp1.1004  8182
netcat-9089          server01   tcp    9192
host           6.0.1.5/26:swp1.1004  9192
netcat-9090          server01   tcp    8182
host           6.0.1.5/26:swp1.1004  8182
netcat-9090          server01   tcp    9192
host           6.0.1.5/26:swp1.1004  9192
netcat-9091          server01   tcp    9192
host           6.0.1.5/26:swp1.1004  9192
netcat-9091          server01   tcp    8182
host           6.0.1.5/26:swp1.1004  8182
netcat-9092          server01   tcp    9192
host           6.0.1.5/26:swp1.1004  9192
netcat-9092          server01   tcp    8182
host           6.0.1.5/26:swp1.1004  8182
```

## Show Docker Swarm Clusters and Networks

To see the elements of a Docker Swarm cluster, run:

```
cumulus@host:~$ netq show docker swarm cluster
Matching swarm records are:
Cluster Name    Num Nodes Manager Nodes
Worker Nodes
--------------- --------- ----------------------------------------
------------------------------
default         3           server01:45.0.0.20:
2377,                    server01, server02, server03
                            server02:45.0.0.24:2377
default         2           server05:45.0.0.27:
2377                     server04, server05
```

Optionally, you can output the results in JSON format:

```
cumulus@host:~$ netq show docker swarm cluster json
```

```
{
    "swarm": [
        {
            "clusterName": "default",
            "managerNodes": "server01:45.0.0.20:2377, server02:
45.0.0.24:2377",
            "workerNodes": "server01, server02, server03",
            "numNodes": 3
        },
        {
            "clusterName": "default",
            "managerNodes": "server05:45.0.0.27:2377",
            "workerNodes": "server04, server05",
            "numNodes": 2
        }
    ],
    "truncatedResult": false
}
```

You can see the changes made to the cluster:

```
cumulus@host:~$ netq server01 show docker swarm cluster changes
Matching swarm records are:
Hostname     Cluster Name    Num Nodes  Manager
Nodes                            Worker Nodes
DBState    Last changed
------------ -------------- ---------
----------------------------------------
------------------------------ ---------- --------------------
server01     default         3            server01:45.0.0.20:
2377,                server01, server02, server03   Add          12:
54.9260 ago
                                          server02:45.0.0.24:2377
server01     default         2            server01:45.0.0.20:
2377,                server01, server02                Add        14:
10.5203 ago
                                          server02:45.0.0.24:2377
```

You can show the nodes in a swarm:

```
cumulus@host:~$ netq show docker swarm node
Matching swarm records are:
Swarm Node      Node Id                    Cluster Name
Role       Docker Version    State     Availability
------------  -------------------------  --------------  -------
----------------  -------  --------------
server01        knyao3pkk8h872cep3vabrpum  default          manager
17.06.1-ce        ready     active
```

```
server02        jatmsbs71rv9nmqw5grqncqw2  default          manager
17.06.1-ce        ready      active
server03        tqrj8ro7b1ycymihquawr1szr  default          worker
17.06.1-ce        ready      active
server04        gwp89587uujywot6d2fo5vi3e  default          worker
17.06.1-ce        ready      active
server05        26boo6bak3exgi6nox8dmm2o2  default          manager
17.06.1-ce        ready      active
```

You can drill down to get information about a specific node in a swarm:

```
cumulus@host:~$ netq show docker swarm cluster node-name server04
Matching swarm records are:
Cluster Name    Num Nodes Manager Nodes
Worker Nodes
--------------- --------- ----------------------------------------
------------------------------
default         2           server05:45.0.0.27:
2377                     server04, server05
```

To view configuration at an earlier point in time, run:

```
cumulus@server05:~$ netq show docker swarm cluster node-name server04
around 10m
Matching swarm records are:
Cluster Name    Num Nodes Manager Nodes
Worker Nodes
--------------- --------- ----------------------------------------
------------------------------
default         2           server05:45.0.0.27:
2377                     server04, server05
```

For details about a Docker Swarm network, run:

```
cumulus@server01:~$ netq show docker swarm network nginx
Matching swarm records are:
Service Name    Port Mapping               Virtual IP      Network
Name
--------------  -------------------------  --------------
--------------
nginx           tcp:9080:80, tcp:9443:443  10.255.0.12/16  ingress
```

## Show Docker Service Connectivity and Impact

You can show the Docker services in a cluster:

```
cumulus@host:~$ netq show docker service
Matching service records are:
Service Name    Manager       Cluster    Mode        Replicas
Running
--------------  ---------  ---------  ----------  ----------
---------
redis           server01   default      Replicated
6          6
redis           server02   default      Replicated
6          6
```

And get detailed information about a Docker service:

```
cumulus@host:~$ netq show docker container service redis
Matching container records are:
Container Name        Hostname   Container IP      IP Masq  Network
Name   Service Name    UpTime
-------------------- ---------- ----------------- --------
-------------- --------------- ---------------
redis.1.d3k6fyx3cmdn server01   10.255.0.6          False
ingress        redis           0:07:11
3y5tr0uveuenk
redis.2.qcs7kt3si79i server02   10.255.0.11         False
ingress        redis           0:06:42
s98tdkbid9k03
redis.3.kh4bvgcpmnfg server02   10.255.0.7          False
ingress        redis           0:06:41
hvihbx2oi9xb0
redis.4.48h1jm5gq3u9 server03   10.255.0.8          False
ingress        redis           0:06:42
rmtb68lzap6kp
redis.5.kz1djm3gczst server03   10.255.0.9          False
ingress        redis           0:06:42
w8xf34oa9592z
redis.6.jicycmsbe8qj server01   10.255.0.10         False
ingress        redis           0:06:50
kw2m5c1mn7dxb
```

To see the connectivity of a given Docker service, run:

```
cumulus@host:~$ netq show docker service name redis connectivity
redis -- redis.3.kh4bvgcpmnfghvihbx2oi9xb0 -- server02 -- leaf01
                                                       -- exit01
                                                       -- leaf05
      -- redis.5.kz1djm3gczstw8xf34oa9592z -- server03 -- leaf05
                                                       -- leaf01
                                                       -- exit01
      -- redis.1.d3k6fyx3cmdn3y5tr0uveuenk -- server01 -- leaf03
                                                       -- leaf02
                                                       -- leaf01
                                                       -- exit01
      -- redis.6.jicycmsbe8qjkw2m5c1mn7dxb -- server01 -- leaf03
                                                       -- leaf02
                                                       -- leaf01
                                                       -- exit01
      -- redis.4.48h1jm5gq3u9rmtb68lzap6kp -- server03 -- leaf05
                                                       -- leaf01
                                                       -- exit01
      -- redis.2.qcs7kt3si79is98tdkbid9k03 -- server02 -- leaf01
                                                       -- exit01
                                                       -- leaf05
```

You can determine the impact on the Docker deployment in the event a host or switch goes down. The output is color coded (not shown in the example below) so you can clearly see the impact: green shows no impact, yellow shows partial impact, and red shows full impact.

```
cumulus@server01:~$ netq leaf05 show impact docker service redis
redis -- redis.3.kh4bvgcpmnfghvihbx2oi9xb0 -- server02 -- leaf01
                                                       -- exit01
                                                       -- leaf05
      -- redis.5.kz1djm3gczstw8xf34oa9592z -- server03 -- leaf05
                                                       -- leaf01
                                                       -- exit01
      -- redis.1.d3k6fyx3cmdn3y5tr0uveuenk -- server01 -- leaf03
                                                       -- leaf02
                                                       -- leaf01
                                                       -- exit01
      -- redis.6.jicycmsbe8qjkw2m5c1mn7dxb -- server01 -- leaf03
                                                       -- leaf02
                                                       -- leaf01
                                                       -- exit01
      -- redis.4.48h1jm5gq3u9rmtb68lzap6kp -- server03 -- leaf05
                                                       -- leaf01
                                                       -- exit01
      -- redis.2.qcs7kt3si79is98tdkbid9k03 -- server02 -- leaf01
                                                       -- exit01
                                                       -- leaf05
```

## View Docker Configuration in the Past

You can use the "time machine" features (see page 230) of NetQ on a Docker container, using the **`around`** and **`changes`** commands to go back in time to check the network status and identify any changes that occurred on the network. This example shows the state of the network one hour earlier.

```
cumulus@leaf01:~$ netq leaf01 show docker container adjacent around 1h
Interface            Peer Node   Peer Interface         Container
Name        IP                   Network     Service Name
```

```
-------------------- ---------- ---------------------
-------------------- -------------------- ---------- ---------------
swp6:VlanA-1         server01   mac:00:02:00:00:00:27 netcat-
9090                               host
swp6:VlanA-1         server01   mac:00:02:00:00:00:27 netcat-
9082                               host
swp6:VlanA-1         server01   mac:00:02:00:00:00:27 netcat-
9091                               host
swp6:VlanA-1         server01   mac:00:02:00:00:00:27 netcat-
9086                               host
swp6:VlanA-1         server01   mac:00:02:00:00:00:27 netcat-
9081                               host
swp6:VlanA-1         server01   mac:00:02:00:00:00:27 netcat-
9083                               host
swp6:VlanA-1         server01   mac:00:02:00:00:00:27 netcat-
9087                               host
swp6:VlanA-1         server01   mac:00:02:00:00:00:27 netcat-
9088                               host
swp6:VlanA-1         server01   mac:00:02:00:00:00:27 netcat-
9085                               host
swp6:VlanA-1         server01   mac:00:02:00:00:00:27 netcat-
9080                               host
swp6:VlanA-1         server01   mac:00:02:00:00:00:27 netcat-
9084                               host
swp6:VlanA-1         server01   mac:00:02:00:00:00:27 netcat-
9089                               host
swp6:VlanA-1         server01   mac:00:02:00:00:00:27 netcat-
9092                               host
swp7:VlanA-1         server02   mac:00:02:00:00:00:2a netcat-
8089          172.17.0.11          bridge
swp7:VlanA-1         server02   mac:00:02:00:00:00:2a netcat-
8084          172.17.0.6           bridge
swp7:VlanA-1         server02   mac:00:02:00:00:00:2a netcat-
8092          172.17.0.14          bridge
swp7:VlanA-1         server02   mac:00:02:00:00:00:2a netcat-
8083          172.17.0.5           bridge
swp7:VlanA-1         server02   mac:00:02:00:00:00:2a netcat-
8085          172.17.0.7           bridge
swp7:VlanA-1         server02   mac:00:02:00:00:00:2a netcat-
8081          172.17.0.3           bridge
swp7:VlanA-1         server02   mac:00:02:00:00:00:2a netcat-
8080          172.17.0.2           bridge
swp7:VlanA-1         server02   mac:00:02:00:00:00:2a netcat-
8086          172.17.0.8           bridge
swp7:VlanA-1         server02   mac:00:02:00:00:00:2a netcat-
8088          172.17.0.10          bridge
swp7:VlanA-1         server02   mac:00:02:00:00:00:2a netcat-
8082          172.17.0.4           bridge
swp7:VlanA-1         server02   mac:00:02:00:00:00:2a netcat-
8091          172.17.0.13          bridge
swp7:VlanA-1         server02   mac:00:02:00:00:00:2a netcat-
8090          172.17.0.12          bridge
```

```
swp7:VlanA-1         server02   mac:00:02:00:00:00:2a netcat-
8087          172.17.0.9           bridge
swp8:VlanA-1         server03   mac:00:02:00:00:00:2d netcat-
8091          172.17.0.13          bridge
swp8:VlanA-1         server03   mac:00:02:00:00:00:2d netcat-
8083          172.17.0.5           bridge
swp8:VlanA-1         server03   mac:00:02:00:00:00:2d netcat-
8087          172.17.0.9           bridge
swp8:VlanA-1         server03   mac:00:02:00:00:00:2d netcat-
8082          172.17.0.4           bridge
swp8:VlanA-1         server03   mac:00:02:00:00:00:2d netcat-
8080          172.17.0.2           bridge
swp8:VlanA-1         server03   mac:00:02:00:00:00:2d netcat-
8092          172.17.0.14          bridge
swp8:VlanA-1         server03   mac:00:02:00:00:00:2d netcat-
8086          172.17.0.8           bridge
swp8:VlanA-1         server03   mac:00:02:00:00:00:2d netcat-
8084          172.17.0.6           bridge
swp8:VlanA-1         server03   mac:00:02:00:00:00:2d netcat-
8088          172.17.0.10          bridge
swp8:VlanA-1         server03   mac:00:02:00:00:00:2d netcat-
8090          172.17.0.12          bridge
swp8:VlanA-1         server03   mac:00:02:00:00:00:2d netcat-
8085          172.17.0.7           bridge
swp8:VlanA-1         server03   mac:00:02:00:00:00:2d netcat-
8089          172.17.0.11          bridge
swp8:VlanA-1         server03   mac:00:02:00:00:00:2d netcat-
8081          172.17.0.3           bridge
```

# Automate Common and Repetitive Tasks

NetQ commands can also be run in an automation tool, such as Ansible, Chef, or Puppet; depending on the outcome of the automation tests, the script can either continue the deployment, or roll back the changes until the issues are addressed.

## Contents

This topic describes how to...

## Run NetQ Commands in Automation Scripts

Using NetQ for preventative care of your network pairs well with automation scripts and playbooks to prevent errors on your network before deploying the configuration to your production network. Red Hat Ansible, Chef and Puppet automation tools, as well as custom automation scripts, all support scripting with NetQ commands.

For example, you can use NetQ in your Ansible playbook to help you configure your network topology. The playbook could pull in BGP data in JSON format before it starts creating the topology:

```
- hosts: localhost leaf spine
  gather_facts: False
  tasks:
     - name: Gather BGP Adjanceny info in JSON format
       local_action: command netq show bgp json
       register: result
       #delegate_to: localhost
       run_once: true
```

Based on the outcome, the playbook can then respond appropriately. Later, it can check IP addresses to verify the connections:

```
#ipv6 address check
     - name: run ipv6check on broken_dict
       command: netq show ipv6 addresses {{item.key}} {{item.value}}
json
       with_dict: "{{broken_dict}}"
       register: command_outputs
       delegate_to: localhost
       run_once: true
```

# Early Access Features

NetQ has early access features that provide advanced access to new functionality before it becomes generally available. The following features are early access in NetQ 1.4:

- Extend NetQ with Custom Commands (see page 199)
- Query the NetQ Database (see page 205)
- Collect Interface Statistics (see page 217)

In NetQ 1.4, early access features are bundled into the `netq-apps` package; there is no specific EA package like there typically is with Cumulus Linux.

You enable these early access features by running the `netq config add experimental` command. You disable the early access features by running the `netq config del experimental` command.

Refer to Configure Optional NetQ Capabilities to access the Image and Provisioning Management application.

## Extend NetQ with Custom Commands

NetQ provides the ability to codify playbooks and extend NetQ with custom commands for use cases specific to your network.

The summary of steps required to do this is a follows:

- The extensions must be written in Python or Cython.
- The commands need to be added must use `network doctopt`.
- The .py file (or the compiled .so if using Cython) is now copied to /usr/lib/python2.7/dist-packages /netq_apps/modules/addons.
- Enable the add-ons with the `netq config add addons` command
- Check that your command works by typing `netq <TAB>`

NetQL is an early access feature in Cumulus NetQ 1.3 and later.

### Contents

This topic describes...

## Sample File with Custom Command

To help you get started, here is the Hello World of NetQ command extension:

**Sample Hello World**

```
'''
hello: A netq app hello world module
Usage:
   netq hello [json]
Options:
   hello                          : Hello world experimental
'''
import json
from netq_apps.modules import NetqModule, RC_SUCCESS, RC_FAIL
app = NetqModule()

@app.route('hello')
def cli_hello_world(cli, netq):
    '''My very own hello'''
    jsonify = cli.get('json')
    if jsonify:
       print json.dumps({'greeting': 'Hello World'})
    else:
       print 'Hello World'
    return RC_SUCCESS
```

Let's break down each part of the code.

### *Command Specification With Help*

The lines at the start of the file within the triple quotes (''') constitute what is called the *docstring* of the file or module. `network-docopt`, the Python library that builds the command parser for NetQ, uses the information provided in the *docstring*. Specifically, everything between **Usage** and **Options** is considered a command specification. In this case, `netq hello` is the only command specified in the file. The command MUST start with the word `netq`. Every `netq` command follows the following structure:

```
netq [<hostname>] <verb> <object> <filters>
```

For example, here is the sample for `show vlan`:

```
netq [<hostname>] show vlan [<1-4096>] [around <text-time>] [json]
```

The <*hostname*> option is used to filter results to just the specified host; hostname can also be a regular expression. The <*verb*> is *show*, the <*object*> is *vlan* and the remaining parameters are filters to viewing the data.

For example, if you wanted to extend hello world by passing an optional greeting, modify the usage to be:

```
netq hello <text-greeting>
```

`network-docopt` understands a few parameter types and validates them before passing them to your code. Some common ones are:

- **<hostname>**: A host known to NetQ
- **<remote-interface>**: An interface on the specified host known to NetQ
- **<text>**: Any free text, but has to be a single word or delimited within quotes
- **<ip>**, **<ip/prefixlen>**: IPv4 or IPv6 address, with prefix length in the second case
- **<ipv4>**, **<ipv4/prefixlen>**: IPv4 address, with prefix length in the second case
- **<ipv6>**, **<ipv6/prefixlen>**: IPv6 address, with prefix length in the second case
- **<wildcard>**: All the remaining text
- Valid number range: Such as **<1-4096>** to limit the allowed range

So in the VLAN example above, specifying a VLAN value outside the 1-4096 range results in an error, with command unknown and a help message indicating that you need to specify a value between 1 and 4096. For hosts and interfaces used with <*hostname*> and <*remote-interface*>, NetQ automatically provides tab completion.

To display meaningful help associated with a keyword, add the help for the command via the **Options** section. In the example code above, the object *hello* has the help text "Hello world experimental". This text is displayed when the user types `netq <TAB>`, as shown in the following example:

```
cumulus@switch:~$ netq
<hostname> : Type first char of netq host for dynamic completion
check : Perform fabric-wide checks
config : Configuration
example : Show examples of usage and workflow
hello : Hello world experimental
help : Show usage info
resolve : Annotate input with names and interesting info
show : Show fabric-wide info about specified object
trace : Control Path Trace
cumulus@torc-11:mgmt-vrf:~$ netq
```

⚠ Any help you provide here overrides the help provided for the keyword by a module loaded previously.

### *Associating the Command with the Function*

After configuring the command, you need to associate or *bind* that command with the function to be called when a user runs the command. This is done by using decorators to functions similar to how other CLI builders or web servers work.

First, create an instance of the class `NetqModule()` called *app*. Then associate the function to the appropriate command via the decorator `@app.route`. As shown in the example above, the function `cli_hello_world()` is decorated to indicate that it is the function to call for the command `hello`. The function takes two parameters: *cli* and *netq*. Usage of these parameters is discussed in the next section.

Keep in mind the following when matching the command to the function:

- If a prior binding has already been assigned to a command, the newer binding will fail. By default, modules in the core NetQ code take precedence over early access modules, which take precedence over the modules defined in addons directory.
- The command string can be as small as possible. For example, the commands `netq hello json` and `netq hello` can be handled by different functions or by the same function. The NetQ command parser does a longest match first to determine which of the competing functions is assigned to execute a command. The command parser supports up to three string matches. In other words, `show ip address` is supported, but `show ip address json` is not. Such longer command strings bound to a function either silently fail or a shorter string version is matched.

### *Using the cli and netq Parameters*

The function that is called to execute a command expects to received two parameters, *cli* and *netq*, in the order shown in the example above.

*cli* is a dictionary containing the parameters provided by the user on the command line. *netq* contains the timestamps provided by the user, if any. Any other object within NetQ can be ignored. The timestamps are provided to query NetQ objects around a specific time or in a time window.

The example shows how to extract the value provided by the user at the command line from *cli*. Since *json* is a keyword, getting the key *json* from *cli* lets you to determine if the user specified *json* at the command line or not. If the user did not specify *json* at the command line, `cli.get('json')` returns *None*, whereas if the user did specify *json*, then `cli.get('json')` returns the string "json". Thus, if the user wants to specify a parameter along with a keyword, for example, as shown in `netq show macs [vlan <1-4096>]`, then the value of the VLAN to search for a MAC address can be found using `cli.get('<1-4096>')`, not via `cli.get('vlan')`.

### *Return Values*

The function returns either *RC_SUCCESS* if successful or *RC_FAIL* if not. The code snippet shows how to import these values from the standard NetQ libraries.

## Query the NetQ Database

While the code snippet above was sufficient to illustrate the general skeleton, if you want to extend the commands, you typically will want to add meaningful functionality such as querying the database and displaying some more meaningful information. For example, consider a new command called `show ip-routes`, which displays the route information available in the database, but with a different set of fields than shown via `show ip routes`. The code to do so is shown below.

```
"""
routes.py: NetQ app module for processing IPv4/v6 routes
Usage:
   netq <hostname> show myroutes [vrf <vrf>] [json]
Options:
   myroutes                               : IPv4/v6 routes
"""
from __future__ import absolute_import
from collections import OrderedDict

from netq_apps.modules import NetqModule, RC_SUCCESS
from netq_apps.cmd.netq import netq_show

from netq_lib.orm.redisdb.models import Route

app = NetqModule()

@app.route('show myroutes')
@netq_show
def cli_show_myroutes(cli, netq, context):
    '''MY very own show routes'''
    hostname = cli.get('<hostname>') or '*'
    vrf = cli.get('<vrf>') or '*'
    context.col_sizes = [16, 8, 32, 26, 16]
    entries = Route.query.filter(timestamp=netq.start_time,
                                 endtimestamp=netq.end_time,
                                 hostname=hostname, vrf=vrf)
    for entry in entries:
        out = OrderedDict()
        if isinstance(entry, tuple):
            route = entry[0]
        else:
            route = entry
        if not route.nexthops:
            route.nexthops = [['None', 'Local']]
        nexthops = ', '.join(
            '%s: %s' % (nh[0], nh[1]) if nh[0] != 'None' else '%s' %
nh[1]
            for nh in sorted(route.nexthops)
        )

        out['Hostname'] = route.hostname
        out['Protocol'] = route.protocol
        out['Prefix'] = route.prefix
        out['Nexthops'] = nexthops
        out['Last Changed'] = route.timestamp
        yield out
```

Much of this code is similar to the hello world example, but the new items are discussed below.

### *The Imports*

There are two additional imports, one for *netq_show* and the other for *Route*.

### *netq_show*

*netq_show* is the decorator that takes care of wrapping the output in a format native to NetQ. For example, it generates the JSON for you automatically, so that you don't have to write a JSON output generator just to support JSON and you don't have to worry about supporting the tabular format, displaying rotten nodes in a different color and so on. All you have to do is generate output in the form of an `OrderedDict` and `yield` for every entry. The `OrderedDict` ensures that the columns are displayed in the order provided in the code. The column headers are generated from the dictionary key, as are the JSON keys.

By wrapping the code with the *netq_show*, all these display complexities are covered for you.

### *Route*

*Route* is the database object that holds all the pertinent information about a route. Its contents are defined in the `/usr/lib/python2.7/dist-packages/netq_lib/orm/redisdb/models.py` file. There are other database objects defined in the file, but this example only involves the *Route* object.

### *The Function Handler*

The function that satisfies the command `show myroutes` is *cli_show_myroutes*, and because of the decorator, takes an additional input parameter, *context*. It's mainly used to pass things between the main NetQ command module and the specific modules, such as this one. This particular case uses the *context* to update the column sizes to be used in the display.

### *The Query Functions*

The meat of the code is the query. Objects are queried using the model of *<object>.query.<query function>. T* his particular example uses *filter* as the query function, as shown by the `Route.query.filter()` call. The filter function produces output filtered by the parameters specified in the keyword arguments passed. For example, the *hostname* keyword argument restricts the results returned by the query function to only those on the specified host. The list of keys that can be specified for an object are listed under the object's definition in the aforementioned `models.py` file under the function `key_fmt()`. A look at that function for the *Route* object shows that the key fields are: hostname, prefix, route type, routing table id, ipv4/v6 route and, If the entry is originated on this node, the protocol that added this route and the VRF name qualifier. The values returned include all the key fields plus the fields shown in the `val_fmt()` function for the object.

The other useful query functions are:

- `query.get()`: which returns just the first element matching the parameters specified.
- `query.latest()`: which returns the latest element matching the parameters specified, and does not take any time parameters.
- `query.count()`: which returns a count of the matching elements instead of the elements themselves.

The filter query functions return an iterator and thus is lazy about retrieving data from the back end. You can stop whenever you want in the iteration. `query.get()` and `query.latest()` both return a single object of the type the query is on while `query.count()` returns an integer.

## Debugging

Inevitably when writing code, coding errors need to be debugged and the fixes tried again. When a module doesn't load or returns an error, it is reported in the `netqd.log`, usually kept under `/var/log` (unless you modified the location). Deploying the module on one node doesn't mean it is automatically available on all nodes. You must copy it to all the required nodes.

To reload the modules after making fixes, run the command `netq config reload parser`.

## Caveats

This feature is an early access feature, and must be treated as such. There may be obscure failures which will require Cumulus Networks engineering intervention to investigate. Finally, please save the modules you write. If you reinstall the `netq-apps` package, your modules may get overwritten when you install the new package. One of the next releases of NetQ should provide the ability to store these modules under `/usr/local/lib`, to keep them from being affected by package management.

# Query the NetQ Database

You can query for even more NetQ data using the SQL-like NetQ Query Language (NetQL) so you can conduct your own custom analysis or otherwise extend NetQ functionality for your specific environment without having to write your own custom code. NetQL directly queries the NetQ database for data that isn't exposed via the check, show and trace commands.

> ⓘ NetQL is an early access feature in Cumulus NetQ 1.3 and later.

## Contents

This topic describes...

## Commands

- netq query
- netq config add|del experimental

## Enable NetQL

Since NetQL is an early access feature, you must enable the experimental option of the NetQ CLI:

```
cumulus@switch:~$ netq config add experimental
```

## Usage

NetQL is a generic structured query language modeled on SQL. The general command syntax is:

```
cumulus@switch:~$ netq query 'SELECT <fields> FROM <tables> WHERE
<conditions> GROUP BY <fields> ORDER BY <fields>[asc|desc]' [json]
```

NetQL supports tab completion. When you press the TAB key after typing *FROM*, a list of objects appears from which you can select.

Between the SELECT, FROM, WHERE, GROUP BY and ORDER BY keywords are the following variables:

| Variable | Definition |
|---|---|
| <fields> | One or more key or non-key fields from one of the NetQ database tables. |
| <tables> | One or more tables in the NetQ database. |
| <conditions> | Qualifiers to the data being queried. |

These items are defined below.

The following is a real-world example:

```
cumulus@switch:~$ netq query 'SELECT hostname, peer_name,
peer_hostname, asn, peer_asn, state FROM BgpSession'
hostname    peer_name        peer_hostname    asn     peer_asn
state
----------  ---------------  ---------------  ------  ----------
-----------
leaf01      swp3             spine01          655536  655435
Established
leaf01      swp6             firewall01       655536  655538
Established
leaf01      swp7             firewall02       655536  655539
Established
leaf01      swp4             spine02          655536  655435
Established
leaf01      swp5             spine03          655536  655435
Established
leaf01      swp6.4           firewall01       655536  655538
Established
```

```
...
```

The keywords are not case sensitive, so you can use *SELECT*, *Select* or *select*. The all caps usage is for easier parsing of the queries.

## Tables and Fields

One example field is *hostname*, which is present in every table. Example tables include Route, Link and BgpSession.

> ⚠ At this time, you cannot have multiple copies of the same table.

You can get a list of all the tables known to NetQ by running this command:

```
cumulus@switch:~$ netq query show tables
Class                    Key Fields
-------------------
----------------------------------------------------------------------
-----------------
ASIC                     hostname, vendor, model, model_id, core_bw, ports
Address                  hostname, ifname, prefix, mask, is_ipv6, vrf
BgpSession               hostname, peer_name, asn, vrf
Board                    hostname, vendor, model, base_mac, part_number,
mfg_date, serial_number, label_revision
CPU                      hostname, arch, nos, model, max_freq, mem_total
ClagSession              hostname, clag_sysmac
Description              hostname, objtype, descrid
Disk                     hostname, name, size, d_type, vendor, transport,
rev, model
...
```

You can get a list of all the fields in a table by running this command:

```
cumulus@switch:~$ netq query show fields BgpSession
Table               Key
Fields
Value Fields
---------------
----------------------------------------------------------------------
------------------------------
----------------------------------------------------------------------
------------------------------
BgpSession          hostname, peer_name, asn,
vrf
state, peer_router_id, peer_asn, peer_hostname, reason,
ipv4_pfx_rcvd, ipv6_pfx_rcvd,
evpn_pfx_rcvd, timestamp, last_reset_time, conn_estd, conn_dropped,
upd8_rx, vrfid, upd8_tx,

up_time, tx_families, objid, rx_families, active, deleted


cumulus@switch:~$ netq query show fields Port
Table           Key
Fields
Value Fields
---------------
----------------------------------------------------------------------
------------------------------
----------------------------------------------------------------------
------------------------------
Port            hostname,
ifname
identifier, speed, autoneg, state, transreceiver, connector, length,
vendor_name, part_number,

serial_number, fec, supported_fec, advertised_fec, active, deleted,
timestamp
```

The *fec*, *supported_fec*, and *advertised_fec* are new in NetQ 1.4.0.

An example query on a single table is:

```
cumulus@switch:~$ netq query 'SELECT hostname, peer_name,
peer_hostname, asn, peer_asn, state FROM BgpSession'
hostname    peer_name        peer_hostname    asn     peer_asn
state
----------  ---------------  ---------------  ------  ----------
-----------
exit01      swp3             spine01          655536  655435
Established
exit01      swp6             firewall01       655536  655538
Established
exit01      swp7             firewall02       655536  655539
Established
exit01      swp4             spine02          655536  655435
Established
exit01      swp5             spine03          655536  655435
Established
exit01      swp6.4           firewall01       655536  655538
Established
...
```

NetQL displays the values of the specified fields in tabular output.

## Conditions

Conditions select what data is presented. An example of a condition is *hostname="leaf01"*. Use double quotes ("") for the specific values you want to match on. You can also use != to indicate non-matching entries.

AND is the only condition supported currently. You cannot perform queries using parenthesized conditions at this time.

An example conditional query is:

```
cumulus@switch:~$ netq query 'SELECT hostname, peer_name,
peer_hostname, asn, peer_asn, state FROM BgpSession WHERE hostname="
*1" AND peer_name="swp3*"'
hostname    peer_name    peer_hostname    asn     peer_asn    state
----------  -----------  ---------------  ------  ----------
-----------
exit01      swp3         spine01          655536  655435
Established
exit01      swp3.4       spine01          655536  655435
Established
exit01      swp3.2       spine01          655536  655435
Established
exit01      swp3.3       spine01          655536  655435
Established
spine01     swp3         leaf01           655435  655561
Established
spine01     swp3.4       leaf01           655435  655561
Established
spine01     swp3.2       leaf01           655435  655561
Established
spine01     swp3.3       leaf01           655435  655561
Established
leaf01      swp3         spine01          655559  655435
Established
leaf01      swp3.4       spine01          655559  655435
Established
leaf01      swp3.2       spine01          655559  655435
Established
leaf01      swp3.3       spine01          655559  655435
Established
leaf01      swp3         spine01          655561  655435
Established
leaf01      swp3.4       spine01          655561  655435
Established
leaf01      swp3.2       spine01          655561  655435
Established
leaf01      swp3.3       spine01          655561  655435
Established
```

```
leaf02      swp3         spine01          655563  655435
Established
leaf02      swp3.4       spine01          655563  655435
Established
leaf02      swp3.2       spine01          655563  655435
Established
leaf02      swp3.3       spine01          655563  655435
Established
```

## Group Results

When you want to see not only the value of a field, but also the aggregated output such as a count or sum, you must specify on which field to aggregate the data. For example, to get the number of peer ASNs for each host, the query is:

```
cumulus@switch:~$ netq query 'SELECT hostname, count(peer_asn) FROM
BgpSession GROUP BY hostname'
hostname      count(peer_asn)
----------  -----------------
exit01                     20
exit02                     20
spine01                    32
spine02                    32
spine03                    32
leaf01                     12
leaf02                     12
leaf03                     13
leaf04                     13
leaf05                     13
leaf06                     13
```

## Order Results

You can specify which columns you want the output sorted on using the "ORDER BY" clause of the query. The general format of the ORDER BY clause is:

```
ORDER BY <field1> [ASC|DESC] [<field2> [ASC|DESC]...]
```

As an example, the output of the query in the previous section can be sorted by the COUNT followed by hostname, as follows:

```
cumulus@switch:~$ netq query 'SELECT hostname, COUNT(peer_asn) FROM
BgpSession GROUP BY hostname ORDER BY COUNT(peer_asn)'
hostname      count(peer_asn)
----------  -----------------
leaf01                     12
```

```
leaf02                    12
leaf03                    13
leaf04                    13
leaf05                    13
leaf06                    13
exit02                    20
exit01                    20
spine01                   32
spine03                   32
```

This sorts the count in ascending order, which is the default and does not have to be specified. To sort by descending order, use the DESC keyword, as follows:

```
cumulus@switch:~$ netq query 'SELECT hostname, COUNT(peer_asn) FROM
BgpSession GROUP BY hostname ORDER BY count(peer_asn) DESC, hostname'
hostname      count(peer_asn)
----------  -----------------
spine01                   32
spine02                   32
spine03                   32
exit01                    20
exit02                    20
leaf03                    13
leaf04                    13
leaf05                    13
leaf06                    13
leaf01                    12
leaf02                    12
```

The DESC keyword applies only to the field preceding it. Thus, in the example above, the output is sorted by the nodes with the most peer ASNs, and nodes with the same number of peer ASNs are sorted based on the ascending alphabetical sort of the hostname. If you want the hostnames to be also sorted in reverse alphabetical order, follow the hostname field also with the DESC keyword, as follows:

```
cumulus@switch:~$ netq query 'SELECT hostname, COUNT(peer_asn) FROM
BgpSession GROUP BY hostname ORDER BY count(peer_asn) DESC, hostname
DESC'
hostname      count(peer_asn)
----------  -----------------
spine03                   32
spine02                   32
spine01                   32
exit02                    20
exit01                    20
leaf06                    13
leaf05                    13
leaf04                    13
leaf03                    13
leaf02                    12
```

```
leaf01                     12
```

The `distinct` keyword, when used with `count`, counts only distinct or unique values. For example, the following queries show the total number of ASNs in use in the fabric, the number of distinct ASNs, and then the list of each ASN:

```
cumulus@switch:~$ netq query 'SELECT COUNT(peer_asn) FROM BgpSession'
  count(peer_asn)
-----------------
              228
cumulus@switch:~$ netq query 'SELECT COUNT(distinct peer_asn) FROM
BgpSession'
  count(distinct peer_asn)
--------------------------
                        11
cumulus@switch:~$ netq query 'SELECT set(peer_asn) FROM BgpSession'
set(peer_asn)
----------------------------------------------------------------------
----------------------------------
set([655435L, 655559L, 655560L, 655561L, 655562L, 655563L, 655564L,
655536L, 655537L, 655538L, 655539L])
```

## Regular Expressions

You can use any regular expression that Redis supports. They include, but are not limited to, the following examples:

- h?llo matches hello, hallo and hxllo
- h*llo matches hllo and heeeello
- h[ae]llo matches hello and hallo, but not hillo
- h[^e]llo matches hallo, hbllo, ... but not hello
- h[a-b]llo matches hallo and hbllo

For example:

```
cumulus@switch:~$ netq query 'SELECT hostname, peer_name,
peer_hostname, asn, peer_asn, state FROM BgpSession WHERE hostname="
*1" AND peer_name="swp[34]"'
hostname    peer_name    peer_hostname    asn     peer_asn    state
----------  -----------  ---------------  ------  ----------
-----------
exit01      swp3         spine01          655536  655435
Established
exit01      swp4         spine02          655536  655435
Established
firewall01  swp4         exit02           655538  655537
Established
```

```
firewall01  swp3         exit01           655538  655536
Established
spine01     swp3         leaf01           655435  655561
Established
spine01     swp4         leaf02           655435  655562
Established
leaf01      swp3         spine01          655559  655435
Established
leaf01      swp4         spine02          655559  655435
Established
```

## JSON Output

Any command's output can be returned in JSON format by ending the command with the optional `json` keyword, as follows:

```
cumulus@hostd-11:~$ netq query 'select hostname, peer_name,
tx_families, rx_families from BgpSession where hostname=tor-1 and
peer_name=swp3' json
[
    {
        "tx_families":[
            "ipv4",
            "ipv6",
            "evpn"
        ],
        "hostname":"tor-1",
        "rx_families":[
            "ipv4",
            "ipv6",
            "evpn"
        ],
        "peer_name":"swp3"
    }
]
```

Here's the output without JSON:

```
cumulus@hostd-11:~$ netq query 'select hostname, peer_name,
tx_families, rx_families from BgpSession where hostname=tor-1 and
peer_name=swp3'
hostname           peer_name tx_families rx_families
------------------ --------- ----------- -----------
tor-1              swp3      [u'ipv4',   [u'ipv4',
                             u'ipv6',    u'ipv6',
                             u'evpn']    u'evpn']
```

## Recommended Tables and Fields

The following tables and fields are supported as part of Early Access.

There are key fields and value fields for each table. You can get a list of the key and value fields by running the `netq show query fields` command. For example:

