Cumulus NetQ 2.1

CLI 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 2.0 offers the ability to easily monitor and manage your data center network infrastructure and operational health. This guide provides instructions and information about monitoring individual components of the network, the network as a whole, and the NetQ software itself using the NetQ command line interface (CLI). If you prefer to use a graphical interface, refer to the Cumulus NetQ UI User Guide.

CLI 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 2.1

Cumulus NetQ has been reinvented to scale with the fast adoption rate of open networks and rapid network growth. In addition to the real-time data collection and fabric-wide performance analysis that are already supported in Cumulus NetQ 1.4.1, Cumulus NetQ now offers a graphical user interface for:

- Data visualizations of the overlay and underlay networks,
- Simplified network troubleshooting with network-wide roll-ups of health and alarm status in a single card, and
- Proactive validation of the network status and configuration to regularly detect network issues.

For further information regarding new features, improvements, 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.

The following NetQ documents are available:

- Cumulus NetQ Deployment Guide
- Cumulus NetQ CLI User Guide (this guide)
- Cumulus NetQ UI User Guide
- Cumulus NetQ Release Notes
- What the NetQ Validation System Checks

- Cumulus NetQ Release Versioning and Support Policy

Document Formatting

This 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 (Version 2.1.1) topic). Simply log in to any network node to access the command line.

To access the CLI from a switch or server:

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

```
<computer>:~<username>$ ssh cumulus@switch
```

2. Enter your password to reach the command prompt. The default password is *CumulusLinux!* For example:

```
Enter passphrase for key '/Users/<username>/.ssh/id_rsa': <enter
CumulusLinux! here>
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: Tue Feb 11 09:28:12 2019 from 10.0.0.14
cumulus@switch:~$
```

3. Run commands. For example:

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

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 15)

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`. Cumulus NetQ commands fall into one of four syntax categories: validation (check), monitoring (show), configuration, and trace :

```
netq check <network-protocol-or-service> [options]
netq show <network-protocol-or-service> [options]
netq config <action> <object> [options]
netq trace <destination> from <source> [options]
```

Symbols	Meaning
Parentheses ( )	Grouping of required parameters. Choose one.
Square brackets [ ]	Single or group of optional parameters. If more than one object or keyword is available, choose one.
Angle brackets < >	Required variable. Value for a keyword or option; enter according to your deployment nomenclature.
Pipe |	Separates object and keyword options, also separates value options; enter one object or keyword and zero or one value.

For example, in the `netq check` command:

- [<hostname>] is an optional parameter with a variable value named *hostname*
- <network-protocol-or-service> represents a number of possible key words, such as *agents, bgp, evpn,* 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 leaf02 check agents json`
- `netq show bgp`
- `netq config restart cli`
- `netq trace 10.0.0.5 from 10.0.0.35`

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 using the **`around`** 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.

Command Prompts

NetQ code examples use the following prompts:

- `cumulus@switch:~$` Indicates the user *cumulus* is logged in to a switch to run the example command
- `cumulus@host:~$` Indicates the user *cumulus* is logged in to a host to run the example command

The switches must be running the Cumulus Linux operating system (OS), NetQ Platform software, and the NetQ Agent. The hosts must be running CentOS, RHEL, or Ubuntu OS and the NetQ Agent. Refer to the Install NetQ (Version 2.1.1) 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@switch:~$ 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 no additional required parameters and two optional parameters, *vrf* and *around*, that can be used with a BGP check.

```
cumulus@switch:~$ netq check bgp help
Commands:
   netq check bgp [vrf <vrf>] [around <text-time>] [json]
```

```
cumulus@switch:~$
```

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:

- Validation Commands (see page)
- Monitoring Commands (see page)
- Configuration Commands (see page)
- Trace Commands (see page)

Validation Commands

The `netq check` commands enable the network administrator to validate the current or historical state of the network by looking for errors and misconfigurations in the network. The commands run fabric-wide validations against various configured protocols and services to determine how well the network is operating. 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 (maximum transmission unit) consistency across paths
- 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 commands take the form of `netq check <network-protocol-or-service> [options]`, where the options vary according to the protocol or service.

This example shows the output for the `netq check bgp` command, followed by the same command using the *json* option. If there had been any failures, they would be have been listed below the summary results or in the *failedNodes* section, respectively.

```
cumulus@switch:~$ netq check bgp
Total Nodes: 8, Failed Nodes: 0, Total Sessions: 30, Failed Sessions:
0

cumulus@switch:~$ netq check bgp json
{
    "failedNodes":[
    ],
    "summary":{
        "checkedNodeCount":8,
        "failedSessionCount":0,
        "failedNodeCount":0,
        "totalSessionCount":30
    }
}
```

Monitoring Commands

The netq `show` commands enable the network administrator to view details about the current or historical configuration and status of the various protocols or services. 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
- clag: CLAG status
- events: Display changes over time
- 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
- notification: Slack or PagerDuty notification configurations
- 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>] show <network-protocol-or-service> [options]`, where the options vary according to the protocol or service. The commands can be restricted from showing the information for *all* devices to showing information for a selected device using the *hostname* variable.

This example shows the standard and restricted output for the `netq show agents` command.

```
cumulus@switch:~$ netq show agents
Matching agents records:
Hostname          Status           NTP Sync
Version                              Sys Uptime                Agent
Uptime              Reinitialize Time          Last Changed
----------------- --------------- --------
------------------------------------ -------------------------
------------------------- --------------------------
-------------------------
edge01            Fresh            yes      2.1.0-ub16.
04u15~1555612152.6e34b56  2d:2h:48m:43s             2d:2h:48m:
36s             2d:2h:48m:36s              Sun Apr 21 16:00:50 2019
exit01            Fresh            yes      2.1.0-cl3u15~1555612272.
6e34b56      2d:2h:48m:1s              2d:2h:47m:53s             2d:
2h:47m:53s              Sun Apr 21 16:00:52 2019
exit02            Fresh            yes      2.1.0-cl3u15~1555612272.
6e34b56      2d:2h:48m:7s              2d:2h:47m:58s             2d:
2h:47m:58s              Sun Apr 21 16:01:19 2019
leaf01            Fresh            yes      2.1.0-cl3u15~1555612272.
6e34b56      2d:2h:47m:59s             2d:2h:47m:51s             2d:
2h:47m:51s              Sun Apr 21 16:00:59 2019
leaf02            Fresh            yes      2.1.0-cl3u15~1555612272.
6e34b56      2d:2h:48m:9s              2d:2h:48m:0s              2d:
2h:48m:0s               Sun Apr 21 16:01:43 2019
leaf03            Fresh            yes      2.1.0-cl3u15~1555612272.
6e34b56      2d:2h:48m:8s              2d:2h:47m:59s             2d:
2h:47m:59s              Sun Apr 21 16:01:23 2019
leaf04            Fresh            yes      2.1.0-cl3u15~1555612272.
6e34b56      2d:2h:48m:10s             2d:2h:48m:2s              2d:
2h:48m:2s               Sun Apr 21 16:01:27 2019
server01          Fresh            yes      2.1.0-ub16.
04u15~1555612152.6e34b56  2d:2h:46m:6s              2d:2h:45m:
58s             2d:2h:45m:58s              Sun Apr 21 16:00:43 2019
server02          Fresh            yes      2.1.0-ub16.
04u15~1555612152.6e34b56  2d:2h:46m:5s              2d:2h:45m:
57s             2d:2h:45m:57s              Sun Apr 21 16:00:46 2019
server03          Fresh            yes      2.1.0-ub16.
04u15~1555612152.6e34b56  2d:2h:46m:5s              2d:2h:45m:
57s             2d:2h:45m:57s              Sun Apr 21 16:00:52 2019
```

```
server04          Fresh            yes      2.1.0-ub16.
04u15~1555612152.6e34b56  2d:2h:46m:5s              2d:2h:45m:
57s             2d:2h:45m:57s             Sun Apr 21 16:00:43 2019
spine01           Fresh            yes      2.1.0-cl3u15~1555612272.
6e34b56      2d:2h:48m:11s             2d:2h:48m:3s              2d:
2h:48m:3s               Sun Apr 21 16:01:33 2019
spine02           Fresh            yes      2.1.0-cl3u15~1555612272.
6e34b56      2d:2h:48m:5s              2d:2h:47m:57s             2d:
2h:47m:57s              Sun Apr 21 16:01:12 2019

cumulus@switch:~$ netq show agents json
{
    "agents":[
        {
            "status":"Fresh",
            "lastChanged":1555862450.0,
            "reinitializeTime":1555689453.0,
            "hostname":"edge01",
            "version":"2.1.0-ub16.04u15~1555612152.6e34b56",
            "sysUptime":1555689446.0,
            "ntpSync":"yes",
            "agentUptime":1555689453.0
        },
        {
            "status":"Fresh",
            "lastChanged":1555862452.0,
            "reinitializeTime":1555689496.0,
            "hostname":"exit01",
            "version":"2.1.0-cl3u15~1555612272.6e34b56",
            "sysUptime":1555689488.0,
            "ntpSync":"yes",
            "agentUptime":1555689496.0
        },
        {
            "status":"Fresh",
            "lastChanged":1555862479.0,
            "reinitializeTime":1555689491.0,
            "hostname":"exit02",
            "version":"2.1.0-cl3u15~1555612272.6e34b56",
            "sysUptime":1555689482.0,
            "ntpSync":"yes",
            "agentUptime":1555689491.0
        },
...

cumulus@switch:~$ netq leaf01 show agents
Matching agents records:
Hostname          Status           NTP Sync
Version                              Sys Uptime                Agent
Uptime              Reinitialize Time          Last Changed
----------------- ---------------- --------
------------------------------------ -------------------------
```

```
------------------------- -------------------------
-------------------------
leaf01            Fresh            yes      2.1.0-cl3u15~1555612272.
6e34b56      2d:2h:49m:59s             2d:2h:49m:51s            2d:
2h:49m:51s              Sun Apr 21 16:00:59 2019
```

Configuration Commands

The `netq config` and `netq notification` commands enable the network administrator to manage NetQ Agent and CLI server configuration, set up container monitoring, and event notification.

NetQ Agent and CLI Server

The agent commands enable the network administrator to configure individual NetQ Agents. Refer to Cumulus NetQ Primer for a description of NetQ Agents and to Manage NetQ Agents (Version 2.1.1) for more detailed usage examples.

The agent configuration commands enable you to add and remove agents from switches and hosts, start and stop agent operations, add and remove Kubernetes container monitoring, add or remove sensors, debug the agent, and add or remove FRR (FRRouting).

> 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 (add|del|show) agent
netq config (start|stop|status|restart) agent
```

This example shows how to configure the agent to send sensor data.

```
cumulus@switch~:$ netq config add agent sensors
```

This example shows how to start monitoring with Kubernetes.

```
cumulus@switch:~$ netq config add agent kubernetes-monitor poll-
period 15
```

This example show how to view the NetQ Agent configuration.

```
cumulus@switch:~$ netq config show agent
netq-agent             value      default
---------------------  ---------  ---------
enable-opta-discovery  True       True
exhibitport
```

```
agenturl
server                 127.0.0.1  127.0.0.1
exhibiturl
vrf                    default    default
agentport              8981       8981
port                   31980      31980
```

> ⓘ After making configuration changes to your agents, you must restart the agent for the changes to take effect. Use the `netq config restart agent` command.

The CLI server commands enable the network administrator to configure and manage the CLI component. These commands enable you to add or remove CLI (essentially enabling/disabling the service), start and restart it, and view the configuration of the service.

> ⓘ Commands apply to one device at a time, and are run from the switch or host where the CLI is run.

The CLI configuration commands include:

```
netq config (add|del|show) cli server
netq config (start|restart) cli
```

This example shows how to start the CLI instance.

```
cumulus@switch~:$ netq config start cli
```

This example shows how to associate a NetQ platform with the device.