```
cumulus@hostd-11:~$ netq query show fields Temp
Table        Key Fields                               Value Fields
------------ ----------------------------------------
----------------------------------------------------------
Temp         hostname, s_name, s_desc                 timestamp,
s_state, s_prev_state, s_input, s_msg, s_crit,
                                                      s_min, s_max,
s_lcrit
```

| Table | Key Fields | Value Field |
|---|---|---|
| ASIC | hostname, vendor, model, model_id, core_bw, ports | timestamp |
| Address | hostname, ifname, prefix, mask, is_ipv6, vrf | timestamp, active, deleted |
| BgpSession | hostname, peer_name, asn, vrf | state, peer_router_id, peer_asn, peer_hostname, reason, ipv4_pfx_rcvd, ipv6_pfx_rcvd, evpn_pfx_rcvd, timestamp, last_reset_time, conn_estd, conn_dropped, upd8_rx, vrfid, upd8_tx, up_time, tx_families, objid, rx_families, active, deleted |
| Board | hostname, vendor, model, base_mac, part_number, mfg_date, serial_number, label_revision | timestamp |
| CPU | hostname, arch, nos, model, max_freq, mem_total | timestamp |
| ClagSession | hostname, clag_sysmac | peer_role, role, peer_state, peer_if, backup_ip, backup_ip_active, single_bonds, dual_bonds, timestamp, conflicted_bonds, vxlan_anycast, proto_down_bonds, active, deleted |
| Description | | description, timestamp, active, deleted |

| Table | Key Fields | Value Field |
|---|---|---|
| | hostname, objtype, descrid | |
| Disk | hostname, name, size, d_type, vendor, transport, rev, model | timestamp |
| DockerContainer | hostname, name, image, network_name, ip, mac, service_name | timestamp, container_id, status, network_id, gw, prefix_len, port_list, service_id, start_time, active, deleted |
| DockerHost | hostname, docker_version | images, containers, containers_running, ip_forwarding, timestamp, active, deleted |
| DockerNetwork | hostname, network_name, driver | gateway, parent_interface, vxlan_id, network_id, mtu, host_binding, ipv6_enabled, ip_masquerade, encrypted, default_bridge, ipam_driver, subnet, container_list, timestamp, active, deleted |
| DockerPortMap | hostname, name, container_ip, proto, container_port, host_ip, host_port, network_name | timestamp, container_id, network_id, image, mac, node_id, active, deleted |
| DockerService | hostname, service_name, mode | image, replicas, parallelism, service_id, port_list, network_list, vip, version, timestamp, active, deleted |
| DockerSwarmCluster | hostname, cluster_name | docker_version, cluster_version, cluster_id, num_nodes, num_managers, managers, timestamp, nodes, active, deleted |
| DockerSwarmNode | hostname, cluster_name, node_name | timestamp, docker_version, cluster_id, node_id, node_state, role, plugins, availability, active, deleted |
| Fan | hostname, s_name, s_desc | timestamp, s_state, s_prev_state, s_input, s_msg, s_max, s_min |
| License | hostname, name | state, license, timestamp |
| Link | hostname, ifname, kind, vni, master | |

| Table | Key Fields | Value Field |
|---|---|---|
| | | admin_state, oper_state, managed, mtu, ifindex, is_vlan_filtering, timestamp, vlans, access_vlan, localip, down_reason, vrf, rt_table_id, parent_if, stp_state, mac_address, dstport, learning_en, objid, arp_suppress_en, active, deleted |
| Liveness | hostname | hostname |
| Lldp | hostname, ifname, peer_hostname | peer_ifname, lldp_peer_bridge, lldp_peer_router, lldp_peer_station, lldp_peer_os, lldp_peer_osv, timestamp, active, deleted |
| LnvSession | hostname, role | role, snd_ip, rd_peers, snd_peers, vnis, state, repl_mode, version, active, deleted, timestamp |
| MacFdb | hostname, mac_address, vlan | origin, nexthop, dst, port, timestamp, is_remote, is_static, active, deleted |
| Memory | hostname, name, size, speed, m_type, vendor, serial_number | timestamp |
| MstpInfo | hostname, bridge_name | root_port_name, topo_chg_ports, time_since_tcn, topo_chg_cntr, ports, edge_ports, state, network_ports, disputed_ports, bpduguard_ports, root_bridge, bridge_id, bpduguard_err_ports, ba_inconsistent_ports, bpdufilter_ports, is_vlan_filtering, active, deleted, timestamp |
| Neighbor | hostname, ifname, ip_address, mac_address, is_ipv6, vrf | ifindex, timestamp, is_remote, active, deleted |
| Node | hostname | lastboot, sys_uptime, last_reinit, ntp_state, version |
| Ntp | hostname | current_server, stratum, ntp_sync, timestamp, ntp_app, active, deleted |
| OS | hostname | timestamp, name, version, version_id |
| OspfIf | hostname, ifname, area | network_type, timestamp, nbr_count, if_up, nbr_adj_count, is_unnumbered, is_passive, cost, mtu, dead_time, rexmit_time, hello_time, router_id, area, active, deleted |
| OspfNbr | hostname, ifname, peer_id, area | state, timestamp, ifname, is_ipv6, peer_addr, area, active, deleted |

| Table | Key Fields | Value Field |
|---|---|---|
| PSU | hostname, s_name | timestamp, s_state, s_prev_state, s_msg |
| Port | hostname, ifname | identifier, speed, autoneg, state, transreceiver, connector, length, vendor_name, part_number, serial_number, fec, supported_fec, advertised_fec, deleted, timestamp |
| Route | hostname, prefix, route_type, rt_table_id, is_ipv6, origin, protocol, vrf | nexthops, src, timestamp, active, deleted |
| Services | hostname, name, vrf | is_enabled, is_active, status, is_monitored, start_time, pid, timestamp, active, deleted |
| Temp | hostname, s_name, s_desc | timestamp, s_state, s_prev_state, s_input, s_msg, s_crit, s_max, s_min, s_lcrit |
| VxlanRemoteDest | hostname, vni, rdst | vni, rdst, active, deleted, timestamp |

# Collect Interface Statistics

Switches collect statistics about the performance of their interfaces. The NetQ Agent on each switch collects these statistics every 15 seconds by reading `/proc/net/dev`, and then sending them to the NetQ Telemetry Server where it is stored in its InfluxDB in `procnetdev`. The Telemetry Server `netq-stats-pushd` service manages the receipt and storage of the statistics.

Only statistics for physical interfaces are collected; NetQ does not collect statistics for non-physical interfaces, such as bonds, bridges, and VXLANs. Specifically, the NetQ Agent collects the following interface statistics:

- **Transmit**: tx_bytes, tx_carrier, tx_colls, tx_drop, tx_errs, tx_packets
- **Receive**: rx_bytes, rx_drop, rx_errs, rx_frame, rx_multicast, rx_packets

Currently, these statistics are available for view in third-party analytic tools.

ⓘ The collection of interface statistics is an early access feature in Cumulus NetQ 1.3 and later.

## Contents

This topic describes how to...

## Configure Interface Statistic Collection

InfluxDB is installed in its own container by default on the Telemetry Server. The `netq-stats-pushd` service is also installed by default, but must be enabled. You also need to enable statistics collection on every node for which you want to gather statistics.

To set up interface statistic collection:

1. Enable and start the `netq-stats-pushd` service on the Telemetry Server.

```
cumulus@ts:~$ sudo systemctl enable netq-stats-pushd.service
cumulus@ts:~$ sudo systemctl start netq-stats-pushd.service
```

2. Verify the service is running.

```
cumulus@ts:~$ sudo systemctl status netq-stats-pushd.service
 netq-stats-pushd.service - NetQ Stats Storage daemon
   Loaded: loaded (/lib/systemd/system/netq-stats-pushd.service;
enabled)
   Active: active (running) since Mon 2017-11-27 00:51:09 UTC;
6s ago
 Main PID: 30550 (netq-stats-push)
```

3. On every node you want to monitor: log in to each node, configure the NetQ Agent to collect the statistics, and then restart the agent.

```
cumulus@ts:~$ ssh spine01
cumulus@spine01:~$ netq config add agent stats
cumulus@spine01:~$ netq config restart agent
```

> Optionally, you can automate the configuration and restart of each node using an IT automation tool, such as Ansible.

As each NetQ Agent is restarted, the `netq-stats-pushd` service starts collecting interface statistics and the agent pushes them to the database on the Telemetry Server.

## View Interface Statistics in Grafana

You can use the open platform Grafana analytics and monitoring tool to view the interface statistics collected by the NetQ Agents. This is accomplished by installing the tool on the Telemetry Server and then configuring the tool to access the NetQ InfluxDB.

To set up Grafana to view NetQ interface statistics:

1. Using a text editor of your choice, add the repository for Grafana.

```
cumulus@ts:~$ sudo vi /etc/apt/sources.list
[sudo] password for cumulus:**********

...
deb https://packagecloud.io/grafana/stable/debian/ jessie main
...
```

2. Install the package cloud key.

```
cumulus@ts:~$ curl https://packagecloud.io/gpg.key | sudo apt-
key add -
```

3. Install, start, and enable the package.

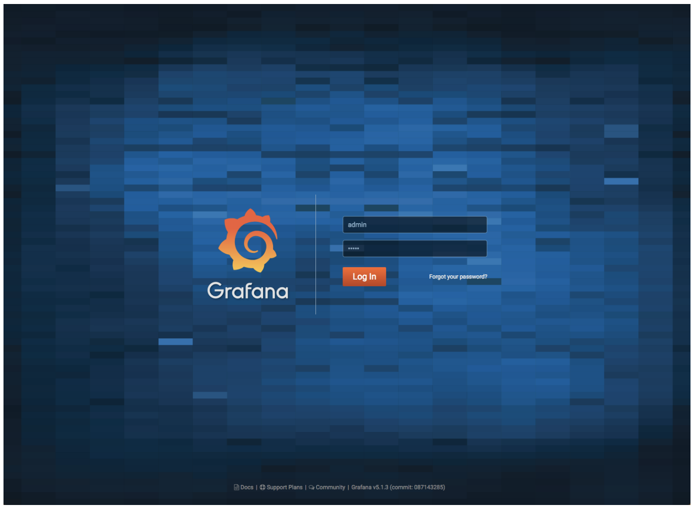
```
cumulus@ts:~$ sudo apt-get update
cumulus@ts:~$ sudo apt-get install grafana
cumulus@ts:~$ sudo systemctl daemon-reload
cumulus@ts:~$ sudo systemctl start grafana-server
cumulus@ts:~$ sudo systemctl enable grafana-server
```

4. Verify Grafana is running.

```
cumulus@ts:~$ sudo systemctl status grafana-server
 grafana-server.service - Grafana instance
   Loaded: loaded (/usr/lib/systemd/system/grafana-server.
service; disabled)
   Active: active (running) since Mon 2018-06-18 20:14:38 UTC;
7s ago
     Docs: http://docs.grafana.org
Main PID: 5755 (grafana-server)
  CGroup: /system.slice/grafana-server.service
          5755 /usr/sbin/grafana-server --config=/etc/grafana
/grafana.ini --pidfile=/var/...
```

> The Grafana GUI is accessed through port 3000 by default. If you are running Grafana on a simulation server, you may need to modify forwarding rules in IPtables to allow access to port 3000.

5. Open Grafana.
   a. In a web browser, enter the <*Telemetry_Server_IPaddress*:3000> in the address field.
   b. Log in with a user name of *admin* and a password of *admin*.



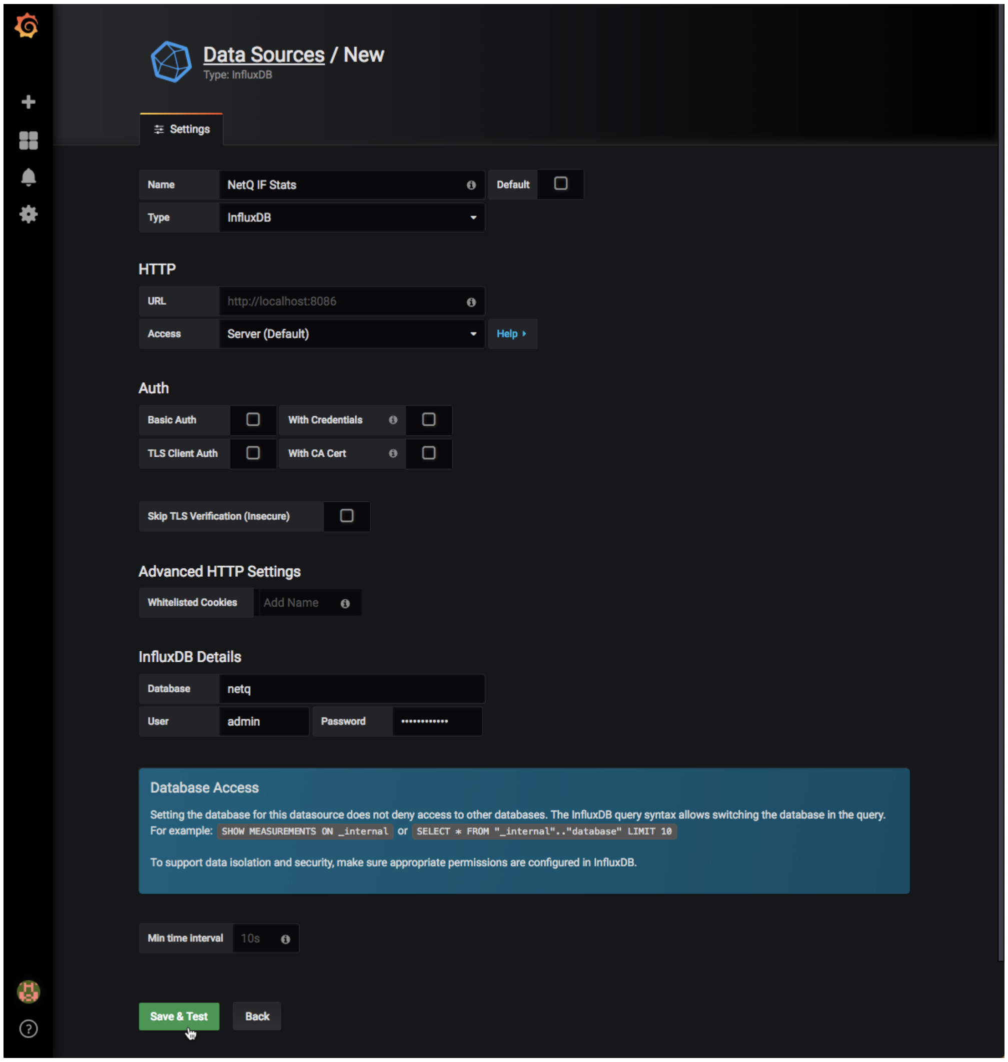
admin

•••••

Grafana
Log In
Forgot your password?
Docs | Support Plans | Community | Grafana v5.1.3 (commit: 087143285)

   The Home Dashboard appears.

6. Create a data source.
   a. Click **Configuration** ( ⚙ ) > **Data Sources**.
   b. Click **Add data source**.
   c. Enter a name for the data source, for example *NetQ IF Stats* or *cumulus-netq*.
   d. Select *InfluxDB* from the **Type** list box.
   e. Verify the URL references port 8086.
   f. In **InfluxDB Details**, enter *netq* for the **Database**.
   g. Enter *admin* for the **User** and *CumulusNetQ!* for the **Password**.



Data Sources / New

Type: InfluxDB

Settings

Name: NetQ IF Stats | Default ☐

Type: InfluxDB

HTTP

URL: http://localhost:8086

Access: Server (Default) | Help ›

Auth

Basic Auth ☐ | With Credentials ☐

TLS Client Auth ☐ | With CA Cert ☐

Skip TLS Verification (Insecure) ☐

Advanced HTTP Settings

Whitelisted Cookies: Add Name

InfluxDB Details

Database: netq

User: admin | Password: ••••••••••••

Database Access

Setting the database for this datasource does not deny access to other databases. The InfluxDB query syntax allows switching the database in the query. For example: `SHOW MEASUREMENTS ON _internal` or `SELECT * FROM "_internal".."database" LIMIT 10`

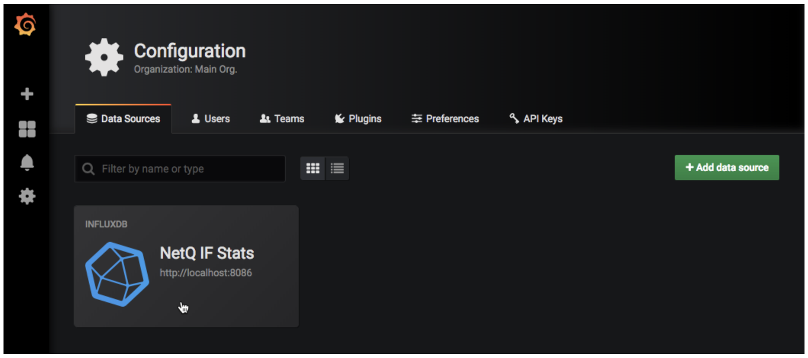
To support data isolation and security, make sure appropriate permissions are configured in InfluxDB.

Min time interval: 10s

Save & Test | Back

h. Click **Save & Test**.
A **Data source is working** confirmation appears when your configuration is good. If a **Network Error: Bad Request(400)** appears, your configuration needs to be modified. Check your configuration and click **Save & Test** again.

i. Click **Data Sources** to view the data source in a card.



Configuration

Organization: Main Org.

INFLUXDB

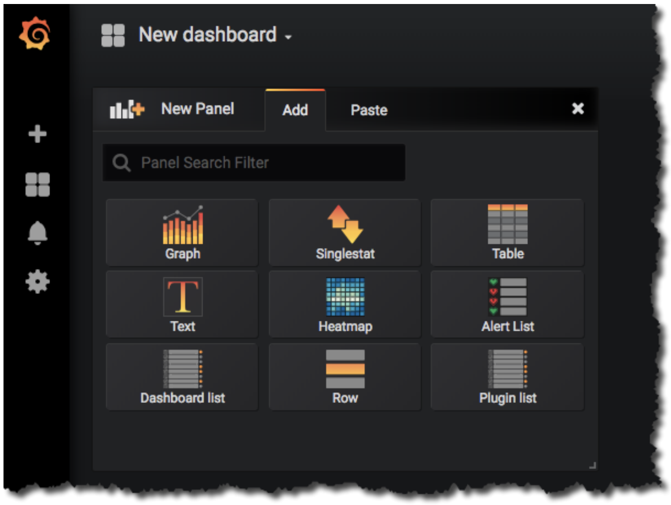
NetQ IF Stats

http://localhost:8086

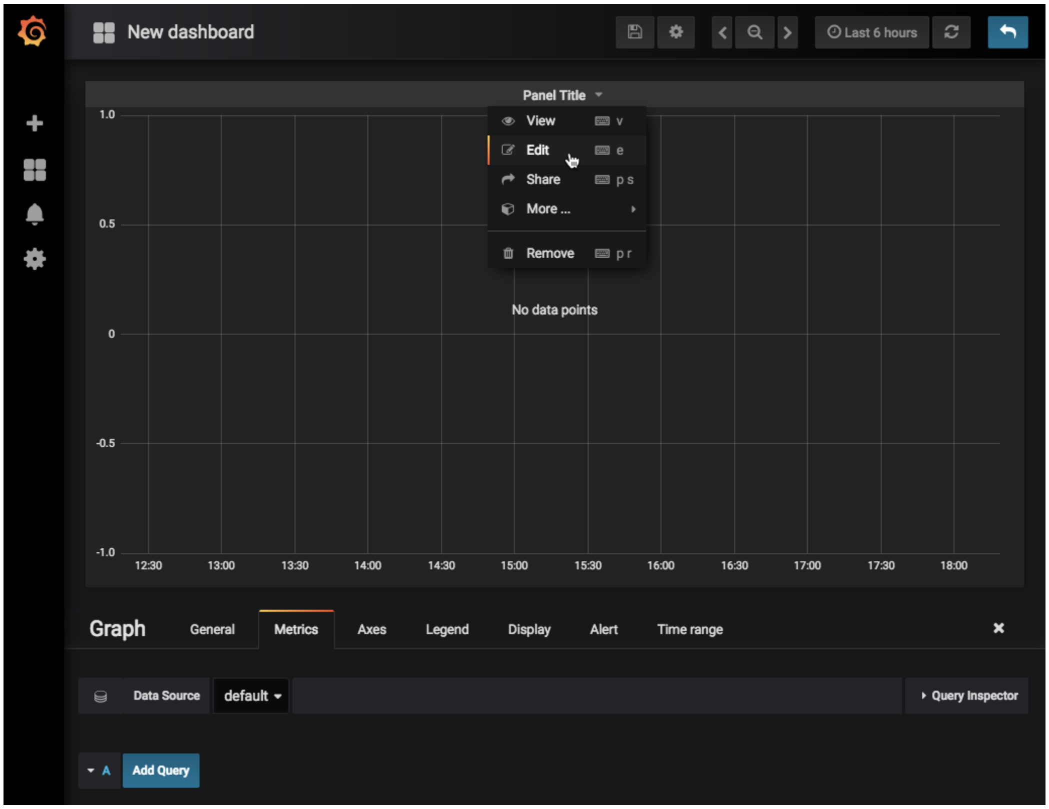
7. Create a Dashboard.

a. Click Create ( + ) > Dashboard.

b. Select a panel type to add to the Dashboard.
You can add as many panels to your dashboard as you want, pick one to start with and then add others as desired. In our example, a Graph panel is selected.



c. Click **Panel Title** > **Edit** to open the edit option tabs.



d. Click through each tab entering the relevant information.

> **Info**: When creating queries in the **Metrics** tab, in FROM select *procnetdev* to access the receive and transmit statistics for display. In WHERE, click ➕, select *hostname* to specify a particular host. In SELECT, choose a statistic to display. And so forth.

   e. When you have added all of the desired panels, click the **New dashboard** title, enter a name for the dashboard, and click **Save**.

   f. Click 💾 to save the Dashboard itself.

You now have a customized view of the NetQ interface statistics. You can add and remove panels at any time.

### *Add Common Interface Statistics to Dashboard*

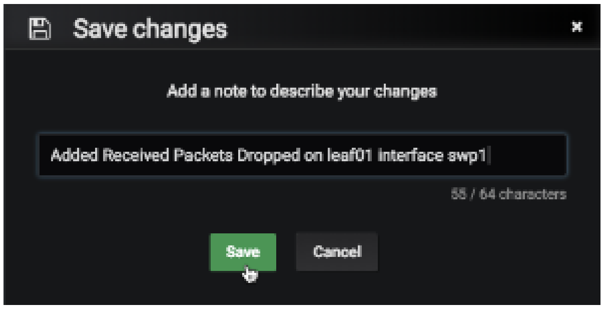
There are many options for displaying the statistics in Grafana. Two examples are provided here.

### *Example: Add Dropped Packets Panels*

When you are monitoring your network, it is useful to see the total number of packets that are being dropped by various interfaces and whether that number is increasing or decreasing. This example creates one panel to display the number of dropped packets on selected leaf01 interfaces and another panel to display the trend for these interfaces.

To add a total number of dropped packets panel:

1. Open Grafana on the Telemetry Server.
2. Open your dashboard (Cumulus Statistics in this example).
3. Click 📊+ to add a new panel.
4. Select **Singlestat**.
5. Click **Panel Title** > **Edit**.
6. On the **Metrics** tab, select the Data Source (*NetQ IF Stats* in this example).
7. Click **Add Query**.
8. Fill out query:
   a. In FROM, select *procnetdev*
   b. In WHERE, click ➕ > select **hostname** > select *leaf01* > click ➕ > select **interface** > select *swp1*
   c. In SELECT, select **rxDrop**
9. On the **General** tab, enter a **Title** for the panel.
10. On the **Options** tab, under **Value** > Stat, select *Total*.
11. Optionally on the **Options** tab, increase the font size and add thresholds.
12. Click ✖ to close the edit options.
13. Click 💾 to save the dashboard.



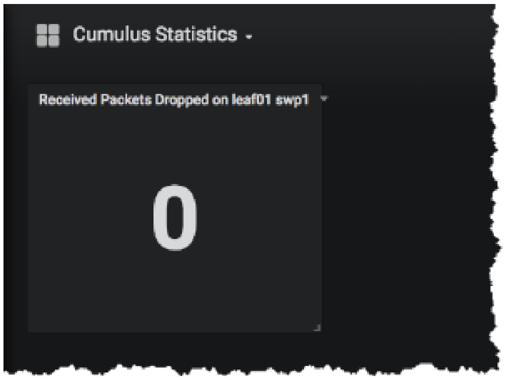
Save changes

Add a note to describe your changes

Added Received Packets Dropped on leaf01 interface swp1

55 / 64 characters

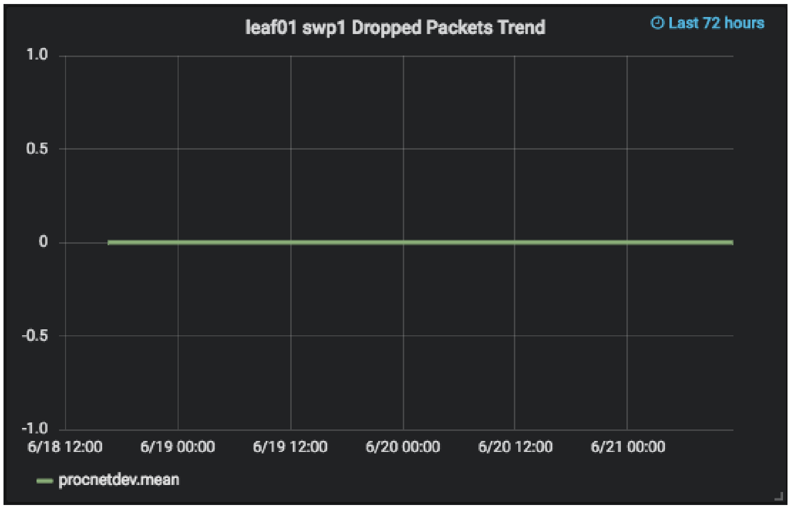
14. Add a comment regarding the change you made, and click **Save**.



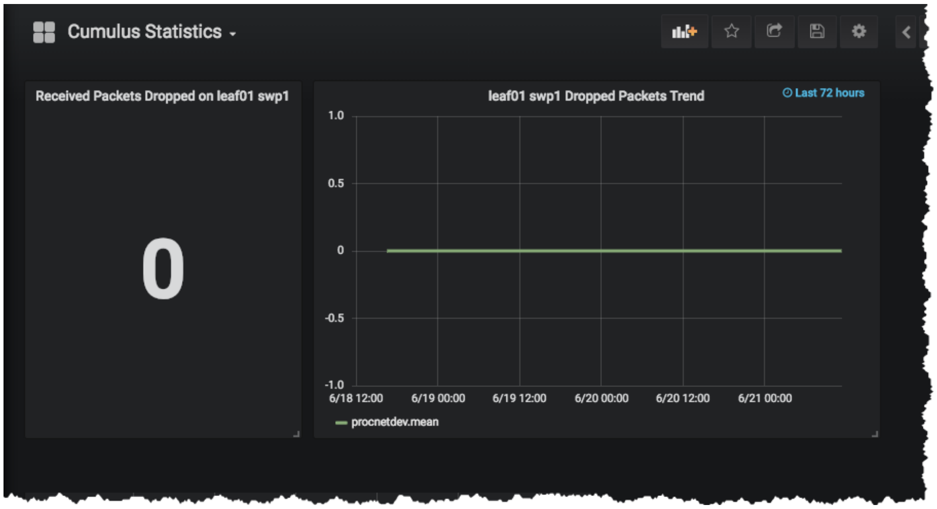
You can now drag and drop the panels to modify their placement, or drag the bottom right corner of a given panel to resize it.

To add a trend view of dropped packets to your dashboard:

1. Open Grafana on the Telemetry Server.
2. Open your dashboard (Cumulus Statistics in this example).
3. Click 📊 to add a new panel.
4. Select **Graph**.
5. Click **Panel Title** > **Edit**.
6. On the **Metrics** tab, select the Data Source (*NetQ IF Stats* in this example).
7. Click **Add Query**.
8. Fill out query:
    a. In FROM, select *procnetdev*
    b. In WHERE, click ➕ > select **hostname** > select *leaf01* > click ➕ > select **interface** > select *swp1*
    c. In SELECT, select **rxDrop**
9. On the **General** tab, enter a **Title** for the panel.
10. Optionally on the **Time range** tab > **Override relative time** > 72h to view rolling results for the last 72 hours.
11. Click ✖ to close the edit options.



12. Click 🖫 to save the dashboard.



### *Example: Add Received Bytes Panel*

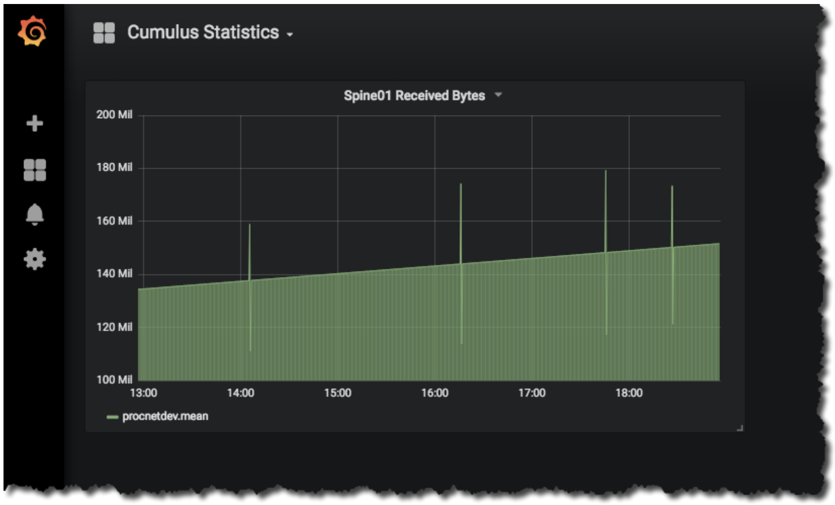
The following example shows the Received Bytes on spine01 for all interfaces over time.

To add trend view of received bytes to the dashboard:

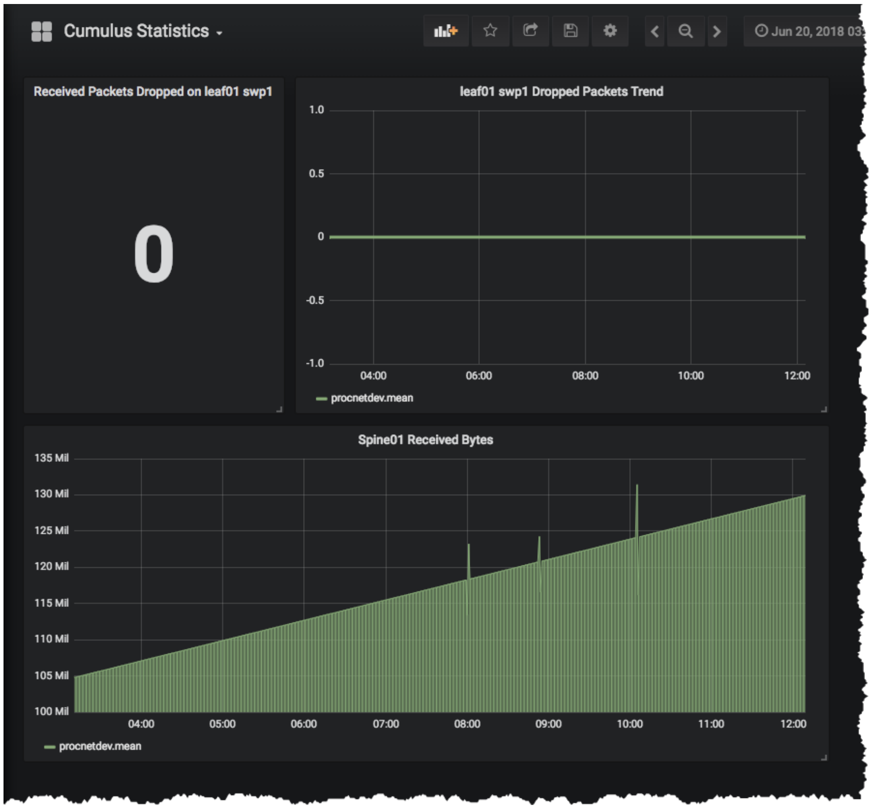
1. Open Grafana on the Telemetry Server.
2. Open your dashboard (Cumulus Statistics in this example).
3. Click ılı+ to add a new panel.
4. Select **Graph**.
5. Click **Panel Title** > **Edit**.
6. On the **Metrics** tab, select the Data Source (*NetQ IF Stats* in this example).
7. Click **Add Query**.
8. Fill out query:
    a. In FROM, select *procnetdev*
    b. In WHERE, click ➕ > select **hostname** > select *spine01*

   c. In SELECT, select **rxBytes**

9. On the **General** tab, enter a **Title** for the panel.
10. Optionally specify other graph criteria using the other tabs.
11. Click ✖ to close the edit options.



12. Click 💾 to save the dashboard.



## Resolve Issues

If you are experiencing issues with the configuration or behavior of the interface statistic collection feature, you can check log files on the Telemetry Server and the individual nodes.

On the Telemetry Server, you can:

- Verify that the InfluxDB is up and running (see example below)
- Review relevant log files
    - `/var/log/cts/cts-influxdb.log`
    - `/var/log/netq-stats-pushd.log`

On each node, check the NetQ Agent log file `/var/log/netq-agent.log`.

### *Example: Verifying InfluxDB Status*

1. Validate the database is up.