```
cumulus@switch~:$ netq config add cli server 10.1.3.101
```

Event Notification Commands

The notification configuration commands enable you to add, remove and show notification application integrations. These commands create the channels, filters, and rules needed to control event messaging. The commands include:

```
netq (add|del|show) notification channel
netq (add|del|show) notification rule
netq (add|del|show) notification filter
```

An integration includes at least one channel, PagerDuty or Slack. Filters are optional and defined by rules you create. If you have a filter, it must have at least one rule.

This example shows how to configure a PagerDuty channel:

```
cumulus@switch:~$ netq add notification channel pagerduty pd-netq-
events integration-key c6d666e210a8425298ef7abde0d1998
Successfully added/updated channel pd-netq-events
```

Refer to Integrate with Third-party Software and Hardware (Version 2.1.1) for details about using these commands and additional examples.

Trace Commands

The `trace` commands enable the network administrator to view the available paths between two nodes on the network currently and at a time in the past. You can perform a layer 2 or layer 3 trace, 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 syntax is:

```
netq trace <mac> [vlan <1-4096>] from (<src-hostname>|<ip-src>) [vrf
<vrf>] [around <text-time>] [json|detail|pretty] [debug]
netq trace <ip> from (<src-hostname>|<ip-src>) [vrf <vrf>] [around
<text-time>] [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 10.0.0.11 from 10.0.0.14 pretty
Number of Paths: 6
  Inconsistent PMTU among paths
Number of Paths with Errors: 0
Number of Paths with Warnings: 0
Path MTU: 9000
 leaf04 swp52 -- swp4 spine02 swp2 -- swp52 leaf02 peerlink.4094 --
peerlink.4094 leaf01 lo
                                                   peerlink.4094 --
peerlink.4094 leaf01 lo
 leaf04 swp51 -- swp4 spine01 swp2 -- swp51 leaf02 peerlink.4094 --
peerlink.4094 leaf01 lo
                                                   peerlink.4094 --
peerlink.4094 leaf01 lo
 leaf04 swp52 -- swp4 spine02 swp1 -- swp52 leaf01 lo
 leaf04 swp51 -- swp4 spine01 swp1 -- swp51 leaf01 lo
```

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

```
cumulus@oob-mgmt-server:~$ netq trace 10.0.0.11 from 10.0.0.14 detail
Number of Paths: 6
  Inconsistent PMTU among paths
Number of Paths with Errors: 0
Number of Paths with Warnings: 0
Path MTU: 9000
Id  Hop Hostname        InPort          InVlan InTunnel
InRtrIf         InVRF           OutRtrIf        OutVRF
OutTunnel             OutPort         OutVlan
--- --- --------------- --------------- ------ ---------------------
--------------- --------------- --------------- ---------------
--------------------- --------------- -------
1   1
leaf04
swp52           default                               swp52
    2   spine02         swp4
swp4            default         swp2
default                               swp2
    3   leaf02          swp52
swp52           default         peerlink.4094
default                               peerlink.4094
    4   leaf01          peerlink.4094
peerlink.4094
default
lo
--- --- --------------- --------------- ------ ---------------------
--------------- --------------- --------------- ---------------
--------------------- --------------- -------
2   1
leaf04
swp52           default                               swp52
    2   spine02         swp4
swp4            default         swp2
default                               swp2
    3   leaf02          swp52
swp52           default         peerlink.4094
default                               peerlink.4094
    4   leaf01          peerlink.4094
peerlink.4094
default
lo
--- --- --------------- --------------- ------ ---------------------
--------------- --------------- --------------- ---------------
--------------------- --------------- -------
3   1
leaf04
swp51           default                               swp51
    2   spine01         swp4
swp4            default         swp2
default                               swp2
```

```
    3   leaf02          swp51
swp51           default         peerlink.4094
default                               peerlink.4094
    4   leaf01          peerlink.4094
peerlink.4094
default
lo
--- --- --------------- --------------- ------ ---------------------
--------------- --------------- --------------- ---------------
--------------------- --------------- -------
4   1
leaf04
swp51           default                               swp51
    2   spine01         swp4
swp4            default         swp2
default                               swp2
    3   leaf02          swp51
swp51           default         peerlink.4094
default                               peerlink.4094
    4   leaf01          peerlink.4094
peerlink.4094
default
lo
--- --- --------------- --------------- ------ ---------------------
--------------- --------------- --------------- ---------------
--------------------- --------------- -------
5   1
leaf04
swp52           default                               swp52
    2   spine02         swp4
swp4            default         swp1
default                               swp1
    3   leaf01          swp52
swp52
default
lo
--- --- --------------- --------------- ------ ---------------------
--------------- --------------- --------------- ---------------
--------------------- --------------- -------
6   1
leaf04
swp51           default                               swp51
    2   spine01         swp4
swp4            default         swp1
default                               swp1
    3   leaf01          swp51
swp51
default
lo
--- --- --------------- --------------- ------ ---------------------
--------------- --------------- --------------- ---------------
--------------------- --------------- -------
```

Example: Running a trace based on the destination MAC address, in *pretty* output:

```
cumulus@switch:~$ netq trace A0:00:00:00:00:11 vlan 1001 from
Server03 pretty
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>
```

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
`netq config (add|del) agent server`	Adds or removes the NetQ Agent daemon to the network device where this command is run.
`netq config (add|del) cli server`	Adds or removes the CLI daemon to the network device where this command is run.

Command	Summary
`netq config show (agent|all|cli) [json]`	Displays configuration settings for the NetQ Agent, CLI, or both.
`netq (add|del|show) notification`	Configures channels, rules and filters for integration with third-party event notification tools, such as Pager Duty and Slack. This configuration was previously accomplished using the `netq config ts (add|del|show) notifier` commands.
`netq (add|del|show) notification proxy`	Adds or removes a notification proxy server to process and distribute event messages to third-party event notification tools, such as Slack and PagerDuty. Displays the configuration of the proxy.
`netq show events`	Displays events that have occurred network-wide during a given time range for a specific device, severity level, configuration file, and/or network protocol or service. Previously these events were obtained through the `netq show changes` command.
`netq show opta-health`	Displays the status of the various applications and services running on the NetQ Platform or NetQ Appliance.

Modified Commands

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

New Command	Old Command	What Changed

`netq check interfaces [around <text-time>] [json]`	`netq check interfaces <physical-hostname> <physical-port> and <peer-physical-hostname> <peer-physical-port> [around <text-time>] [json]`
	`netq check interfaces [unverified] [<physical-hostname> <physical-port> | <physical-hostname>] [around <text-time>] [json]`
		Removed ability to check interfaces on an individual device and port. All interface information is output for all devices in the network.

`netq <hostname> show * [around <text-time>] [json]`	`netq <hostname> show * changes [between <text-time> and <text-endtime>] [around <text-time>] [json]`	The `changes` and `between` options have been removed from all of the `netq show` commands. Collect this data using the new `netq show events` command.

New Command	Old Command	What Changed
`netq [<hostname>] show inventory license [cumulus] [status ok | status missing] [around <text-time>] [json]`	`netq [<hostname>] show inventory license [cumulus] [around <text-time>] [json]`	Added the ability to view only devices with license status of *ok* or *missing*.

`netq trace <ip> from (<src-hostname>|<ip-src>) [vrf <vrf>] [around <text-time>] [json|detail|pretty]`
`netq trace <mac> [vlan <1-4096>] from (<src-hostname>|<ip-src>) [vrf <vrf>] [around <text-time>] [json|detail|pretty]`
	`netq trace <ip> from (<src-hostname>|<ip-src>) [vrf <vrf>] [around <text-time>] [bidir] [json|detail|pretty]`
	`netq trace <mac> [vlan <1-4096>] from (<src-hostname>|<ip-src>) [vrf <vrf>] [around <text-time>] [bidir] [json|detail|pretty]`
		The bidirectional option (`bidir`) has been deprecated. Run the command in each direction when path validation is desired for both directions.

Deprecated Commands

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

Command	Alternative Command
`netq config ts (add|del|show) (notifier|server)`	`netq (add|del|show) notification` and `netq config (add|del|show) (agent|cli server)`
`netq config ts (start|stop|status|restart) notifier`	None. No longer necessary.
`netq config ts decommission`	`netq config del agent server`
`netq [<hostname>] show docker`	`netq [<hostname>] show kubernetes`
`netq example`	`netq help`
`netq resolve`	None
`netq-shell`	None. The netq-shell has been removed since all NetQ commands can be run from any node where a NetQ Agent is installed.

Command	Alternative Command
`netq check/show lnv`	LNV was deprecated in Cumulus Linux 3.7.4 and will be removed in Cumulus Linux 4.0.0. Cumulus NetQ will continue to support and return LNV data as long as you are running a supported version of Cumulus Linux (earlier than 4.0.0). For information on the support timeline, read this knowledge base article.

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 an earlier point 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.

Validate the Network Fabric

You can validate the following network fabric elements:

```
cumulus@leaf01:mgmt-vrf:~$ netq check
    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
```

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 fabric elements. 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: 1
Hostname          Status           Last Changed
----------------- ---------------- -------------------------
leaf01            Rotten           8d:13h:34m:51s
```

Validate that all corresponding interface links have matching MTUs. The first shows no mismatches, the second shows an error.

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

cumulus@switch:~$ netq check mtu
Checked Nodes: 13, Checked Links: 173, Failed Nodes: 1, Failed Links:
1
MTU mismatch found on following links
Hostname          Interface                 MTU    Peer
Peer Interface            Peer MTU Error
----------------- ------------------------- ------ -----------------
------------------------- -------- ---------------
leaf01            -                         -      -
-                         -        Rotten Agent
```

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
```

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 communicating over a particular VRF:

```
cumulus@switch:~$ netq check bgp vrf DataVrf1081
Total Nodes: 25, Failed Nodes: 1, Total Sessions: 52 , Failed
Sessions: 1
Hostname          VRF             Peer Name         Peer Hostname
Reason                                        Last Changed
----------------- --------------- ----------------- -----------------
---------------------------------------------
-------------------------
exit-1            DataVrf1081     swp7.3            firewall-2
BGP session with peer firewall-2 (swp7.3 vrf  1d:5h:47m:31s

DataVrf1081) failed,

reason: Peer not configured
```

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 might 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            swp7                      firewall02
swp3                      Speed mismatch (10G, n/a),

Autoneg mismatch (off, n/a)
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
    clag          :  Cumulus Multi-chassis LAG
    events        :  Display changes over time
    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
    notification  :  Send notifications to Slack or PagerDuty
    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@switch:~$ netq show agents
Matching agents records:
Hostname          Status           NTP Sync
Version                              Sys Uptime                Agent
Uptime              Reinitialize Time          Last Changed
----------------- ---------------- --------
------------------------------------ -------------------------
------------------------- --------------------------
-------------------------
edge01            Fresh            yes      2.1.0-ub16.
04u15~1555612152.6e34b56  2d:4h:27m:34s             2d:4h:27m:
27s             2d:4h:27m:27s              Sun Apr 21 16:00:50 2019
exit01            Fresh            yes      2.1.0-cl3u15~1555612272.
6e34b56      2d:4h:26m:52s             2d:4h:26m:44s             2d:
4h:26m:44s              Sun Apr 21 16:00:52 2019
exit02            Fresh            yes      2.1.0-cl3u15~1555612272.
6e34b56      2d:4h:26m:58s             2d:4h:26m:49s             2d:
4h:26m:49s              Sun Apr 21 16:01:19 2019
leaf01            Fresh            yes      2.1.0-cl3u15~1555612272.
6e34b56      2d:4h:26m:50s             19m:34.763s               19m:
34.763s                Sun Apr 21 20:05:45 2019
leaf02            Fresh            yes      2.1.0-cl3u15~1555612272.
6e34b56      2d:4h:27m:0s              2d:4h:26m:51s             2d:
4h:26m:51s              Sun Apr 21 16:01:43 2019
```

```
leaf03            Fresh            yes      2.1.0-cl3u15~1555612272.
6e34b56      2d:4h:26m:59s             2d:4h:26m:50s             2d:
4h:26m:50s              Sun Apr 21 16:01:23 2019
leaf04            Fresh            yes      2.1.0-cl3u15~1555612272.
6e34b56      2d:4h:27m:1s              2d:4h:26m:53s             2d:
4h:26m:53s              Sun Apr 21 16:01:27 2019
server01          Fresh            yes      2.1.0-ub16.
04u15~1555612152.6e34b56  2d:4h:24m:57s             2d:4h:24m:
49s             2d:4h:24m:49s              Sun Apr 21 16:00:43 2019
server02          Fresh            yes      2.1.0-ub16.
04u15~1555612152.6e34b56  2d:4h:24m:56s             2d:4h:24m:
48s             2d:4h:24m:48s              Sun Apr 21 16:00:46 2019
server03          Fresh            yes      2.1.0-ub16.
04u15~1555612152.6e34b56  2d:4h:24m:56s             2d:4h:24m:
48s             2d:4h:24m:48s              Sun Apr 21 16:00:52 2019
server04          Fresh            yes      2.1.0-ub16.
04u15~1555612152.6e34b56  2d:4h:24m:56s             2d:4h:24m:
48s             2d:4h:24m:48s              Sun Apr 21 16:00:43 2019
spine01           Fresh            yes      2.1.0-cl3u15~1555612272.
6e34b56      2d:4h:27m:2s              2d:4h:26m:54s             2d:
4h:26m:54s              Sun Apr 21 16:01:33 2019
spine02           Fresh            yes      2.1.0-cl3u15~1555612272.
6e34b56      2d:4h:26m:56s             2d:4h:26m:48s             2d:
4h:26m:48s              Sun Apr 21 16:01:12 2019
```

Some additional examples follow.

View the status of BGP:

```
cumulus@switch:~$ netq show bgp
Hostname          Neighbor                     VRF
ASN        Peer ASN   PfxRx        Last Changed
----------------- ---------------------------- ---------------
---------- ---------- ------------ -------------------------
exit01            swp44(internet)              vrf1
65041      25253      2/-/-        Fri Apr 19 16:00:40 2019
exit01            swp51(spine01)               default
65041      65020      8/-/59       Fri Apr 19 16:00:40 2019
exit01            swp52(spine02)               default
65041      65020      8/-/59       Fri Apr 19 16:00:40 2019
exit02            swp44(internet)              vrf1
65042      25253      7/-/-        Fri Apr 19 16:00:40 2019
exit02            swp51(spine01)               default
65042      65020      8/-/59       Fri Apr 19 16:00:40 2019
exit02            swp52(spine02)               default
65042      65020      8/-/59       Fri Apr 19 16:00:40 2019
leaf01            peerlink.4094(leaf02)        default
65011      65011      9/-/34       Fri Apr 19 16:00:40 2019
leaf01            swp51(spine01)               default
65011      65020      6/-/34       Fri Apr 19 16:00:40 2019
```