```
cumulus@ts:~$ sudo docker ps
CONTAINER
ID     IMAGE           COMMAND                   CREATED       ST
ATUS      PORTS  NAMES
dacbf4f234e9     cumulus-netq    "/tini -g -- /usr/sb…"    2
hours ago   Up 2 hours         netq_netq_1
9cfd1e768be4     redis           "docker-entrypoint.s…"    2
hours ago   Up 2 hours         netq_redis_snt1_1
5ad5e17a0089     redis           "docker-entrypoint.s…"    2
hours ago   Up 2 hours         netq_redis_master_1
9bb544d9bb1b     influxdb        "/entrypoint.sh infl…"    2
hours ago   Up 2 hours         netq_influxdb_1
77a7478bb6dc     cumulus-tsgui   "/portainer"              2
hours ago   Up 2 hours         netq_tsgui_1
```

2. Validate the statistics being collected on which nodes and interfaces.

```
cumulus@ts:~$ sudo docker exec -it 9bb544d9bb1b /bin/bash
bash-4.3# influx -precision rfc3339
Connected to http://localhost:8086 version 1.3.6
InfluxDB shell version: 1.3.6
> SHOW FIELD KEYS ON "netq"
name: procnetdev
fieldKey    fieldType
--------    ---------
rxBytes     integer
rxDrop      integer
rxErrs      integer
rxFrame     integer
rxMulticast integer
rxPackets   integer
txBytes     integer
txCarrier   integer
txColls     integer
txDrop      integer
txErrs      integer
```

```
txPackets   integer

> SHOW SERIES
key
...
downsampled_stats
procnetdev,hostname=spine01,interface=eth0
procnetdev,hostname=spine01,interface=lo
procnetdev,hostname=spine01,interface=mgmt
procnetdev,hostname=spine01,interface=swp1
procnetdev,hostname=spine01,interface=swp2
procnetdev,hostname=spine01,interface=swp3
procnetdev,hostname=spine01,interface=swp31
procnetdev,hostname=spine01,interface=swp32
procnetdev,hostname=spine01,interface=swp4
procnetdev,hostname=spine01,interface=vagrant
procnetdev,hostname=spine02,interface=eth0
procnetdev,hostname=spine02,interface=lo
procnetdev,hostname=spine02,interface=mgmt
procnetdev,hostname=spine02,interface=swp1
procnetdev,hostname=spine02,interface=swp2
procnetdev,hostname=spine02,interface=swp3
procnetdev,hostname=spine02,interface=swp31
procnetdev,hostname=spine02,interface=swp32
procnetdev,hostname=spine02,interface=swp4
procnetdev,hostname=spine02,interface=vagrant
...
```

## Disable Interface Statistics Collection

If you no longer wish to collect interface statistics, you can disable the feature.

Disabling this feature does not purge the data already collected from the database.

To disable interface statistics collection:

1. For each node, disable the feature and restart the NetQ Agent.

```
cumulus@switch:~$ netq config del stats
cumulus@switch:~$ netq config restart agent
```

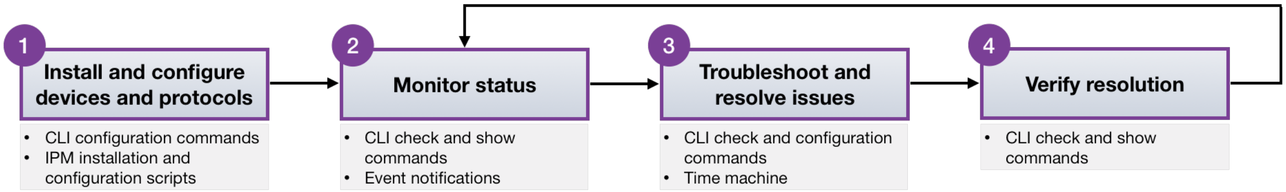
You must restart the NetQ Agent after you disable statistics collection.

2. Once all nodes have stopped pushing statistics, stop and disable the `netq-stats-pushd` service on the Telemetry Server.

```
cumulus@ts:~$ sudo systemctl stop netq-stats-pushd.service
cumulus@ts:~$ sudo systemctl disable netq-stats-pushd.service
```

# Resolve Issues

Monitoring of systems inevitably leads to the need to troubleshoot and resolve the issues found. In fact network management follows a common pattern as shown in this diagram.


1
Install and configure devices and protocols
CLI configuration commands
IPM installation and configuration scripts
2
Monitor status
CLI check and show commands
Event notifications
3
Troubleshoot and resolve issues
CLI check and configuration commands
Time machine
4
Verify resolution
CLI check and show commands

This topic describes some of the tools and commands you can use to troubleshoot issues with the network and NetQ itself.

- Methods for Diagnosing Network Issues (see page 230)
- Resolve MLAG Issues (see page 236)
- Investigate NetQ Issues (see page 244)

## Methods for Diagnosing Network Issues

NetQ provides users with the ability to go back in time to replay the network state, see fabric-wide event change logs and root cause state deviations. The NetQ Telemetry Server maintains data collected by NetQ agents in a time-series database, making fabric-wide events available for analysis. This enables you to replay and analyze network-wide events for better visibility and to correlate patterns. This allows for root-cause analysis and optimization of network configs for the future.

### Contents

This topic describes how to...

- Diagnose an Event after It Occurs (see page 230)
- Use NetQ as a Time Machine (see page 232)
    - How Far Back in Time Can You Travel? (see page 233)
- Trace Paths in a VRF (see page 234)
- Sample Commands for Various Components (see page 235)

### Diagnose an Event after It Occurs

NetQ provides a number of commands for diagnosing past events.

NetQ Notifier records network events and sends them to `syslog`, or another third-party service like PagerDuty or Slack. You can use `netq show changes` to look for any changes made to the runtime configuration that may have triggered the alert, then use `netq trace` to track the connection between the nodes.

The `netq trace` command traces the route of an IP or MAC address from one endpoint to another. It works across bridged, routed and VXLAN connections, computing the path using available data instead of sending real traffic — this way, it can be run from anywhere. It performs MTU and VLAN consistency checks for every link along the path.

For example, say you get an alert about a BGP session failure. You can quickly run `netq check bgp` to determine what sessions failed:

```
cumulus@leaf01:~$ netq check bgp
Total Nodes: 25, Failed Nodes: 4, Total Sessions: 228 , Failed
Sessions: 6,
Node        Neighbor    Peer ID     Reason    Time
----------  ----------  ----------  --------  -------
exit01      swp7.2      spine02     Idle      53m ago
exit01      swp7.3      spine02     Idle      53m ago
exit02      swp6.4      spine01     Idle      53m ago
spine01     swp4.4      exit02      Idle      53m ago
spine02     swp3.2      exit01      Idle      53m ago
spine02     swp3.3      exit01      Idle      53m ago
```

You can run a trace from spine01 to leaf02, which has the IP address 10.1.20.252:

```
cumulus@leaf01:~$ netq trace 10.1.20.252 from spine01 around 5m
spine01 -- spine01:swp1 -- leaf01:vlan20
        -- spine01:swp2 -- leaf02:vlan20
```

Then you can check what's changed on the network to help you identify the problem. Notice the nodes in a *Failed* state filter to the top of the list:

```
cumulus@leaf01:~$ netq show bgp changes
Matching BGP Session records are:
Node             Neighbor                     VRF
ASN        Peer ASN   State  PfxRx        DbState  Last Changed
---------------- ---------------------------- ----------------
---------- ---------- ------ ------------ -------- ------------
leaf04           swp52(spine02)               default
64516      65000      Estd   6            Add      5h ago
leaf03           swp52(spine02)               default
64515      65000      Estd   5            Add      5h ago
leaf01           swp52(spine02)               default
64513      65000      Estd   5            Add      5h ago
leaf02           swp52(spine02)               default
64514      65000      Estd   6            Add      5h ago
spine02          swp2(leaf02)                 default
65000      64514      Estd   2            Add      5h ago
spine02          swp3(leaf03)                 default
65000      64515      Estd   2            Add      5h ago
spine02          swp1(leaf01)                 default
65000      64513      Estd   2            Add      5h ago
spine02          swp4(leaf04)                 default
65000      64516      Estd   2            Add      5h ago
leaf04           swp51(spine01)               default
64516      65000      Estd   6            Add      5h ago
```

```
spine01          swp2(leaf02)                 default
65000      64514      Estd   2            Add      5h ago
leaf02           swp51(spine01)               default
64514      65000      Estd   6            Add      5h ago
leaf01           swp51(spine01)               default
64513      65000      Estd   5            Add      5h ago
spine01          swp1(leaf01)                 default
65000      64513      Estd   2            Add      5h ago
spine01          swp4(leaf04)                 default
65000      64516      Estd   2            Add      5h ago
leaf03           swp51(spine01)               default
64515      65000      Estd   5            Add      5h ago
spine01          swp3(leaf03)                 default
65000      64515      Estd   2            Add      5h ago
```

## Use NetQ as a Time Machine

With NetQ, you can travel back to a specific point in time or a range of times to help you isolate errors and issues.

For example, if you think you had an issue with your sensors last night, you can check the sensors on all your nodes around the time you think the issue occurred:

```
cumulus@leaf01:~$ netq check sensors around 12h
Total Nodes: 25, Failed Nodes: 0, Checked Sensors: 221, Failed Sensors:
0
```

Or you can specify a range of times using the `between` option. The units of time you can specify are second (*s*), minutes (*m*), hours (*h*) and days (*d*). Always specify the most recent time first, then the more distant time. For example, to see the changes made to the network between the past minute and 5 minutes ago, you'd run:

```
cumulus@leaf01:~$ netq show changes between 1m and 5m
No changes to specified interfaces found
No changes to interface addresses found
Matching MAC table records are:
Origin MAC                  VLAN     Node Name        Egress
Port      DbState Last Changed
------ -------------------- -------- ----------------
---------------- ------- --------------
1      44:38:39:00:00:17    20       leaf02           bond-
swp1        Add     3m ago
1      44:38:39:00:00:17    20       leaf01           bond-
swp1        Add     3m ago
```