```
leaf01            swp52(spine02)               default
65011      65020      6/-/34       Fri Apr 19 16:00:40 2019
leaf02            peerlink.4094(leaf01)        default
65011      65011      9/-/34       Fri Apr 19 16:00:40 2019
leaf02            swp51(spine01)               default
65011      65020      6/-/34       Fri Apr 19 16:00:40 2019
leaf02            swp52(spine02)               default
65011      65020      6/-/34       Fri Apr 19 16:00:40 2019
leaf03            peerlink.4094(leaf04)        default
65012      65012      9/-/34       Fri Apr 19 16:00:40 2019
leaf03            swp51(spine01)               default
65012      65020      6/-/34       Fri Apr 19 16:00:40 2019
leaf03            swp52(spine02)               default
65012      65020      6/-/34       Fri Apr 19 16:00:40 2019
leaf04            peerlink.4094(leaf03)        default
65012      65012      9/-/34       Fri Apr 19 16:00:40 2019
leaf04            swp51(spine01)               default
65012      65020      6/-/34       Fri Apr 19 16:00:40 2019
leaf04            swp52(spine02)               default
65012      65020      6/-/34       Fri Apr 19 16:00:40 2019
spine01           swp1(leaf01)                 default
65020      65011      3/-/14       Fri Apr 19 16:00:40 2019
spine01           swp2(leaf02)                 default
65020      65011      3/-/14       Fri Apr 19 16:00:40 2019
spine01           swp29(exit02)                default
65020      65042      1/-/3        Fri Apr 19 16:00:40 2019
spine01           swp3(leaf03)                 default
65020      65012      3/-/14       Fri Apr 19 16:00:40 2019
spine01           swp30(exit01)                default
65020      65041      1/-/3        Fri Apr 19 16:00:40 2019
spine01           swp4(leaf04)                 default
65020      65012      3/-/14       Fri Apr 19 16:00:40 2019
spine02           swp1(leaf01)                 default
65020      65011      3/-/12       Fri Apr 19 16:00:40 2019
spine02           swp2(leaf02)                 default
65020      65011      3/-/12       Fri Apr 19 16:00:40 2019
spine02           swp29(exit02)                default
65020      65042      1/-/3        Fri Apr 19 16:00:40 2019
spine02           swp3(leaf03)                 default
65020      65012      3/-/12       Fri Apr 19 16:00:40 2019
spine02           swp30(exit01)                default
65020      65041      1/-/3        Fri Apr 19 16:00:40 2019
spine02           swp4(leaf04)                 default
65020      65012      3/-/12       Fri Apr 19 16:00:40 2019
```

View the status of your VLANs:

```
cumulus@switch:~$ netq show vlan
Matching vlan records:
```

```
Hostname          VLANs                     SVIs
Last Changed
----------------- ------------------------- -------------------------
-------------------------
server11          1
Thu Feb  7 00:17:48 2019
server21          1
Thu Feb  7 00:17:48 2019
server11          1
Thu Feb  7 00:17:48 2019
server13          1
Thu Feb  7 00:17:48 2019
server21          1
Thu Feb  7 00:17:48 2019
server23          1
Thu Feb  7 00:17:48 2019
leaf01            100-106,1000-1009         100-106 1000-1009
Thu Feb  7 00:17:49 2019
leaf02            100-106,1000-1009         100-106 1000-1009
Thu Feb  7 00:17:49 2019
leaf11            100-106,1000-1009         100-106 1000-1009
Thu Feb  7 00:17:49 2019
leaf12            100-106,1000-1009         100-106 1000-1009
Thu Feb  7 00:17:50 2019
leaf21            100-106,1000-1009         100-106 1000-1009
Thu Feb  7 00:17:50 2019
leaf22            100-106,1000-1009         100-106 1000-1009
Thu Feb  7 00:17:50 2019
```

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                                             Wed Feb  6 23:02:35
2019
exit01            fan2            fan tray 1, fan 2
ok                                             Wed Feb  6 23:02:35
2019
exit01            fan3            fan tray 2, fan 1
ok                                             Wed Feb  6 23:02:35
2019
```

```
exit01            fan4            fan tray 2, fan 2
ok                                             Wed Feb  6 23:02:35
2019
exit01            fan5            fan tray 3, fan 1
ok                                             Wed Feb  6 23:02:35
2019
exit01            fan6            fan tray 3, fan 2
ok                                             Wed Feb  6 23:02:35
2019
exit01            psu1fan1        psu1 fan
ok                                             Wed Feb  6 23:02:35
2019
exit01            psu2fan1        psu2 fan
ok                                             Wed Feb  6 23:02:35
2019
exit02            fan1            fan tray 1, fan 1
ok                                             Wed Feb  6 23:03:35
2019
exit02            fan2            fan tray 1, fan 2
ok                                             Wed Feb  6 23:03:35
2019
exit02            fan3            fan tray 2, fan 1
ok                                             Wed Feb  6 23:03:35
2019
exit02            fan4            fan tray 2, fan 2
ok                                             Wed Feb  6 23:03:35
2019
exit02            fan5            fan tray 3, fan 1
ok                                             Wed Feb  6 23:03:35
2019
exit02            fan6            fan tray 3, fan 2
ok                                             Wed Feb  6 23:03:35
2019
exit02            psu1fan1        psu1 fan
ok                                             Wed Feb  6 23:03:35
2019
exit02            psu2fan1        psu2 fan
ok                                             Wed Feb  6 23:03:35
2019
leaf01            fan1            fan tray 1, fan 1
ok                                             Wed Feb  6 23:01:12
2019
leaf01            fan2            fan tray 1, fan 2
ok                                             Wed Feb  6 23:01:12
2019
leaf01            fan3            fan tray 2, fan 1
ok                                             Wed Feb  6 23:01:12
2019
leaf01            fan4            fan tray 2, fan 2
ok                                             Wed Feb  6 23:01:12
2019
```

```
leaf01            fan5            fan tray 3, fan 1
ok                                             Wed Feb  6 23:01:12
2019
leaf01            fan6            fan tray 3, fan 2
ok                                             Wed Feb  6 23:01:12
2019
leaf01            psu1fan1        psu1 fan
ok                                             Wed Feb  6 23:01:12
2019
leaf01            psu2fan1        psu2 fan
ok                                             Wed Feb  6 23:01:12
2019
leaf02            fan1            fan tray 1, fan 1
ok                                             Wed Feb  6 22:59:54
2019
leaf02            fan2            fan tray 1, fan 2
ok                                             Wed Feb  6 22:59:54
2019
leaf02            fan3            fan tray 2, fan 1
ok                                             Wed Feb  6 22:59:54
2019
leaf02            fan4            fan tray 2, fan 2
ok                                             Wed Feb  6 22:59:54
2019
leaf02            fan5            fan tray 3, fan 1
ok                                             Wed Feb  6 22:59:54
2019
...
```

Monitor Switch Hardware and Software

With NetQ, a network administrator can monitor both the switch hardware and software components for misconfigurations. NetQ helps answer questions such as:

- What switches do I have in the network?
- What hardware and software are installed on my switches?
- 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.

The NetQ CLI provides the `netq show inventory`, `netq show sensors`, and `netq show events` commands to monitor switches.

Contents

This topic describes how to...

Monitor Switch and Host Hardware Information

You can view summary information about all switches and hosts along with their key components, including the motherboard, ASIC, microprocessor, disk and memory information.

To view the switch and host information with the CLI, use the following `netq show` commands:

```
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 license [cumulus] [status ok |
status missing] [around <text-time>] [json]
netq [<hostname>] show inventory memory [type <memory-type>|vendor
<memory-vendor>] [json]
netq [<hostname>] show inventory os [version <os-version>|name <os-
name>] [json]

netq [<hostname>] show sensors all [around <text-time>] [json]
netq [<hostname>] show sensors psu [<psu-name>] [around <text-time>]
[json]
netq [<hostname>] show sensors temp [<temp-name>] [around <text-
time>] [json]
netq [<hostname>] show sensors fan [<fan-name>] [around <text-time>]
[json]

netq [<hostname>] show events [level info | level error | level
warning | level critical | level debug] [type sensors] [between <text-
time> and <text-endtime>] [json]
```

ⓘ 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.

ⓘ

The keyword values for the `vendor`, `model`, `model-id`, `arch`, `name`, `transport`, `type`, `version`, `psu`, `temp`, and `fan` 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.

To view the switch and host information with the GUI, use the Devices Inventory card workflow which contains a small card with a count of each device type in your network, a medium card displaying the operating systems running on each set of devices, large cards with component information statistics, and full-screen cards displaying tables with attributes of all switches and all hosts in your network.

View a Summary of Your Network Inventory

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
----------------- -------------------- --------------- --------
--------------- -----------------------------------
edge01            N/A                  Ubuntu          x86_64   N
/A             N/A
exit01            VX                   CL              x86_64
VX              N/A
exit02            VX                   CL              x86_64
VX              N/A
leaf01            VX                   CL              x86_64
VX              N/A
leaf02            VX                   CL              x86_64
VX              N/A
leaf03            VX                   CL              x86_64
VX              N/A
leaf04            VX                   CL              x86_64
VX              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                   CL              x86_64
VX              N/A
spine02           VX                   CL              x86_64
VX              N/A
```

View Information about the ASIC on all Switches

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@switch:~$ 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  QTFCO7623002C             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 Fan Health for All Switches

Fan, power supply unit, and temperature sensors are available to provide additional data about the NetQ Platform operation. To view the health of fans in your switches, use the `netq show sensors fan` command. If you name the fans in all of your switches consistently, you can view more information at once.

In this example, we look at the state of all fans with the name *fan1*.

```
cumulus@switch:~$ netq show sensors fan fan1
Hostname          Name            Description
State      Speed      Max      Min
Message                             Last Changed
```

```
----------------- --------------- -----------------------------------
---------- ---------- -------- --------
----------------------------------- -------------------------
exit01            fan1            fan tray 1, fan 1
ok         2500       29000
2500                                         Fri Apr 19 16:01:17 2019
exit02            fan1            fan tray 1, fan 1
ok         2500       29000
2500                                         Fri Apr 19 16:01:33 2019
leaf01            fan1            fan tray 1, fan 1
ok         2500       29000
2500                                         Sun Apr 21 20:07:12 2019
leaf02            fan1            fan tray 1, fan 1
ok         2500       29000
2500                                         Fri Apr 19 16:01:41 2019
leaf03            fan1            fan tray 1, fan 1
ok         2500       29000
2500                                         Fri Apr 19 16:01:44 2019
leaf04            fan1            fan tray 1, fan 1
ok         2500       29000
2500                                         Fri Apr 19 16:01:36 2019
spine01           fan1            fan tray 1, fan 1
ok         2500       29000
2500                                         Fri Apr 19 16:01:52 2019
spine02           fan1            fan tray 1, fan 1
ok         2500       29000
2500                                         Fri Apr 19 16:01:08 2019
```

Use tab completion to determine the names of the fans in your switches:

```
cumulus@switch:~$ netq show sensors fan <<press tab>>
    around : Go back in time to around ...
    fan1 : Fan Name
    fan2 : Fan Name
    fan3 : Fan Name
    fan4 : Fan Name
    fan5 : Fan Name
    fan6 : Fan Name
    json : Provide output in JSON
    psu1fan1 : Fan Name
    psu2fan1 : Fan Name
    <ENTER>
```

To view the status for a particular switch, use the optional *hostname* parameter.

```
cumulus@switch:~$ netq leaf01 show sensors fan fan1
```

```
Hostname          Name            Description
State      Speed      Max      Min
Message                             Last Changed
----------------- --------------- -----------------------------------
---------- ---------- -------- --------
----------------------------------- -------------------------
leaf01            fan1            fan tray 1, fan 1
ok         2500       29000
2500                                         Sun Apr 21 20:07:12 2019
```

View PSU Health for All Switches

Fan, power supply unit, and temperature sensors are available to provide additional data about the NetQ Platform operation. To view the health of PSUs in your switches, use the `netq show sensors psu` command. If you name the PSUs in all of your switches consistently, you can view more information at once.

In this example, we look at the state of all PSUs with the name *psu2*.

```
cumulus@switch:~$ netq show sensors psu psu2
Matching sensors records:
Hostname          Name            State
Message                             Last Changed
----------------- --------------- ----------
----------------------------------- -------------------------
exit01            psu2
ok                                               Fri Apr 19 16:01:17
2019
exit02            psu2
ok                                               Fri Apr 19 16:01:33
2019
leaf01            psu2
ok                                               Sun Apr 21 20:07:12
2019
leaf02            psu2
ok                                               Fri Apr 19 16:01:41
2019
leaf03            psu2
ok                                               Fri Apr 19 16:01:44
2019
leaf04            psu2
ok                                               Fri Apr 19 16:01:36
2019
spine01           psu2
ok                                               Fri Apr 19 16:01:52
2019
spine02           psu2
ok                                               Fri Apr 19 16:01:08
2019
```

Use Tab completion to determine the names of the PSUs in your switches. Use the optional *hostname* parameter to view the PSU state for a given switch.

View the Temperature in All switches

Fan, power supply unit, and temperature sensors are available to provide additional data about the NetQ Platform operation. To view the temperature sensor status, current temperature, and configured threshold values, use the `netq show sensors temp` command. If you name the temperature sensors in all of your switches consistently, you can view more information at once.

In this example, we look at the state of all temperature sensors with the name *psu1temp1*.

```
cumulus@switch:~$ netq show sensors temp psu2temp1
Matching sensors records:
Hostname          Name            Description
State      Temp     Critical
Max      Min      Message                                Last Changed

----------------- --------------- -----------------------------------
---------- -------- -------- -------- --------
----------------------------------- -------------------------

exit01            psu2temp1       psu2 temp
sensor                    ok          25       85       80       5
                                      Fri Apr 19 16:01:17 2019

exit02            psu2temp1       psu2 temp
sensor                    ok          25       85       80       5
                                      Fri Apr 19 16:01:33 2019

leaf01            psu2temp1       psu2 temp
sensor                    ok          25       85       80       5
                                      Sun Apr 21 20:07:12 2019

leaf02            psu2temp1       psu2 temp
sensor                    ok          25       85       80       5
                                      Fri Apr 19 16:01:41 2019

leaf03            psu2temp1       psu2 temp
sensor                    ok          25       85       80       5
                                      Fri Apr 19 16:01:44 2019

leaf04            psu2temp1       psu2 temp
sensor                    ok          25       85       80       5
                                      Fri Apr 19 16:01:36 2019

spine01           psu2temp1       psu2 temp
sensor                    ok          25       85       80       5
                                      Fri Apr 19 16:01:52 2019
```

```
spine02           psu2temp1       psu2 temp
sensor                    ok          25        85        80        5
                                      Fri Apr 19 16:01:08 2019
```

Use Tab completion to determine the names of the temperature sensors in your switches. Use the optional *hostname* parameter to view the temperature state, current temperature, and threshold values for a given switch.

View All Sensor Data

To view all fan data, all PSU data, or all temperature data from the sensors, you must view all of the sensor data. The more consistently you name your sensors, the easier it will be to view the full sensor data.

```
cumulus@switch:~$ netq show sensors all
Matching sensors records:
Hostname          Name            Description
State      Message                            Last Changed
----------------- --------------- -----------------------------------
---------- -----------------------------------
-------------------------
exit01            fan1            fan tray 1, fan
1                   ok                                               Fri
Apr 19 16:01:17 2019
exit01            fan2            fan tray 1, fan
2                   ok                                               Fri
Apr 19 16:01:17 2019
exit01            fan3            fan tray 2, fan
1                   ok                                               Fri
Apr 19 16:01:17 2019
exit01            fan4            fan tray 2, fan
2                   ok                                               Fri
Apr 19 16:01:17 2019
exit01            fan5            fan tray 3, fan
1                   ok                                               Fri
Apr 19 16:01:17 2019
exit01            fan6            fan tray 3, fan
2                   ok                                               Fri
Apr 19 16:01:17 2019
exit01            psu1fan1        psu1
fan                            ok
        Fri Apr 19 16:01:17 2019
exit01            psu1temp1       psu1 temp
sensor                    ok
  Fri Apr 19 16:01:17 2019
exit01            psu2fan1        psu2
fan                            ok
        Fri Apr 19 16:01:17 2019
```

```
exit01            psu2temp1       psu2 temp
sensor                    ok
  Fri Apr 19 16:01:17 2019
exit01            temp1           board sensor near
cpu               ok                                             Fri
Apr 19 16:01:17 2019
exit01            temp2           board sensor near virtual
switch    ok                                             Fri Apr 19
16:01:17 2019
exit01            temp3           board sensor at front left
corner   ok                                             Fri Apr 19 16:
01:17 2019
exit01            temp4           board sensor at front right
corner  ok                                             Fri Apr 19 16:
01:17 2019
exit01            temp5           board sensor near
fan               ok                                             Fri
Apr 19 16:01:17 2019
exit02            fan1            fan tray 1, fan
1                   ok                                             Fri
Apr 19 16:01:33 2019
exit02            fan2            fan tray 1, fan
2                   ok                                             Fri
Apr 19 16:01:33 2019
exit02            fan3            fan tray 2, fan
1                   ok                                             Fri
Apr 19 16:01:33 2019
exit02            fan4            fan tray 2, fan
2                   ok                                             Fri
Apr 19 16:01:33 2019
exit02            fan5            fan tray 3, fan
1                   ok                                             Fri
Apr 19 16:01:33 2019
exit02            fan6            fan tray 3, fan
2                   ok                                             Fri
Apr 19 16:01:33 2019
exit02            psu1fan1        psu1
fan                            ok
        Fri Apr 19 16:01:33 2019
exit02            psu1temp1       psu1 temp
sensor                    ok
  Fri Apr 19 16:01:33 2019
...
```

View All Sensor-related Events

You can view the events that are triggered by the sensors using the `netq show events` command. You can narrow the focus to only critical events using the severity *level* option.

```
cumulus@switch:~$ netq show events type sensors
```

```
No matching events records found

cumulus@switch:~$ netq show events level critical type sensors
No matching events records found
```

Monitor Switch Software Information

The syntax for this command is:

```
netq [<hostname>] show agents
netq [<hostname>] show inventory brief [json]
netq [<hostname>] show inventory license [cumulus] [status ok|status
missing] [around <text-time>] [json]
netq [<hostname>] show inventory os [version <os-version>|name <os-
name>] [json]
netq [<hostname>] show events [level info|level error|level
warning|level critical|level debug] [type license|type os] [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
----------------- ---------------
------------------------------------ -------------------------
edge01            Ubuntu
16.04                                Fri Apr 19 16:01:18 2019
exit01            CL
3.7.5                                Fri Apr 19 16:01:13 2019
exit02            CL
3.7.5                                Fri Apr 19 16:01:38 2019
leaf01            CL
3.7.5                                Sun Apr 21 20:07:09 2019
leaf02            CL
3.7.5                                Fri Apr 19 16:01:46 2019
leaf03            CL
3.7.5                                Fri Apr 19 16:01:41 2019
leaf04            CL
3.7.5                                Fri Apr 19 16:01:32 2019
server01          Ubuntu
16.04                                Fri Apr 19 16:01:55 2019
server02          Ubuntu
16.04                                Fri Apr 19 16:01:55 2019
server03          Ubuntu
16.04                                Fri Apr 19 16:01:55 2019
server04          Ubuntu
16.04                                Fri Apr 19 16:01:55 2019
spine01           CL
3.7.5                                Fri Apr 19 16:01:49 2019
spine02           CL
3.7.5                                Fri Apr 19 16:01:05 2019
```

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 3.7.5 installed.

```
cumulus@switch:~$ netq show inventory os version 3.7.5
Matching inventory records:
Hostname          Name
Version                              Last Changed
----------------- ---------------
------------------------------------ -------------------------
exit01            CL
3.7.5                                Fri Apr 19 16:01:13 2019
exit02            CL
3.7.5                                Fri Apr 19 16:01:38 2019
leaf01            CL
3.7.5                                Sun Apr 21 20:07:09 2019
leaf02            CL
3.7.5                                Fri Apr 19 16:01:46 2019
```

```
leaf03            CL
3.7.5                                Fri Apr 19 16:01:41 2019
leaf04            CL
3.7.5                                Fri Apr 19 16:01:32 2019
spine01           CL
3.7.5                                Fri Apr 19 16:01:49 2019
spine02           CL
3.7.5                                Fri Apr 19 16:01:05 2019
```

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 events type os between 16d and 21d
Matching inventory records:
Hostname          Name
Version                                DB State   Last Changed
----------------- ---------------
------------------------------------ ----------
-------------------------
mlx-2410a1-05       Cumulus Linux
3.7.3                                Add           Tue Feb 12 18:30:
53 2019
mlx-2700-11       Cumulus Linux
3.7.3                                Add           Tue Feb 12 18:30:
45 2019
mlx-2100-05       Cumulus Linux
3.7.3                                Add           Tue Feb 12 18:30:
26 2019
mlx-2100-05       Cumulus Linux   3.7.3~1533263174.
bce9472                Add           Wed Feb 13 11:10:47 2019
mlx-2700-11       Cumulus Linux   3.7.3~1533263174.
bce9472                Add           Wed Feb 13 11:10:38 2019
mlx-2100-05       Cumulus Linux   3.7.3~1533263174.
bce9472                Add           Wed Feb 13 11:10:42 2019
mlx-2700-11       Cumulus Linux   3.7.3~1533263174.
bce9472                Add           Wed Feb 13 11:10:51 2019
```

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
----------------- --------------- ---------- -------------------------
```

```
edge01            Cumulus Linux   N/A        Fri Apr 19 16:01:18 2019
exit01            Cumulus Linux   ok         Fri Apr 19 16:01:13 2019
exit02            Cumulus Linux   ok         Fri Apr 19 16:01:38 2019
leaf01            Cumulus Linux   ok         Sun Apr 21 20:07:09 2019
leaf02            Cumulus Linux   ok         Fri Apr 19 16:01:46 2019
leaf03            Cumulus Linux   ok         Fri Apr 19 16:01:41 2019
leaf04            Cumulus Linux   ok         Fri Apr 19 16:01:32 2019
server01          Cumulus Linux   N/A        Fri Apr 19 16:01:55 2019
server02          Cumulus Linux   N/A        Fri Apr 19 16:01:55 2019
server03          Cumulus Linux   N/A        Fri Apr 19 16:01:55 2019
server04          Cumulus Linux   N/A        Fri Apr 19 16:01:55 2019
spine01           Cumulus Linux   ok         Fri Apr 19 16:01:49 2019
spine02           Cumulus Linux   ok         Fri Apr 19 16:01:05 2019
```

You can view the historical state of licenses using the around keyword. 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
----------------- --------------- ---------- -------------------------
edge01            Cumulus Linux   N/A        Tue Apr 2 14:01:18 2019
exit01            Cumulus Linux   ok         Tue Apr 2 14:01:13 2019
exit02            Cumulus Linux   ok         Tue Apr 2 14:01:38 2019
leaf01            Cumulus Linux   ok         Tue Apr 2 20:07:09 2019
leaf02            Cumulus Linux   ok         Tue Apr 2 14:01:46 2019
leaf03            Cumulus Linux   ok         Tue Apr 2 14:01:41 2019
leaf04            Cumulus Linux   ok         Tue Apr 2 14:01:32 2019
server01          Cumulus Linux   N/A        Tue Apr 2 14:01:55 2019
server02          Cumulus Linux   N/A        Tue Apr 2 14:01:55 2019
server03          Cumulus Linux   N/A        Tue Apr 2 14:01:55 2019
server04          Cumulus Linux   N/A        Tue Apr 2 14:01:55 2019
spine01           Cumulus Linux   ok         Tue Apr 2 14:01:49 2019
spine02           Cumulus Linux   ok         Tue Apr 2 14:01:05 2019
```

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 24 hours ago.

```
cumulus@switch:~$ netq spine01 show events type license between now
and 24h
No matching events 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
----------------- -------------------- --------------- --------
--------------- -----------------------------------
edge01            N/A                  Ubuntu          x86_64   N
/A             N/A
exit01            VX                   CL              x86_64
VX              N/A
exit02            VX                   CL              x86_64
VX              N/A
leaf01            VX                   CL              x86_64
VX              N/A
leaf02            VX                   CL              x86_64
VX              N/A
leaf03            VX                   CL              x86_64
VX              N/A
leaf04            VX                   CL              x86_64
VX              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                   CL              x86_64
VX              N/A
spine02           VX                   CL              x86_64
VX              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.