```
1       44:38:39:00:00:32    20        leaf03            bond-
swp2         Add      4m ago
1       44:38:39:00:00:32    20        leaf04            bond-
swp2         Add      4m ago
1       44:38:39:00:00:15    20        leaf01            bond-
swp2         Del      4m ago
1       44:38:39:00:00:15    20        leaf02            bond-
swp2         Del      4m ago
1       44:38:39:00:00:32    20        leaf03            bond-
swp2         Del      4m ago
1       44:38:39:00:00:32    20        leaf04            bond-
swp2         Del      4m ago
1       44:38:39:00:00:17    20        leaf02            bond-
swp1         Del      4m ago
1       44:38:39:00:00:17    20        leaf01            bond-
swp1         Del      4m ago
Matching IP route records are:
Origin Table            IP
Node             Nexthops                 DbState        Last Changed
------ ---------------- --------------------------------
---------------- ------------------------ -------------- ------------
0      default          ff02::1:ff00:5c/128
spine01          swp1                     Del            3m ago
0      default          ff02::1:ff00:12/128
leaf02           eth0                     Del            3m ago
No changes to IP neighbor table found
No changes to BGP sessions found
No changes to CLAG session found
No changes to LNV session found
```

You can travel back in time 5 minutes and run a trace from spine02 to exit01, which has the IP address 27.0.0.1:

```
cumulus@leaf01:~$ netq trace 27.0.0.1 from spine02 around 5m
Detected Routing Loop. Node exit01 (now via Local Node exit01 and
Ports swp6 <==> Remote  Node/s spine01 and Ports swp3) visited twice.
Detected Routing Loop. Node spine02 (now via mac:00:02:00:00:00:15)
visited twice.
spine02 -- spine02:swp3 -- exit01:swp6.4 -- exit01:swp3 -- exit01
                        -- spine02:swp7  -- spine02
```

### *How Far Back in Time Can You Travel?*

The NetQ Telemetry Server stores an amount of data limited by a few factors:

- The size of the network: The larger the network, the more complex it is because of the number of routes and nodes.
- The amount of memory in the telemetry server. The more memory, the more data you can retrieve. By default, the REDIS memory size is 60% of the virtual RAM. After reaching that size, REDIS does not load any more data.

- The types of nodes you are monitoring with NetQ. You can monitor just network switches, or switches and hosts, or switches, hosts and containers.
- The number of changes in the network over time.

In general, you can expect to be able to query to a point back in time follows:

| Using NetQ to Monitor ... | Data Point | Small Network | Medium Network | Large Network |
|---|---|---|---|---|
| Switches only | Telemetry server memory minimum | 8G | 16G | 24G |
| | Years of data retrievable | 25.5 | 17.4 | 15.6 |
| Switches and Linux hosts | Telemetry server memory minimum | 16G | 32G | 48G |
| | Years of data retrievable | 4.3 | 2.7 | 2.4 |
| Switches, Linux hosts and containers | Telemetry server memory minimum | 32G | 64G | 96G |
| | Years of data retrievable | 2.9 | 1.5 | 1.2 |

The sizing numbers in this table rely on the following assumptions and definitions:

- The types of configuration and operational data being recorded:
    - Switches and hosts: Interfaces; MLAG; LLDP-enabled links; IPv4/v6 addresses, neighbors and routes; BGP sessions; link flaps per day; IPv4/v6 route flaps per day; BGP and MLAG session flaps.
    - Containers: Exposed ports, networks, container flaps per day.
- A small network has 20 racks with 40 leaf nodes, 10 spine nodes and 40 hosts per rack.
- A medium network has 60 racks with 120 leaf nodes, 30 spine nodes and 40 hosts per rack.
- A large network has 100 racks with 200 leaf nodes, 50 spine nodes and 40 hosts per rack.
- The hosts are dual-attached.
- The network is oversubscribed 4:1.
- Adding more memory to the telemetry server allows you to go back even further in time, in a near linear fashion. So doubling the memory should double the range.
- The DB is configured to use up to 70% of the total vRAM allocated to the Telemetry Server.

## Trace Paths in a VRF

The `netq trace` command works with VRFs as well:

```
cumulus@leaf01:~$ netq trace 10.1.20.252 from spine01 vrf default
around 5m
```

```
spine01 -- spine01:swp1 -- leaf01:vlan20
        -- spine01:swp2 -- leaf02:vlan20
```

## Sample Commands for Various Components

NetQ provides network validation for the entire stack, providing algorithmic answers to many questions, both simple and intractable, that pertain to your network fabric.

| Component | Problem | Solution |
|---|---|---|
| Host | Where is this container located?<br>Open ports? What image is being used?<br>Which containers are part of this service? How are they connected? | netq show docker container<br>netq show docker container service |
| Overlay | Is my overlay configured correctly?<br>Can A reach B?<br>Is my control plane configured correctly? | netq check\|show vxlan<br>netq check evpn\|lnv<br>netq trace overlay |
| L3 | Is OSPF working as expected?<br>Is BGP working as expected?<br>Can IP A reach IP B? | netq check\|show ospf<br>netq check\|show bgp<br>netq trace l3 |
| L2 | Is MLAG configured correctly?<br>Is there an STP loop?<br>Is VLAN or MTU misconfigured?<br>How does MAC A reach B? | netq check\|show clag<br>netq show stp<br>netq check\|show vlan<br>netq check mtu<br>netq trace L2 |
| OS | Are all switches licensed correctly?<br>Do all switches have NetQ agents running? | netq check license<br>netq check\|show agents |
| Interfaces | Is my link down? Are all bond links up?<br>What optics am I using? What's the peer for this port?<br>Which ports are empty? Is there a link mismatch? Are links flapping? | netq show\|check interfaces |
| Hardware | Have any components crashed?<br>What switches do I have in the network? | netq check sensors<br>netq show sensors all<br>netq show inventory brief |

# Resolve MLAG Issues

This topic outlines a few scenarios that illustrate how you use NetQ to troubleshoot MLAG on Cumulus Linux switches. Each starts with a log message that indicates the current MLAG state.

NetQ can monitor many aspects of an MLAG configuration, including:

- Verifying the current state of all nodes
- Verifying the dual connectivity state
- Checking that the peer link is part of the bridge
- Verifying whether MLAG bonds are not bridge members
- Verifying whether the VXLAN interface is not a bridge member
- Checking for remote-side service failures caused by `systemctl`
- Checking for VLAN-VNI mapping mismatches
- Checking for layer 3 MTU mismatches on peerlink subinterfaces
- Checking for VXLAN active-active address inconsistencies
- Verifying that STP priorities are the same across both peers

## Contents

This topic describes...

## Scenario: All Nodes Are Up

When the MLAG configuration is running smoothly, NetQ Notifier sends out a message that all nodes are up:

```
2017-05-22T23:13:09.683429+00:00 noc-pr netq-notifier[5501]: INFO:
CLAG: All nodes are up
```

Running `netq show clag` confirms this:

```
cumulus@switch:~$ netq show clag
Matching CLAG session records are:
Node             Peer             SysMac            State Backup
#Bonds #Dual Last Changed
---------------- ---------------- ----------------- ----- ------
------ ----- --------------
```

```
mlx-2700-03      torc-11(P)       44:38:39:ff:ff:01 up    up
8      8     26s ago
noc-pr(P)        noc-se           00:01:01:10:00:01 up    up
9      9     39m ago
noc-se           noc-pr(P)        00:01:01:10:00:01 up    up
9      9     40m ago
torc-11(P)       mlx-2700-03      44:38:39:ff:ff:01 up    up
8      8     27s ago
torc-21(P)       torc-22          44:38:39:ff:ff:02 up    up
8      8     2h ago
torc-22          torc-21(P)       44:38:39:ff:ff:02 up    up
8      8     2h ago
```

You can also verify a specific node is up:

```
cumulus@switch:~$ netq mlx-2700-03 show clag
Matching CLAG session records are:
Node             Peer             SysMac            State Backup
#Bonds #Dual Last Changed
---------------- ---------------- ----------------- ----- ------
------ ----- --------------
mlx-2700-03      torc-11(P)       44:38:39:ff:ff:01 up    up
8      8     45s ago
```

Similarly, checking the MLAG state with NetQ also confirms this:

```
cumulus@switch:~$ netq check clag
Checked Nodes: 6, Failed Nodes: 0
```

When you are logged directly into a switch, you can run `clagctl` to get the state:

```
cumulus@mlx-2700-03:/var/log# sudo clagctl

The peer is alive
Peer Priority, ID, and Role: 4096 00:02:00:00:00:4e primary
Our Priority, ID, and Role: 8192 44:38:39:00:a5:38 secondary
Peer Interface and IP: peerlink-3.4094 169.254.0.9
VxLAN Anycast IP: 36.0.0.20
Backup IP: 27.0.0.20 (active)
System MAC: 44:38:39:ff:ff:01

CLAG Interfaces
Our Interface    Peer Interface   CLAG Id Conflicts            Proto-
Down Reason
---------------- ---------------- ------- --------------------
-----------------
vx-38            vx-38            -       -                    -
```

```
vx-33            vx-33            -       -                    -
hostbond4        hostbond4        1       -                    -
hostbond5        hostbond5        2       -                    -
vx-37            vx-37            -       -                    -
vx-36            vx-36            -       -                    -
vx-35            vx-35            -       -                    -
vx-34            vx-34            -       -                    -
```

## Scenario: Dual-connected Bond Is Down

When dual connectivity is lost in an MLAG configuration, you receive messages from NetQ Notifier similar to the following:

```
2017-05-22T23:14:40.290918+00:00 noc-pr netq-notifier[5501]: WARNING:
LINK: 1 link(s) are down. They are: mlx-2700-03 hostbond5
2017-05-22T23:14:53.081480+00:00 noc-pr netq-notifier[5501]: WARNING:
CLAG: 1 node(s) have failures. They are: mlx-2700-03
2017-05-22T23:14:58.161267+00:00 noc-pr netq-notifier[5501]: WARNING:
CLAG: 2 node(s) have failures. They are: mlx-2700-03, torc-11
```

To begin your investigation, show the status of the `clagd` service:

```
cumulus@switch:~$ netq mlx-2700-03 show services clagd

Matching services records are:
Hostname    Service   PID   VRF     Enabled   Active    Monitored
Status   Up Time   Last Changed
----------- --------- ----- ------- --------- -------- -----------
-------- --------- --------------
mlx-2700-03 clagd     5802  default yes       yes       yes
warning  1h ago    2m ago
```

Checking the MLAG status provides the reason for the failure:

```
cumulus@switch:~$ netq check clag
Checked Nodes: 6, Warning Nodes: 2
Node             Reason
----------------
----------------------------------------------------------------------
----
mlx-2700-03      Link Down: hostbond5
torc-11          Singly Attached Bonds: hostbond5
```

You can retrieve the output in JSON format for export to another tool:

```
cumulus@switch:~$ netq check clag json
{
"warningNodes": [
{ "node": "mlx-2700-03", "reason": "Link Down: hostbond5" }
,
{ "node": "torc-11", "reason": "Singly Attached Bonds: hostbond5" }
],
"failedNodes": [],
"summary":
{ "checkedNodeCount": 6, "failedNodeCount": 0, "warningNodeCount": 2 }
}
```

After you fix the issue, you can show the MLAG state to see if all the nodes are up. The notifications from NetQ Notifier indicate all nodes are UP, and the `netq check` flag also indicates there are no failures.

```
cumulus@switch:~$ netq show clag
Matching CLAG session records are:
Node             Peer             SysMac            State Backup
#Bonds #Dual Last Changed
---------------- ---------------- ----------------- ----- ------
------ ----- --------------
mlx-2700-03      torc-11(P)       44:38:39:ff:ff:01 up    up
8      7     52s ago
noc-pr(P)        noc-se           00:01:01:10:00:01 up    up
9      9     27m ago
noc-se           noc-pr(P)        00:01:01:10:00:01 up    up
9      9     27m ago
torc-11(P)       mlx-2700-03      44:38:39:ff:ff:01 up    up
8      7     50s ago
torc-21(P)       torc-22          44:38:39:ff:ff:02 up    up
8      8     1h ago
torc-22          torc-21(P)       44:38:39:ff:ff:02 up    up
8      8     1h ago
```

When you are logged directly into a switch, you can run `clagctl` to get the state:

```
cumulus@mlx-2700-03:/var/log# sudo clagctl

The peer is alive
Peer Priority, ID, and Role: 4096 00:02:00:00:00:4e primary
Our Priority, ID, and Role: 8192 44:38:39:00:a5:38 secondary
Peer Interface and IP: peerlink-3.4094 169.254.0.9
VxLAN Anycast IP: 36.0.0.20
Backup IP: 27.0.0.20 (active)
System MAC: 44:38:39:ff:ff:01

CLAG Interfaces
Our Interface    Peer Interface   CLAG Id Conflicts            Proto-
Down Reason
```

```
---------------- ---------------- ------- --------------------
-----------------
vx-38            vx-38            -       -                    -
vx-33            vx-33            -       -                    -
hostbond4        hostbond4        1       -                    -
hostbond5        -                2       -                    -
vx-37            vx-37            -       -                    -
vx-36            vx-36            -       -                    -
vx-35            vx-35            -       -                    -
vx-34            vx-34            -       -                    -
```

## Scenario: VXLAN Active-active Device or Interface Is Down

When a VXLAN active-active device or interface in an MLAG configuration is down, log messages also include VXLAN and LNV checks.