```
cumulus@switch:~$ netq show agents
Matching agents records:
Hostname          Status           NTP Sync
Version                              Sys Uptime                Agent
Uptime              Reinitialize Time          Last Changed
```

```
----------------- ---------------- --------
------------------------------------ -------------------------
------------------------- --------------------------
-------------------------
edge01            Fresh            yes      2.1.0-ub16.
04u15~1555612152.6e34b56  2d:7h:2m:12s              2d:7h:2m:
5s               2d:7h:2m:5s                Sun Apr 21 16:00:50 2019
exit01            Fresh            yes      2.1.0-cl3u15~1555612272.
6e34b56      2d:7h:1m:30s              2d:7h:1m:22s              2d:
7h:1m:22s               Sun Apr 21 16:00:52 2019
exit02            Fresh            yes      2.1.0-cl3u15~1555612272.
6e34b56      2d:7h:1m:36s              2d:7h:1m:27s              2d:
7h:1m:27s               Sun Apr 21 16:01:19 2019
leaf01            Fresh            yes      2.1.0-cl3u15~1555612272.
6e34b56      2d:7h:1m:28s              2h:54m:12s                2h:
54m:12s                 Sun Apr 21 20:05:45 2019
leaf02            Fresh            yes      2.1.0-cl3u15~1555612272.
6e34b56      2d:7h:1m:38s              2d:7h:1m:29s              2d:
7h:1m:29s               Sun Apr 21 16:01:43 2019
leaf03            Fresh            yes      2.1.0-cl3u15~1555612272.
6e34b56      2d:7h:1m:37s              2d:7h:1m:28s              2d:
7h:1m:28s               Sun Apr 21 16:01:23 2019
leaf04            Fresh            yes      2.1.0-cl3u15~1555612272.
6e34b56      2d:7h:1m:39s              2d:7h:1m:31s              2d:
7h:1m:31s               Sun Apr 21 16:01:27 2019
server01          Fresh            yes      2.1.0-ub16.
04u15~1555612152.6e34b56  2d:6h:59m:35s             2d:6h:59m:
27s             2d:6h:59m:27s              Sun Apr 21 16:00:43 2019
server02          Fresh            yes      2.1.0-ub16.
04u15~1555612152.6e34b56  2d:6h:59m:34s             2d:6h:59m:
26s             2d:6h:59m:26s              Sun Apr 21 16:00:46 2019
server03          Fresh            yes      2.1.0-ub16.
04u15~1555612152.6e34b56  2d:6h:59m:34s             2d:6h:59m:
26s             2d:6h:59m:26s              Sun Apr 21 16:00:52 2019
server04          Fresh            yes      2.1.0-ub16.
04u15~1555612152.6e34b56  2d:6h:59m:34s             2d:6h:59m:
26s             2d:6h:59m:26s              Sun Apr 21 16:00:43 2019
spine01           Fresh            yes      2.1.0-cl3u15~1555612272.
6e34b56      2d:7h:1m:40s              2d:7h:1m:32s              2d:
7h:1m:32s               Sun Apr 21 16:01:33 2019
spine02           Fresh            yes      2.1.0-cl3u15~1555612272.
6e34b56      2d:7h:1m:34s              2d:7h:1m:26s              2d:
7h:1m:26s               Sun Apr 21 16:01:12 2019
```

You can narrow your focus in several ways:

- View the state of the NetQ Agent on a given device using the *hostname* keyword.
- View only the NetQ Agents that are fresh or rotten using the *fresh* or *rotten* keyword.
- View the state of NetQ Agents at an earlier time using the *around* keyword.

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
- helpledmgrd: 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
- netqd: NetQ application daemon
- ntp: NTP service
- ntpd: NTP 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
- 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>] status (ok|warning|error|fail) [around <text-time>] [json]
netq [<hostname>] show events [level info | level error | level warning | level critical | level debug] type services [between <text-time> and <text-endtime>] [json]
```

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.

It is useful to have colored output for this show command. To configure colored output, run the **`netq config add color`** command.

```
cumulus@switch:~$ netq show services
Hostname          Service              PID   VRF             Enabled
Active Monitored Status           Uptime                    Last
Changed
----------------- -------------------- ----- --------------- -------
------ --------- ---------------- -------------------------
-------------------------
leaf01            bgpd                 2872  default         yes
yes    yes       ok               1d:6h:43m:59s             Fri Feb
15 17:28:24 2019
leaf01            clagd                n/a   default         yes
no     yes       n/a              1d:6h:43m:35s             Fri Feb
15 17:28:48 2019
leaf01            ledmgrd              1850  default         yes
yes    no        ok               1d:6h:43m:59s             Fri Feb
15 17:28:24 2019
leaf01            lldpd                2651  default         yes
yes    yes       ok               1d:6h:43m:27s             Fri Feb
15 17:28:56 2019
leaf01            mstpd                1746  default         yes
yes    yes       ok               1d:6h:43m:35s             Fri Feb
15 17:28:48 2019
leaf01            neighmgrd            1986  default         yes
yes    no        ok               1d:6h:43m:59s             Fri Feb
15 17:28:24 2019
leaf01            netq-agent           8654  mgmt            yes
yes    yes       ok               1d:6h:43m:29s             Fri Feb
15 17:28:54 2019
leaf01            netqd                8848  mgmt            yes
yes    yes       ok               1d:6h:43m:29s             Fri Feb
15 17:28:54 2019
leaf01            ntp                  8478  mgmt            yes
yes    yes       ok               1d:6h:43m:29s             Fri Feb
15 17:28:54 2019
leaf01            ptmd                 2743  default         yes
yes    no        ok               1d:6h:43m:59s             Fri Feb
15 17:28:24 2019
leaf01            pwmd                 1852  default         yes
yes    no        ok               1d:6h:43m:59s             Fri Feb
15 17:28:24 2019
leaf01            smond                1826  default         yes
yes    yes       ok               1d:6h:43m:27s             Fri Feb
15 17:28:56 2019
leaf01            ssh                  2106  default         yes
yes    no        ok               1d:6h:43m:59s             Fri Feb
15 17:28:24 2019
```

```
leaf01            syslog               8254  default         yes
yes    no       ok             1d:6h:43m:59s             Fri Feb
15 17:28:24 2019
leaf01            zebra                2856  default         yes
yes    yes      ok             1d:6h:43m:59s             Fri Feb
15 17:28:24 2019
leaf02            bgpd                 2867  default         yes
yes    yes      ok             1d:6h:43m:55s             Fri Feb
15 17:28:28 2019
leaf02            clagd                n/a   default         yes
no     yes      n/a            1d:6h:43m:31s             Fri Feb
15 17:28:53 2019
leaf02            ledmgrd              1856  default         yes
yes    no       ok             1d:6h:43m:55s             Fri Feb
15 17:28:28 2019
leaf02            lldpd                2646  default         yes
yes    yes      ok             1d:6h:43m:30s             Fri Feb
15 17:28:53 2019
...
```

You can also view services information in JSON format:

```
cumulus@switch:~$ netq show services json
{
    "services":[
        {
            "status":"ok",
            "uptime":1550251734.0,
            "monitored":"yes",
            "service":"ntp",
            "lastChanged":1550251734.4790000916,
            "pid":"8478",
            "hostname":"leaf01",
            "enabled":"yes",
            "vrf":"mgmt",
            "active":"yes"
        },
        {
            "status":"ok",
            "uptime":1550251704.0,
            "monitored":"no",
            "service":"ssh",
            "lastChanged":1550251704.0929999352,
            "pid":"2106",
            "hostname":"leaf01",
            "enabled":"yes",
            "vrf":"default",
            "active":"yes"
        },
        {
            "status":"ok",
            "uptime":1550251736.0,
            "monitored":"yes",
            "service":"lldpd",
            "lastChanged":1550251736.5160000324,
            "pid":"2651",
            "hostname":"leaf01",
            "enabled":"yes",
            "vrf":"default",
            "active":"yes"
        },
        {
            "status":"ok",
            "uptime":1550251704.0,
            "monitored":"yes",
            "service":"bgpd",
            "lastChanged":1550251704.1040000916,
            "pid":"2872",
            "hostname":"leaf01",
            "enabled":"yes",
            "vrf":"default",
            "active":"yes"
        },
        {
            "status":"ok",
            "uptime":1550251704.0,
            "monitored":"no",
            "service":"neighmgrd",
            "lastChanged":1550251704.0969998837,
            "pid":"1986",
            "hostname":"leaf01",
            "enabled":"yes",
            "vrf":"default",
            "active":"yes"
        },
...
```

If you want to view the service information for a given device, simply use the *hostname* variable when running 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 `bgpd`, `lldpd`, and `clagd`.

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

```
Hostname          Service              PID   VRF             Enabled
Active Monitored Status           Uptime                    Last
Changed
----------------- -------------------- ----- --------------- -------
------ --------- ---------------- -------------------------
-------------------------
exit01            ntp                  8478  mgmt            yes
yes    yes       ok               1d:6h:52m:41s             Fri Feb
15 17:28:54 2019
exit02            ntp                  8497  mgmt            yes
yes    yes       ok               1d:6h:52m:36s             Fri Feb
15 17:28:59 2019
firewall01        ntp                  n/a   default         yes
yes    yes       ok               1d:6h:53m:4s              Fri Feb
15 17:28:31 2019
hostd-11          ntp                  n/a   default         yes
yes    yes       ok               1d:6h:52m:46s             Fri Feb
15 17:28:49 2019
hostd-21          ntp                  n/a   default         yes
yes    yes       ok               1d:6h:52m:37s             Fri Feb
15 17:28:58 2019
hosts-11          ntp                  n/a   default         yes
yes    yes       ok               1d:6h:52m:28s             Fri Feb
15 17:29:07 2019
hosts-13          ntp                  n/a   default         yes
yes    yes       ok               1d:6h:52m:19s             Fri Feb
15 17:29:16 2019
hosts-21          ntp                  n/a   default         yes
yes    yes       ok               1d:6h:52m:14s             Fri Feb
15 17:29:21 2019
hosts-23          ntp                  n/a   default         yes
yes    yes       ok               1d:6h:52m:4s              Fri Feb
15 17:29:31 2019
noc-pr            ntp                  2148  default         yes
yes    yes       ok               1d:6h:53m:43s             Fri Feb
15 17:27:52 2019
noc-se            ntp                  2148  default         yes
yes    yes       ok               1d:6h:53m:38s             Fri Feb
15 17:27:57 2019
spine01           ntp                  8414  mgmt            yes
yes    yes       ok               1d:6h:53m:30s             Fri Feb
15 17:28:05 2019
spine02           ntp                  8419  mgmt            yes
yes    yes       ok               1d:6h:53m:27s             Fri Feb
15 17:28:08 2019
spine03           ntp                  8443  mgmt            yes
yes    yes       ok               1d:6h:53m:22s             Fri Feb
15 17:28:13 2019
leaf01             ntp                  8765  mgmt            yes
yes    yes       ok               1d:6h:52m:52s             Fri Feb
15 17:28:43 2019
```

```
leaf02             ntp                  8737  mgmt            yes
yes    yes       ok               1d:6h:52m:46s             Fri Feb
15 17:28:49 2019
leaf11            ntp                  9305  mgmt            yes
yes    yes       ok               1d:6h:49m:22s             Fri Feb
15 17:32:13 2019
leaf12            ntp                  9339  mgmt            yes
yes    yes       ok               1d:6h:49m:9s              Fri Feb
15 17:32:26 2019
leaf21            ntp                  9367  mgmt            yes
yes    yes       ok               1d:6h:49m:5s              Fri Feb
15 17:32:30 2019
leaf22            ntp                  9403  mgmt            yes
yes    yes       ok               1d:6h:52m:57s             Fri Feb
15 17:28:38 2019
```

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                 2872  default         yes
yes    yes       ok               1d:6h:54m:37s             Fri Feb
15 17:28:24 2019
exit02            bgpd                 2867  default         yes
yes    yes       ok               1d:6h:54m:33s             Fri Feb
15 17:28:28 2019
firewall01        bgpd                 21766 default         yes
yes    yes       ok               1d:6h:54m:54s             Fri Feb
15 17:28:07 2019
spine01           bgpd                 2953  default         yes
yes    yes       ok               1d:6h:55m:27s             Fri Feb
15 17:27:34 2019
spine02           bgpd                 2948  default         yes
yes    yes       ok               1d:6h:55m:23s             Fri Feb
15 17:27:38 2019
spine03           bgpd                 2953  default         yes
yes    yes       ok               1d:6h:55m:18s             Fri Feb
15 17:27:43 2019
leaf01            bgpd                 3221  default         yes
yes    yes       ok               1d:6h:54m:48s             Fri Feb
15 17:28:13 2019
leaf02            bgpd                 3177  default         yes
yes    yes       ok               1d:6h:54m:42s             Fri Feb
15 17:28:19 2019
```

```
leaf11            bgpd                 3521  default         yes
yes    yes       ok               1d:6h:51m:18s            Fri Feb
15 17:31:43 2019
leaf12            bgpd                 3527  default         yes
yes    yes       ok               1d:6h:51m:6s             Fri Feb
15 17:31:55 2019
leaf21            bgpd                 3512  default         yes
yes    yes       ok               1d:6h:51m:1s             Fri Feb
15 17:32:00 2019
leaf22            bgpd                 3536  default         yes
yes    yes       ok               1d:6h:54m:54s            Fri Feb
15 17:28:07 2019
```

View Events Related to a Given Service

To view changes over a given time period, use the `netq show events` command. For more detailed information about events, refer to Monitor Events.

In this example, we want to view changes to the bgpd service in the last 48 hours.

```
cumulus@switch:/$ netq show events type bgp between now and 48h
Matching events records:
Hostname          Message Type Severity
Message                            Timestamp
----------------- ------------ --------
----------------------------------- -------------------------
leaf01            bgp          info     BGP session with peer spine-1
swp3. 1d:6h:55m:37s
                                        3 vrf DataVrf1081 state
changed fro
                                        m failed to Established
leaf01            bgp          info     BGP session with peer spine-2
swp4. 1d:6h:55m:37s
                                        3 vrf DataVrf1081 state
changed fro
                                        m failed to Established
leaf01            bgp          info     BGP session with peer spine-3
swp5. 1d:6h:55m:37s
                                        3 vrf DataVrf1081 state
changed fro
                                        m failed to Established
leaf01            bgp          info     BGP session with peer spine-1
swp3. 1d:6h:55m:37s
                                        2 vrf DataVrf1080 state
changed fro
                                        m failed to Established
leaf01            bgp          info     BGP session with peer spine-3
swp5. 1d:6h:55m:37s
                                        2 vrf DataVrf1080 state
changed fro
```

```
                                        m failed to Established
leaf01            bgp          info     BGP session with peer spine-2
swp4. 1d:6h:55m:37s
                                        2 vrf DataVrf1080 state
changed fro
                                        m failed to Established
leaf01            bgp          info     BGP session with peer spine-3
swp5. 1d:6h:55m:37s
                                        4 vrf DataVrf1082 state
changed fro
                                        m failed to Established
```

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 events [level info|level error|level
warning|level critical|level debug] type interfaces-physical [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.

```
exit01            eth0                      up         1G
off     RJ45      n/a                  n/a              Thu Feb  7 18:
31:52 2019
exit01            swp1                      up         10G
off     RJ45      n/a                  n/a              Thu Feb  7 18:
31:52 2019
exit01            swp2                      up         10G
off     RJ45      n/a                  n/a              Thu Feb  7 18:
31:52 2019
exit01            swp3                      up         10G
off     RJ45      n/a                  n/a              Thu Feb  7 18:
31:52 2019
exit01            swp4                      up         10G
off     RJ45      n/a                  n/a              Thu Feb  7 18:
31:52 2019
exit01            swp5                      up         10G
off     RJ45      n/a                  n/a              Thu Feb  7 18:
31:52 2019
exit01            swp6                      up         10G
off     RJ45      n/a                  n/a              Thu Feb  7 18:
31:52 2019
exit01            swp7                      up         10G
off     RJ45      n/a                  n/a              Thu Feb  7 18:
31:52 2019
exit02            eth0                      up         1G
off     RJ45      n/a                  n/a              Thu Feb  7 18:
31:57 2019
exit02            swp1                      up         10G
off     RJ45      n/a                  n/a              Thu Feb  7 18:
31:57 2019
exit02            swp2                      up         10G
off     RJ45      n/a                  n/a              Thu Feb  7 18:
31:57 2019
exit02            swp3                      up         10G
off     RJ45      n/a                  n/a              Thu Feb  7 18:
31:57 2019
exit02            swp4                      up         10G
off     RJ45      n/a                  n/a              Thu Feb  7 18:
31:57 2019
exit02            swp5                      up         10G
off     RJ45      n/a                  n/a              Thu Feb  7 18:
31:57 2019
exit02            swp6                      up         10G
off     RJ45      n/a                  n/a              Thu Feb  7 18:
31:57 2019
exit02            swp7                      up         10G
off     RJ45      n/a                  n/a              Thu Feb  7 18:
31:57 2019
firewall01        eth0                      up         Unknown
off     empty     n/a                  n/a              Thu Feb  7 18:
31:25 2019
```

```
firewall01        swp1                      up         10G
off     empty     n/a                  n/a              Thu Feb  7 18:
31:25 2019
firewall01        swp2                      up         10G
off     empty     n/a                  n/a              Thu Feb  7 18:
31:25 2019
firewall01        swp3                      up         10G
off     empty     n/a                  n/a              Thu Feb  7 18:
31:25 2019
firewall01        swp4                      up         10G
off     empty     n/a                  n/a              Thu Feb  7 18:
31:25 2019
firewall02        eth0                      up         n/a        n
/a     empty     n/a                  n/a              Thu Feb  7 18:
31:30 2019
firewall02        swp1                      up         n/a        n
/a     empty     n/a                  n/a              Thu Feb  7 18:
31:30 2019
firewall02        swp2                      up         n/a        n
/a     empty     n/a                  n/a              Thu Feb  7 18:
31:30 2019
firewall02        swp3                      up         n/a        n
/a     empty     n/a                  n/a              Thu Feb  7 18:
31:30 2019
firewall02        swp4                      up         n/a        n
/a     empty     n/a                  n/a              Thu Feb  7 18:
31:30 2019
server11          eth0                      up         Unknown
off     empty     n/a                  n/a              Thu Feb  7 18:
31:42 2019
server11          swp1                      up         10G
off     empty     n/a                  n/a              Thu Feb  7 18:
31:42 2019
server11          swp2                      up         10G
off     empty     n/a                  n/a              Thu Feb  7 18:
31:42 2019
server11          swp3                      up         10G
off     empty     n/a                  n/a              Thu Feb  7 18:
31:42 2019
server11          swp4                      up         10G
off     empty     n/a                  n/a              Thu Feb  7 18:
31:42 2019
server12          eth0                      up         n/a        n
/a     empty     n/a                  n/a              Thu Feb  7 18:
31:47 2019

...
```

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    Thu Feb  7 22:49:37 2019
leaf02            swp2                      SFP
Mellanox                MC2609130-003    MT1507VS05177
1000Base-CX,Copp Copper pigtail   3m     Thu Feb  7 22:49:37 2019

er Passive,Twin

Axial Pair (TW)
leaf02            swp47                     QSFP+
CISCO                   AFBR-7IER05Z-CS1 AVE1823402U               n
/a              n/a              5m     Thu Feb  7 22:49:37 2019
leaf02            swp48                     QSFP28    TE
Connectivity      2231368-1        15250052                 100G
Base-CR4 or n/a                3m      Thu Feb  7 22:49:37 2019

25G Base-CR CA-L

,40G Base-CR4
leaf02            swp49                     SFP
OEM                     SFP-10GB-LR      ACSLR130408
10G Base-LR      LC               10km,  Thu Feb  7 22:49:37 2019

10000m
leaf02            swp50                     SFP
JDSU                    PLRXPLSCS4322N   CG03UF45M
10G Base-SR,Mult LC               80m,   Thu Feb  7 22:49:37 2019

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      Thu Feb  7 22:49:37 2019

er Passive,Twin

Axial Pair (TW)
leaf02            swp52                     SFP       FINISAR
CORP.           FCLF8522P2BTL    PTN1VH2                  1000Base-
T       RJ45             100m   Thu Feb  7 22:49:37 2019
```

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              Thu Feb  7 22:49:37 2019
leaf01             swp52     down  Unknown    on      empty     n
/a              n/a              Thu Feb  7 22:49:37 2019
```

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 *leaf01* switch that have attached cables.

```
cumulus@switch:~$ netq leaf01 show interfaces physical plugged
Matching cables records:
Hostname          Interface                 State      Speed
AutoNeg Module    Vendor                Part No         Last Changed
----------------- ------------------------- ---------- ----------
------- --------- -------------------- ----------------
-------------------------
leaf01            eth0                      up         1G
on      RJ45      n/a                   n/a             Thu Feb  7 22:
49:37 2019
leaf01            swp1                      up         10G
off     SFP       Amphenol              610640005       Thu Feb  7 22:
49:37 2019
leaf01            swp2                      up         10G
off     SFP       Amphenol              610640005       Thu Feb  7 22:
49:37 2019
leaf01            swp3                      down       10G
off     SFP       Mellanox              MC3309130-001   Thu Feb  7 22:
49:37 2019
leaf01            swp33                     down       10G
off     SFP       OEM                   SFP-H10GB-CU1M  Thu Feb  7 22:
49:37 2019
leaf01            swp34                     down       10G
off     SFP       Amphenol              571540007       Thu Feb  7 22:
49:37 2019
leaf01            swp35                     down       10G
off     SFP       Amphenol              571540007       Thu Feb  7 22:
49:37 2019
leaf01            swp36                     down       10G
off     SFP       OEM                   SFP-H10GB-CU1M  Thu Feb  7 22:
49:37 2019
leaf01            swp37                     down       10G
off     SFP       OEM                   SFP-H10GB-CU1M  Thu Feb  7 22:
49:37 2019
leaf01            swp38                     down       10G
off     SFP       OEM                   SFP-H10GB-CU1M  Thu Feb  7 22:
49:37 2019
leaf01            swp39                     down       10G
off     SFP       Amphenol              571540007       Thu Feb  7 22:
49:37 2019
leaf01            swp40                     down       10G
off     SFP       Amphenol              571540007       Thu Feb  7 22:
49:37 2019
leaf01            swp49                     up         40G
off     QSFP+     Amphenol              624410001       Thu Feb  7 22:
49:37 2019
leaf01            swp5                      down       10G
off     SFP       Amphenol              571540007       Thu Feb  7 22:
49:37 2019
leaf01            swp50                     down       40G
off     QSFP+     Amphenol              624410001       Thu Feb  7 22:
49:37 2019
```

```
leaf01            swp51                     down       40G
off     QSFP+     Amphenol             603020003        Thu Feb  7 22:
49:37 2019
leaf01            swp52                     up         40G
off     QSFP+     Amphenol             603020003        Thu Feb  7 22:
49:37 2019
leaf01            swp54                     down       40G
off     QSFP+     Amphenol             624410002        Thu Feb  7 22:
49:37 2019
```

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 leaf01 show interfaces physical vendor OEM
Matching cables records:
Hostname          Interface                 State      Speed
AutoNeg Module    Vendor               Part No          Last Changed
----------------- ------------------------- ---------- ----------
------- --------- -------------------- ----------------
-------------------------
leaf01            swp33                     down       10G
off     SFP       OEM                  SFP-H10GB-CU1M   Thu Feb  7 22:
49:37 2019
leaf01            swp36                     down       10G
off     SFP       OEM                  SFP-H10GB-CU1M   Thu Feb  7 22:
49:37 2019
leaf01            swp37                     down       10G
off     SFP       OEM                  SFP-H10GB-CU1M   Thu Feb  7 22:
49:37 2019
leaf01            swp38                     down       10G
off     SFP       OEM                  SFP-H10GB-CU1M   Thu Feb  7 22:
49:37 2019
```