```
2017-05-22T23:16:51.517522+00:00 noc-pr netq-notifier[5501]: WARNING:
VXLAN: 2 node(s) have failures. They are: mlx-2700-03, torc-11
2017-05-22T23:16:51.525403+00:00 noc-pr netq-notifier[5501]: WARNING:
LINK: 2 link(s) are down. They are: torc-11 vx-37, mlx-2700-03 vx-37
2017-05-22T23:16:54.194681+00:00 noc-pr netq-notifier[5501]: WARNING:
LNV: 1 node(s) have failures. They are: torc-22
2017-05-22T23:16:59.448755+00:00 noc-pr netq-notifier[5501]: WARNING:
LNV: 3 node(s) have failures. They are: tor-2, torc-21, torc-22
2017-05-22T23:17:04.703044+00:00 noc-pr netq-notifier[5501]: WARNING:
CLAG: 2 node(s) have failures. They are: mlx-2700-03, torc-11
```

To begin your investigation, show the status of the `clagd` service:

```
cumulus@switch:~$ netq mlx-2700-03 show service clagd
Matching services records are:
Node        Service   PID   VRF     Enabled   Active   Monitored
Status   Up Time   Last Changed
----------- --------- ----- ------- --------- -------- -----------
-------- --------- --------------
mlx-2700-03 clagd     5802  default yes       yes      yes
error    2h ago    3m ago
```

Checking the MLAG status provides the reason for the failure:

```
cumulus@switch:~$ netq check clag
Checked Nodes: 6, Warning Nodes: 2, Failed Nodes: 2
Node             Reason
----------------
----------------------------------------------------------------------
----
mlx-2700-03      Protodown Bonds: vx-37:vxlan-single
```

```
torc-11          Protodown Bonds: vx-37:vxlan-single
```

You can retrieve the output in JSON format for export to another tool:

```
cumulus@switch:~$ netq check clag json
{
"failedNodes": [
{ "node": "mlx-2700-03", "reason": "Protodown Bonds: vx-37:vxlan-
single" }
,
{ "node": "torc-11", "reason": "Protodown Bonds: vx-37:vxlan-single" }
],
"summary":
{ "checkedNodeCount": 6, "failedNodeCount": 2, "warningNodeCount": 2 }
}
```

After you fix the issue, you can show the MLAG state to see if all the nodes are up:

```
cumulus@switch:~$ netq show clag
Matching CLAG session records are:
Node             Peer             SysMac            State Backup
#Bonds #Dual Last Changed
---------------- ---------------- ----------------- ----- ------
------ ----- --------------
mlx-2700-03      torc-11(P)       44:38:39:ff:ff:01 up    up
8      7     52s ago
noc-pr(P)        noc-se           00:01:01:10:00:01 up    up
9      9     27m ago
noc-se           noc-pr(P)        00:01:01:10:00:01 up    up
9      9     27m ago
torc-11(P)       mlx-2700-03      44:38:39:ff:ff:01 up    up
8      7     50s ago
torc-21(P)       torc-22          44:38:39:ff:ff:02 up    up
8      8     1h ago
torc-22          torc-21(P)       44:38:39:ff:ff:02 up    up
8      8     1h ago
```

When you are logged directly into a switch, you can run `clagctl` to get the state:

```
cumulus@mlx-2700-03:/var/log# sudo clagctl

The peer is alive
Peer Priority, ID, and Role: 4096 00:02:00:00:00:4e primary
Our Priority, ID, and Role: 8192 44:38:39:00:a5:38 secondary
Peer Interface and IP: peerlink-3.4094 169.254.0.9
VxLAN Anycast IP: 36.0.0.20
Backup IP: 27.0.0.20 (active)
```

```
System MAC: 44:38:39:ff:ff:01

CLAG Interfaces
Our Interface    Peer Interface   CLAG Id Conflicts            Proto-
Down Reason
---------------- ---------------- ------- --------------------
-----------------
vx-38            vx-38            -       -                    -
vx-33            vx-33            -       -                    -
hostbond4        hostbond4        1       -                    -
hostbond5        hostbond5        2       -                    -
vx-37            -                -       -                    vxlan-
single
vx-36            vx-36            -       -                    -
vx-35            vx-35            -       -                    -
vx-34            vx-34            -       -                    -
```

## Scenario: Remote-side clagd Stopped by systemctl Command

In the event the `clagd` service is stopped via the `systemctl` command, NetQ Notifier sends messages similar to the following:

```
2017-05-22T23:51:19.539033+00:00 noc-pr netq-notifier[5501]: WARNING:
VXLAN: 1 node(s) have failures. They are: torc-11
2017-05-22T23:51:19.622379+00:00 noc-pr netq-notifier[5501]: WARNING:
LINK: 2 link(s) flapped and are down. They are: torc-11 hostbond5,
torc-11 hostbond4
2017-05-22T23:51:19.622922+00:00 noc-pr netq-notifier[5501]: WARNING:
LINK: 23 link(s) are down. They are: torc-11 VlanA-1-104-v0, torc-11
VlanA-1-101-v0, torc-11 VlanA-1, torc-11 vx-33, torc-11 vx-36, torc-
11 vx-37, torc-11 vx-34, torc-11 vx-35, torc-11 swp7, torc-11 VlanA-1-
102-v0, torc-11 VlanA-1-103-v0, torc-11 VlanA-1-100-v0, torc-11 VlanA-
1-106-v0, torc-11 swp8, torc-11 VlanA-1.106, torc-11 VlanA-1.105,
torc-11 VlanA-1.104, torc-11 VlanA-1.103, torc-11 VlanA-1.102, torc-
11 VlanA-1.101, torc-11 VlanA-1.100, torc-11 VlanA-1-105-v0, torc-11
vx-38
2017-05-22T23:51:27.696572+00:00 noc-pr netq-notifier[5501]: INFO:
LINK: 15 link(s) are up. They are: torc-11 VlanA-1.106, torc-11 VlanA-
1-104-v0, torc-11 VlanA-1.104, torc-11 VlanA-1.103, torc-11 VlanA-
1.101, torc-11 VlanA-1-100-v0, torc-11 VlanA-1.100, torc-11 VlanA-
1.102, torc-11 VlanA-1-101-v0, torc-11 VlanA-1-102-v0, torc-11 VlanA-
1.105, torc-11 VlanA-1-103-v0, torc-11 VlanA-1-106-v0, torc-11 VlanA-
1, torc-11 VlanA-1-105-v0
2017-05-22T23:51:30.863789+00:00 noc-pr netq-notifier[5501]: WARNING:
LNV: 1 node(s) have failures. They are: torc-11
2017-05-22T23:51:36.156708+00:00 noc-pr netq-notifier[5501]: WARNING:
CLAG: 2 node(s) have failures. They are: mlx-2700-03, torc-11
2017-05-22T23:51:36.183638+00:00 noc-pr netq-notifier[5501]: WARNING:
LNV: 2 node(s) have failures. They are: spine-2, torc-11
```

```
2017-05-22T23:51:41.444670+00:00 noc-pr netq-notifier[5501]: WARNING:
LNV: 1 node(s) have failures. They are: torc-11
```

Showing the MLAG state reveals which nodes are down:

```
cumulus@switch:~$ netq show clag
Matching CLAG session records are:
Node             Peer             SysMac            State Backup
#Bonds #Dual Last Changed
---------------- ---------------- ----------------- ----- ------
------ ----- --------------
mlx-2700-03                       44:38:39:ff:ff:01 down  down
8      0     33s ago
noc-pr(P)        noc-se           00:01:01:10:00:01 up    up
9      9     1h ago
noc-se           noc-pr(P)        00:01:01:10:00:01 up    up
9      9     1h ago
torc-11                           44:38:39:ff:ff:01 down  n/a
0      0     32s ago
torc-21(P)       torc-22          44:38:39:ff:ff:02 up    up
8      8     2h ago
torc-22          torc-21(P)       44:38:39:ff:ff:02 up    up
8      8     2h ago
```

Checking the MLAG status provides the reason for the failure:

```
cumulus@switch:~$ netq check clag
Checked Nodes: 6, Warning Nodes: 1, Failed Nodes: 2
Node             Reason
----------------
----------------------------------------------------------------------
----
mlx-2700-03      Peer Connectivity failed
torc-11          Peer Connectivity failed
```

You can retrieve the output in JSON format for export to another tool:

```
cumulus@switch:~$ netq check clag json
{
"failedNodes": [
{ "node": "mlx-2700-03", "reason": "Peer Connectivity failed" }
,
{ "node": "torc-11", "reason": "Peer Connectivity failed" }
],
"summary":
{ "checkedNodeCount": 6, "failedNodeCount": 2, "warningNodeCount": 1 }
}
```

When you are logged directly into a switch, you can run `clagctl` to get the state:

```
root@mlx-2700-03:/var/log# clagctl

The peer is not alive
Our Priority, ID, and Role: 8192 44:38:39:00:a5:38 primary
Peer Interface and IP: peerlink-3.4094 169.254.0.9
VxLAN Anycast IP: 36.0.0.20
Backup IP: 27.0.0.20 (inactive)
System MAC: 44:38:39:ff:ff:01

CLAG Interfaces
Our Interface    Peer Interface   CLAG Id Conflicts            Proto-
Down Reason
---------------- ---------------- ------- --------------------
-----------------
vx-38            -                -       -                    -
vx-33            -                -       -                    -
hostbond4        -                1       -                    -
hostbond5        -                2       -                    -
vx-37            -                -       -                    -
vx-36            -                -       -                    -
vx-35            -                -       -                    -
vx-34            -                -       -                    -
```

# Investigate NetQ Issues

There are several tacks you can take to locate and investigate issues that occur in the NeQ software itself, including viewing configuration and log files, verifying NetQ Agent health, and verifying Telemetry Server configuration. If these do not produce a resolution, you can capture a log to use in discussion with Cumulus Networks support team.

## Contents

This topic describes how to...

## Browse Configuration and Log Files

To aid in troubleshooting issues with NetQ, there are several configuration and log files on the **telemetry server** that can provide insight into the root cause of the issue:

| File | Description |
|---|---|
| `/etc/netq/netq.yml` | The NetQ Telemetry Server configuration file. |
| `/var/log/cts/cts-backup.log` | Database service backup log file. |
| `/var/log/cts/cts-redis.log` | The Redis log file. |
| `/var/log/cts/cts-sentinel.log` | The Redis sentinel log file. |
| `/var/log/cts/cts-dockerd.log` | The Docker daemon log file. |
| `/var/log/cts/cts-docker-compose.log` | The backup log file. |
| `/var/log/netqd.log` | The NetQ daemon log file for the NetQ CLI. |
| `/var/log/netq-notifier.log` | The NetQ Notifier log file. |
| `/etc/cts/netq/netq.yml` | The configuration file for NetQ running in the web-browser. |
| `/etc/cts/run/redis/redis_6379.conf` | The runtime configuration file for the REDIS database. |
| `/etc/cts/run/redis/snt1.conf` | The runtime configuration file for REDIS Sentinels. |
| `/etc/cts/redis/redis.conf` | Contains the base REDIS configuration, which is inherited by and overriden by the `/etc/cts/run/redis/redis_6379.conf` file. |

| File | Description |
|---|---|
| `/etc/cts /environment` | The configuration file for environment variables that configure and control NetQ Telemetry Server services. This file contains the `REDIS_MEMORY_PCT` environment variable. Setting this variable to a value between 10-90 allocates that much of the VM's total memory to REDIS. The default value is 60%. |

A **node** running the NetQ Agent has the following configuration and log files:

| File | Description |
|---|---|
| `/etc/netq/netq.yml` | The NetQ configuration file. |
| `/var/log/netq-agent.log` | The NetQ Agent log file. |
| `/etc/netq/config.d/netq-agent-commands.yml` | Contains key-value command pairs and relevant custom configuration settings. |
| `/run/netq-agent-running.json` | Contains the full command list that will be pushed when the agent starts. |

## Check NetQ Agent Health

Checking the health of the NetQ agents is a good way to start troubleshooting NetQ on your network. If any agents are rotten, meaning three heartbeats in a row were not sent, then you can investigate the rotten node. In the example below, the NetQ Agent on server01 is rotten, so you know where to start looking for problems:

```
netq@446c0319c06a:/$ netq check agents
Checked nodes: 12,

Rotten nodes: 1
netq@446c0319c06a:/$ netq show agents
Node      Status    Sys Uptime    Agent Uptime
--------  --------  ------------  --------------
exit01
Fresh
     8h ago        4h ago
exit02
Fresh
     8h ago        4h ago
leaf01
Fresh
```

```
     8h ago        4h ago
leaf02
Fresh
     8h ago        4h ago
leaf03
Fresh
     8h ago        4h ago
leaf04
Fresh
     8h ago        4h ago
server01
Rotten
    4h ago        4h ago
server02
Fresh
     4h ago        4h ago
server03
Fresh
     4h ago        4h ago
server04
Fresh
     4h ago        4h ago
spine01
Fresh
     8h ago        4h ago
spine02
Fresh
     8h ago        4h ago
```

## Verify Telemetry Server Configuration on a Node

If you get an error when your run the `netq config add server` command on a node, it is usually due to one of two reasons:

- The hostname or IP address for the telemetry server was input incorrectly when you ran `netq config add server`. Check what you input and try again.
- The Telemetry Server is not responding. Try pinging the IP address you entered and see if the ping works.

## Generate a Support File

The `cts-support` command generates an archive of useful information for troubleshooting issues with NetQ. It is an extension of the `cl-support` command in Cumulus Linux. It provides information about the telemetry server configuration and runtime statistics as well as output from the `docker ps` command. The

Cumulus Networks support team may request the output of this command when assisting with any issues that you could not solve with your own troubleshooting. Run the following command on the Telemetry Server:

```
cumulus@ts:~$ cts-support
```