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   Thu Feb  7 18:31:20 2019
leaf02            swp52                     up
1G            off     QSFP+     OEM                     QSFP-H40G-
CU1M   Thu Feb  7 18:31:20 2019
```

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 events type interfaces-physical 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  Thu Feb  7 18:
34:20 2019
leaf01            swp2                      up           10G
off     SFP       OEM                  SFP-10GB-LR      Thu Feb  7 18:
34:20 2019
leaf01            swp47                        up
10G           off     SFP       JDSU
PLRXPLSCS4322N   Thu Feb  7 18:34:20 2019
leaf01              swp48                     up
40G           off     QSFP+     Mellanox              MC2210130-
002    Thu Feb  7 18:34:20 2019
leaf01              swp49                     down
10G           off     empty     n/a                   n
/a              Thu Feb  7 18:34:20 2019
```

```
leaf01              swp50                     up
1G          off      SFP       FINISAR CORP.         FCLF8522P2BTL
Thu Feb  7 18:34:20 2019
leaf01             swp51                      up
1G            off      SFP       FINISAR CORP.
FTLF1318P3BTL    Thu Feb  7 18:34:20 2019
leaf01            swp52                       down
1G            off      SFP       CISCO-AGILENT         QFBR-
5766LP      Thu Feb  7 18:34:20 2019
leaf02            swp1                        up
1G            on       RJ45      n/a                   n
/a                Thu Feb  7 18:34:20 2019
leaf02            swp2                        up
10G           off      SFP       Mellanox              MC2609130-
003    Thu Feb  7 18:34:20 2019
leaf02            swp47                         up
10G          off      QSFP+     CISCO                 AFBR-7IER05Z-CS1
Thu Feb  7 18:34:20 2019
leaf02            swp48                         up
10G           off      QSFP+     Mellanox              MC2609130-
003    Thu Feb  7 18:34:20 2019
leaf02            swp49                       up
10G           off      SFP       FIBERSTORE            SFP-10GLR-
31     Thu Feb  7 18:34:20 2019
leaf02            swp50                       up
1G            off      SFP       OEM                   SFP-GLC-
T        Thu Feb  7 18:34:20 2019
leaf02            swp51                        up
10G           off      SFP       Mellanox              MC2609130-
003    Thu Feb  7 18:34:20 2019
leaf02            swp52                       up
1G            off      SFP       FINISAR CORP.
FCLF8522P2BTL    Thu Feb  7 18:34:20 2019
leaf03            swp1                        up
10G           off      SFP       Mellanox             MC2609130-003
Thu Feb  7 18:34:20 2019
leaf03            swp2                           up
10G          off      SFP       Mellanox               MC3309130-
001    Thu Feb  7 18:34:20 2019
leaf03            swp47                        up          10G
off      SFP       CISCO-AVAGO          AFBR-7IER05Z-CS1 Thu Feb  7
18:34:20 2019
leaf03            swp48                       up
10G           off      SFP       Mellanox              MC3309130-
001    Thu Feb  7 18:34:20 2019
leaf03            swp49                        down
1G            off      SFP       FINISAR CORP.
FCLF8520P2BTL    Thu Feb  7 18:34:20 2019
leaf03            swp50                       up          1G
off      SFP       FINISAR CORP.        FCLF8522P2BTL    Thu Feb  7
18:34:20 2019
```

```
leaf03               swp51                       up
10G          off     QSFP+    Mellanox             MC2609130-003
Thu Feb  7 18:34:20 2019
...
oob-mgmt-server   swp1                      up         1G
off     RJ45      n/a                  n/a              Thu Feb  7 18:
34:20 2019
oob-mgmt-server   swp2                      up         1G
off     RJ45      n/a                  n/a              Thu Feb  7 18:
34:20 2019

cumulus@switch:~$ netq show events interfaces-physical 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  Thu Feb  7 18:
34:20 2019
leaf01            swp2                      up           10G
off     SFP       OEM                  SFP-10GB-LR      Thu Feb  7 18:
34:20 2019
leaf01            swp47                        up
10G           off     SFP       JDSU
PLRXPLSCS4322N   Thu Feb  7 18:34:20 2019
leaf01              swp48                     up
40G           off     QSFP+     Mellanox              MC2210130-
002    Thu Feb  7 18:34:20 2019
leaf01              swp49                     down
10G           off     empty     n/a                   n
/a              Thu Feb  7 18:34:20 2019
leaf01              swp50                     up
1G           off     SFP       FINISAR CORP.        FCLF8522P2BTL
Thu Feb  7 18:34:20 2019
leaf01             swp51                     up
1G            off     SFP       FINISAR CORP.
FTLF1318P3BTL    Thu Feb  7 18:34:20 2019
leaf01            swp52                     down
1G            off     SFP       CISCO-AGILENT         QFBR-
5766LP      Thu Feb  7 18:34:20 2019
...

cumulus@switch:~$ netq show events type interfaces-physical 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            Message
```

```
----------------- ------------------------- -----------------
------------------------- -----------------------------------
...
leaf03            swp1                      oob-mgmt-switch
swp7
leaf03
swp2
Peer port unknown
leaf03            swp47                     leaf04
swp47
leaf03            swp48                     leaf04
swp48                      State mismatch (up, down)
leaf03            swp49                     leaf04
swp49
leaf03            swp50                     leaf04
swp50
leaf03
swp52
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            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` command. 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
```

```
Matching cables records:
Hostname          Interface                 Peer Hostname     Peer
Interface            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...

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 events [level info|level error|level
warning|level critical|level debug] type lldp [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
----------------- ------------------------- -----------------
------------------------- -------------------------
exit01            swp1                      edge01
swp5                      Thu Feb  7 18:31:53 2019
exit01            swp2                      edge02
swp5                      Thu Feb  7 18:31:53 2019
exit01            swp3                      spine01
swp9                      Thu Feb  7 18:31:53 2019
exit01            swp4                      spine02
swp9                      Thu Feb  7 18:31:53 2019
exit01            swp5                      spine03
swp9                      Thu Feb  7 18:31:53 2019
exit01            swp6                      firewall01        mac:00:
02:00:00:00:11     Thu Feb  7 18:31:53 2019
exit01            swp7                      firewall02
swp3                      Thu Feb  7 18:31:53 2019
```

```
exit02            swp1                      edge01
swp6                      Thu Feb  7 18:31:49 2019
exit02            swp2                      edge02
swp6                      Thu Feb  7 18:31:49 2019
exit02            swp3                      spine01
swp10                     Thu Feb  7 18:31:49 2019
exit02            swp4                      spine02
swp10                     Thu Feb  7 18:31:49 2019
exit02            swp5                      spine03
swp10                     Thu Feb  7 18:31:49 2019
exit02            swp6                      firewall01        mac:00:
02:00:00:00:12     Thu Feb  7 18:31:49 2019
exit02            swp7                      firewall02
swp4                      Thu Feb  7 18:31:49 2019
firewall01        swp1                      edge01
swp14                     Thu Feb  7 18:31:26 2019
firewall01        swp2                      edge02
swp14                     Thu Feb  7 18:31:26 2019
firewall01        swp3                      exit01
swp6                      Thu Feb  7 18:31:26 2019
firewall01        swp4                      exit02
swp6                      Thu Feb  7 18:31:26 2019
firewall02        swp1                      edge01
swp15                     Thu Feb  7 18:31:31 2019
firewall02        swp2                      edge02
swp15                     Thu Feb  7 18:31:31 2019
firewall02        swp3                      exit01
swp7                      Thu Feb  7 18:31:31 2019
firewall02        swp4                      exit02
swp7                      Thu Feb  7 18:31:31 2019
server11          swp1                      leaf01
swp7                      Thu Feb  7 18:31:43 2019
server11          swp2                      leaf02
swp7                      Thu Feb  7 18:31:43 2019
server11          swp3                      edge01
swp16                     Thu Feb  7 18:31:43 2019
server11          swp4                      edge02
swp16                     Thu Feb  7 18:31:43 2019
server12          swp1                      leaf01
swp8                      Thu Feb  7 18:31:47 2019
server12          swp2                      leaf02
swp8                      Thu Feb  7 18:31:47 2019
```

Monitor Interface Health

Interface (link) health can be monitored using the `netq show interfaces` command. You can view status of the links, whether they are operating over a VRF interface, the MTU of the link, and so forth. Using the *hostname* keyword enables you to view only the interfaces for a given device. View changes to interfaces using the `netq show events` command.

The syntax for these commands is:

```
netq [<hostname>] show interfaces [type bond|type bridge|type
eth|type loopback|type macvlan|type swp|type vlan|type vrf|type
vxlan] [state <remote-interface-state>] [around <text-time>] [json]
netq <hostname> show interfaces [type bond|type bridge|type eth|type
loopback|type macvlan|type swp|type vlan|type vrf|type vxlan] [state
<remote-interface-state>] [around <text-time>] [count] [json]
netq [<hostname>] show events [level info | level error | level
warning | level critical | level debug] type interfaces [between
<text-time> and <text-endtime>] [json]
```

View Status for All Interfaces

Viewing the status of all interfaces at once can be helpful when you are trying to compare configuration or status of a set of links, or generally when changes have been made.

This example shows all interfaces network-wide.

```
cumulus@switch:~$ netq show interfaces
Matching link records:
Hostname          Interface                 Type             State
  VRF             Details                                    Last Changed
----------------- ------------------------- ----------------
---------- --------------- -----------------------------------
-------------------------
exit01            bridge                    bridge           up
  default         , Root bridge:  exit01,             Mon Apr 29 20:
57:59 2019

                  Root port: , Members:  vxlan4001,

                  bridge,
exit01            eth0                      eth              up
  mgmt            MTU: 1500                                  Mon Apr 29 20:
57:59 2019
exit01            lo                        loopback         up
  default         MTU: 65536                                 Mon Apr 29 20:
57:58 2019
exit01            mgmt                      vrf              up
                  table: 1001, MTU: 65536,                   Mon Apr 29 20:
57:58 2019

                  Members:  mgmt,  eth0,
exit01            swp1                      swp              down
  default         VLANs: , PVID: 0 MTU: 1500                 Mon Apr 29 20:
57:59 2019
```

```
exit01            swp44                     swp              up
  vrf1            VLANs: ,                            Mon Apr 29 20:
57:58 2019

                  PVID: 0 MTU: 1500 LLDP: internet:sw

                  p1
exit01            swp45                     swp              down
  default         VLANs: , PVID: 0 MTU: 1500          Mon Apr 29 20:
57:59 2019
exit01            swp46                     swp              down
  default         VLANs: , PVID: 0 MTU: 1500          Mon Apr 29 20:
57:59 2019
exit01            swp47                     swp              down
  default         VLANs: , PVID: 0 MTU: 1500          Mon Apr 29 20:
57:59 2019

...

leaf01            bond01                    bond
up         default         Slave:swp1 LLDP: server01:eth1      Mon
Apr 29 20:57:59 2019
leaf01            bond02                    bond
up         default         Slave:swp2 LLDP: server02:eth1      Mon
Apr 29 20:57:59 2019
leaf01            bridge                    bridge
up         default         , Root bridge:  leaf01,             Mon
Apr 29 20:57:59 2019

Root port: , Members:  vxlan4001,

bond02,  vni24,  vni13,  bond01,

bridge,  peerlink,
leaf01            eth0                      eth
up         mgmt            MTU: 1500                           Mon
Apr 29 20:58:00 2019
leaf01            lo                        loopback
up         default         MTU: 65536                          Mon
Apr 29 20:57:59 2019
leaf01            mgmt                      vrf
up                         table: 1001, MTU: 65536,            Mon
Apr 29 20:57:59 2019

Members:  mgmt,  eth0,
leaf01            peerlink                  bond
up         default         Slave:swp50 LLDP: leaf02:swp49 LLDP Mon
Apr 29 20:58:00 2019

: leaf02:swp50
...
```

View Interface Status for a Given Device

If you are interested in only a the interfaces on a specific device, you can view only those.

This example shows all interfaces on the *spine01* device.

```
cumulus@switch:~$ netq spine01 show interfaces
Matching link records:
Hostname          Interface                 Type             State
  VRF             Details                                   Last Changed
----------------- ------------------------- ----------------
---------- --------------- -----------------------------------
-------------------------
spine01           eth0                      eth              up
  mgmt            MTU: 1500                                 Mon Apr 29 21:
12:47 2019
spine01           lo                        loopback         up
  default         MTU: 65536                                Mon Apr 29 21:
12:47 2019
spine01           mgmt                      vrf              up
                  table: 1001, MTU: 65536,                  Mon Apr 29 21:
12:46 2019

                  Members:  mgmt,  eth0,
spine01           swp1                      swp              up
  default         VLANs: ,                                  Mon Apr 29 21:
12:47 2019

                  PVID: 0 MTU: 9216 LLDP: leaf01:swp5

                  1
spine01           swp2                      swp              up
  default         VLANs: ,                                  Mon Apr 29 21:
12:47 2019

                  PVID: 0 MTU: 9216 LLDP: leaf02:swp5

                  1
spine01           swp29                     swp              up
  default         VLANs: ,                                  Mon Apr 29 21:
12:47 2019

                  PVID: 0 MTU: 9216 LLDP: exit02:swp5

                  1
spine01           swp3                      swp              up
  default         VLANs: ,                                  Mon Apr 29 21:
12:46 2019

                  PVID: 0 MTU: 9216 LLDP: leaf03:swp5
                  1
spine01           swp30                     swp              up
  default         VLANs: ,                            Mon Apr 29 21:
12:47 2019

                  PVID: 0 MTU: 9216 LLDP: exit01:swp5

                  1
spine01           swp31                     swp              up
  default         VLANs: ,                            Mon Apr 29 21:
12:46 2019

                  PVID: 0 MTU: 9216 LLDP: spine02:swp

                  31
spine01           swp32                     swp              up
  default         VLANs: ,                            Mon Apr 29 21:
12:46 2019

                  PVID: 0 MTU: 9216 LLDP: spine02:swp

                  32
spine01           swp4                      swp              up
  default         VLANs: ,                            Mon Apr 29 21:
12:47 2019

                  PVID: 0 MTU: 9216 LLDP: leaf04:swp5

                  1
```

View All Interfaces of a Given Type

It can be can be useful to see the status of a particular type of interface.

This example shows all bond interfaces that are down, and then those that are up.

```
cumulus@switch:~$ netq show interfaces type bond state down
No matching link records found

cumulus@switch:~$ netq show interfaces type bond state up
Matching link records:
Hostname          Interface                 Type             State
  VRF             Details                             Last Changed
----------------- ------------------------- ----------------
---------- -------------- -----------------------------------
-------------------------
leaf01            bond01                    bond             up
  default         Slave:swp1 LLDP: server01:eth1      Mon Apr 29 21:
19:07 2019
```

```
leaf01            bond02                    bond             up
  default         Slave:swp2 LLDP: server02:eth1      Mon Apr 29 21:
19:07 2019
leaf01            peerlink                  bond             up
  default         Slave:swp50 LLDP: leaf02:swp49 LLDP Mon Apr 29 21:
19:07 2019

                  : leaf02:swp50
leaf02            bond01                    bond             up
  default         Slave:swp1 LLDP: server01:eth2      Mon Apr 29 21:
19:07 2019
leaf02            bond02                    bond             up
  default         Slave:swp2 LLDP: server02:eth2      Mon Apr 29 21:
19:07 2019
leaf02            peerlink                  bond             up
  default         Slave:swp50 LLDP: leaf01:swp49 LLDP Mon Apr 29 21:
19:07 2019

                  : leaf01:swp50
leaf03            bond03                    bond             up
  default         Slave:swp1 LLDP: server03:eth1      Mon Apr 29 21:
19:07 2019
leaf03            bond04                    bond             up
  default         Slave:swp2 LLDP: server04:eth1      Mon Apr 29 21:
19:07 2019
leaf03            peerlink                  bond             up
  default         Slave:swp50 LLDP: leaf04:swp49 LLDP Mon Apr 29 21:
19:07 2019

                  : leaf04:swp50
leaf04            bond03                    bond             up
  default         Slave:swp1 LLDP: server03:eth2      Mon Apr 29 21:
19:07 2019
leaf04            bond04                    bond             up
  default         Slave:swp2 LLDP: server04:eth2      Mon Apr 29 21:
19:07 2019
leaf04            peerlink                  bond             up
  default         Slave:swp50 LLDP: leaf03:swp49 LLDP Mon Apr 29 21:
19:07 2019

                  : leaf03:swp50
server01          bond0                     bond             up
  default         Slave:bond0 LLDP: leaf02:swp1       Mon Apr 29 21:
19:07 2019
server02          bond0                     bond             up
  default         Slave:bond0 LLDP: leaf02:swp2       Mon Apr 29 21:
19:07 2019
server03          bond0                     bond             up
  default         Slave:bond0 LLDP: leaf04:swp1       Mon Apr 29 21:
19:07 2019
```

```
server04          bond0                     bond             up
  default         Slave:bond0 LLDP: leaf04:swp2       Mon Apr 29 21:
19:07 2019
```

View the Total Number of Interfaces

For a quick view of the amount of interfaces currently operating on a device, use the *hostname* and *count* keywords together.

This example shows the count of interfaces on the *leaf03* switch.

```
cumulus@switch:~$ netq leaf03 show interfaces count
Count of matching link records: 28
```

View the Total Number of a Given Interface Type

It can be useful to see how many interfaces of a particular type you have on a device.

This example shows the count of swp interfaces are on the *leaf03* switch.

```
cumulus@switch:~$ netq leaf03 show interfaces type swp count
Count of matching link records: 11
```

View Changes to Interfaces

If you suspect that an interface is not working as expected, seeing a drop in performance or a large number of dropped messages for example, you can view changes that have been made to interfaces network-wide.

This example shows info level events for all interfaces in your network:

```
cumulus@switch:~$ netq show events level info type interfaces between
now and 30d
Matching events records:
Hostname          Message
Type             Severity         Message
Timestamp
----------------- ------------------------ ----------------
----------------------------------- -------------------------
server03          link                     info             HostName
server03 changed state fro 3d:12h:8m:28s
                                                            m down to
up Interface:eth2
server03          link                     info             HostName
server03 changed state fro 3d:12h:8m:28s
                                                            m down to
up Interface:eth1
```

```
server01          link                     info             HostName
server01 changed state fro 3d:12h:8m:30s
                                                            m down to

up Interface:eth2
server01          link                     info             HostName
server01 changed state fro 3d:12h:8m:30s
                                                            m down to

up Interface:eth1
server02          link                     info             HostName
server02 changed state fro 3d:12h:8m:34s
                                                            m down to

up Interface:eth2
...
```

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:
7
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            -                         -      -                 -
                    -        Rotten Agent
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
```

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 interfaces [type vlan] [state <remote-
interface-state>] [around <text-time>] [json]
netq <hostname> show interfaces [type vlan] [state <remote-interface-
state>] [around <text-time>] [count] [json]
netq [<hostname>] show events [level info | level error | level
warning | level critical | level debug] type vlan [between <text-
time> and <text-endtime>] [json]
netq show macs [<mac>] [vlan <1-4096>] [origin] [around <text-time>]
[json]
netq <hostname> show macs [<mac>] [vlan <1-4096>] [origin | count]
[around <text-time>] [json]
netq <hostname> show macs egress-port <egress-port> [<mac>] [vlan <1-
4096>] [origin] [around <text-time>] [json]
netq [<hostname>] show vlan [<1-4096>] [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 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
Thu Feb  7 18:31:38 2019
exit02            4001                      4001
Thu Feb  7 18:31:38 2019
leaf01            1,13,24,4001              13 24 4001
Thu Feb  7 18:31:38 2019
leaf02            1,13,24,4001              13 24 4001
Thu Feb  7 18:31:38 2019
leaf03            1,13,24,4001              13 24 4001
Thu Feb  7 18:31:38 2019
leaf04            1,13,24,4001              13 24 4001
Thu Feb  7 18:31:38 2019
```

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                         Fri Feb  8 00:
24:28 2019
exit02            vlan4001                  vlan             up
  vrf1            MTU:1500                         Fri Feb  8 00:
24:28 2019
leaf01            peerlink.
4094             vlan             up          default         MTU:
9000                            Fri Feb  8 00:24:28 2019
leaf01            vlan13                    vlan             up
  vrf1            MTU:1500                         Fri Feb  8 00:
24:28 2019
leaf01            vlan24                    vlan             up
  vrf1            MTU:1500                         Fri Feb  8 00:
24:28 2019
leaf01            vlan4001                  vlan             up
  vrf1            MTU:1500                         Fri Feb  8 00:
24:28 2019
leaf02            peerlink.
4094             vlan             up          default         MTU:
9000                            Fri Feb  8 00:24:28 2019
```

```
leaf02            vlan13                    vlan             up
  vrf1            MTU:1500                            Fri Feb  8 00:
24:28 2019
leaf02            vlan24                    vlan             up
  vrf1            MTU:1500                            Fri Feb  8 00:
24:28 2019
leaf02            vlan4001                  vlan             up
  vrf1            MTU:1500                            Fri Feb  8 00:
24:28 2019
leaf03            peerlink.
4094             vlan             up          default          MTU:
9000                            Fri Feb  8 00:24:28 2019
leaf03            vlan13                    vlan             up
  vrf1            MTU:1500                            Fri Feb  8 00:
24:28 2019
leaf03            vlan24                    vlan             up
  vrf1            MTU:1500                            Fri Feb  8 00:
24:28 2019
leaf03            vlan4001                  vlan             up
  vrf1            MTU:1500                            Fri Feb  8 00:
24:28 2019
leaf04            peerlink.
4094             vlan             up          default          MTU:
9000                            Fri Feb  8 00:24:28 2019
leaf04            vlan13                    vlan             up
  vrf1            MTU:1500                            Fri Feb  8 00:
24:28 2019
leaf04            vlan24                    vlan             up
  vrf1            MTU:1500                            Fri Feb  8 00:
24:28 2019
leaf04            vlan4001                  vlan             up
  vrf1            MTU:1500                            Fri Feb  8 00:
24:28 2019
```

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     Fri Feb  8 00:24:28 2019
no     00:03:00:11:11:01  13     leaf02            bond01:
server01      no     Fri Feb  8 00:24:28 2019
no     00:03:00:11:11:01  13     leaf03            vni13:
leaf01         yes    Fri Feb  8 00:24:28 2019
no     00:03:00:11:11:01  13     leaf04            vni13:
leaf01         yes    Fri Feb  8 00:24:28 2019
no     00:03:00:33:33:01  13     leaf01            vni13:
10.0.0.134     yes    Fri Feb  8 00:24:28 2019
no     00:03:00:33:33:01  13     leaf02            vni13:
10.0.0.134     yes    Fri Feb  8 00:24:28 2019
no     00:03:00:33:33:01  13     leaf03            bond03:
server03      no     Fri Feb  8 00:24:28 2019
no     00:03:00:33:33:01  13     leaf04            bond03:
server03      no     Fri Feb  8 00:24:28 2019
no     02:03:00:11:11:01  13     leaf01            bond01:
server01      no     Fri Feb  8 00:24:28 2019
no     02:03:00:11:11:01  13     leaf02            bond01:
server01      no     Fri Feb  8 00:24:28 2019
no     02:03:00:11:11:01  13     leaf03            vni13:
leaf01         yes    Fri Feb  8 00:24:28 2019
no     02:03:00:11:11:01  13     leaf04            vni13:
leaf01         yes    Fri Feb  8 00:24:28 2019
no     02:03:00:11:11:02  13     leaf01            bond01:
server01      no     Fri Feb  8 00:24:28 2019
no     02:03:00:11:11:02  13     leaf02            bond01:
server01      no     Fri Feb  8 00:24:28 2019
no     02:03:00:11:11:02  13     leaf03            vni13:
leaf01         yes    Fri Feb  8 00:24:28 2019
no     02:03:00:11:11:02  13     leaf04            vni13:
leaf01         yes    Fri Feb  8 00:24:28 2019
no     02:03:00:33:33:01  13     leaf01            vni13:
10.0.0.134     yes    Fri Feb  8 00:24:28 2019
no     02:03:00:33:33:01  13     leaf02            vni13:
10.0.0.134     yes    Fri Feb  8 00:24:28 2019
no     02:03:00:33:33:01  13     leaf03            bond03:
server03      no     Fri Feb  8 00:24:28 2019
no     02:03:00:33:33:01  13     leaf04            bond03:
server03      no     Fri Feb  8 00:24:28 2019
no     02:03:00:33:33:02  13     leaf01            vni13:
10.0.0.134     yes    Fri Feb  8 00:24:28 2019
no     02:03:00:33:33:02  13     leaf02            vni13:
10.0.0.134     yes    Fri Feb  8 00:24:28 2019
no     02:03:00:33:33:02  13     leaf03            bond03:
server03      no     Fri Feb  8 00:24:28 2019
no     02:03:00:33:33:02  13     leaf04            bond03:
server03      no     Fri Feb  8 00:24:28 2019
yes    44:38:39:00:00:
03  13     leaf01            bridge               no      Fri Feb  8
00:24:28 2019
```

```
yes    44:38:39:00:00:
15  13      leaf02            bridge               no     Fri Feb  8
00:24:28 2019
yes    44:38:39:00:00:
23  13      leaf03            bridge               no     Fri Feb  8
00:24:28 2019
yes    44:38:39:00:00:
5c  13      leaf04            bridge               no     Fri Feb  8
00:24:28 2019
yes    44:39:39:ff:00:
13  13      leaf01            bridge               no     Fri Feb  8
00:24:28 2019
yes    44:39:39:ff:00:
13  13      leaf02            bridge               no     Fri Feb  8
00:24:28 2019
yes    44:39:39:ff:00:
13  13      leaf03            bridge               no     Fri Feb  8
00:24:28 2019
yes    44:39:39:ff:00:
13  13      leaf04            bridge               no     Fri Feb  8
00:24:28 2019
```

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     Fri Feb  8
00:24:28 2019
yes    44:39:39:ff:00:
13  13      leaf03            bridge               no     Fri Feb  8
00:24:28 2019
```

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,             Fri Feb  8 00:28:09 2019

                  Mode: Private
leaf01            vlan24-
v0                 macvlan          up         vrf1            MAC:
44:39:39:ff:00:24,             Fri Feb  8 00:28:09 2019

                  Mode: Private
leaf02            vlan13-
v0                 macvlan          up         vrf1            MAC:
44:39:39:ff:00:13,             Fri Feb  8 00:28:09 2019

                  Mode: Private
leaf02            vlan24-
v0                 macvlan          up         vrf1            MAC:
44:39:39:ff:00:24,             Fri Feb  8 00:28:09 2019

                  Mode: Private
leaf03            vlan13-
v0                 macvlan          up         vrf1            MAC:
44:39:39:ff:00:13,             Fri Feb  8 00:28:09 2019

                  Mode: Private
leaf03            vlan24-
v0                 macvlan          up         vrf1            MAC:
44:39:39:ff:00:24,             Fri Feb  8 00:28:09 2019

                  Mode: Private
leaf04            vlan13-
v0                 macvlan          up         vrf1            MAC:
44:39:39:ff:00:13,             Fri Feb  8 00:28:09 2019

                  Mode: Private
leaf04            vlan24-
v0                 macvlan          up         vrf1            MAC:
44:39:39:ff:00:24,             Fri Feb  8 00:28:09 2019

                  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 events [level info|level error|level
warning|level critical|level debug] type clag [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 leaf11 and leaf12 (which happens to be down), edge01(P) and edge02, and leaf21(P) and leaf22.

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

s
----------------- ----------------- ------------------ ----------
------ ----- ----- -------------------------
leaf11                              44:38:39:ff:ff:01  down       n
/a    0     0     Thu Feb  7 18:30:49 2019
leaf12                              44:38:39:ff:ff:01  down
down   8     0     Thu Feb  7 18:30:53 2019
```

```
edge01(P)         edge02            00:01:01:10:00:01  up
up     25    25    Thu Feb  7 18:31:02 2019
edge02            edge01(P)         00:01:01:10:00:01  up
up     25    25    Thu Feb  7 18:31:15 2019
leaf21(P)         leaf22            44:38:39:ff:ff:02  up
up     8     8     Thu Feb  7 18:31:20 2019
leaf22            leaf21(P)         44:38:39:ff:ff:02  up
up     8     8     Thu Feb  7 18:31:30 2019
```

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
----------------- ----------------- ------------------ ----------
------ ----- ----- -------------------------
edge01(P)         edge02            00:01:01:10:00:
01  up          up      25    25    Thu Feb  7 18:31:30 2019
edge02            edge01(P)         00:01:01:10:00:
01  up          up      25    25    Thu Feb  7 18:31:30 2019
leaf11(P)         leaf12            44:38:39:ff:ff:
01  up          up      8     8     Thu Feb  7 18:31:30 2019
leaf12            leaf11(P)         44:38:39:ff:ff:
01  up          up      8     8     Thu Feb  7 18:31:30 2019
leaf21(P)         leaf22            44:38:39:ff:ff:
02  up          up      8     8     Thu Feb  7 18:31:30 2019
leaf22            leaf21(P)         44:38:39:ff:ff:
02  up          up      8     8     Thu Feb  7 18:31:30 2019
```

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 leaf22 show clag
Matching CLAG session records are:
Hostname          Peer              SysMac             State
Backup #Bond #Dual Last Changed

s
----------------- ----------------- ------------------ ----------
------ ----- ----- -------------------------
leaf22            leaf21(P)         44:38:39:ff:ff:02  up
up     8     8     Thu Feb  7 18:31:30 2019
```

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

```
cumulus@switch:~$ 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 NetQ Platform 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] [around <text-time>]
[json]
netq [<hostname>] show events [level info|level error|level
warning|level critical|level debug] type ntp [between <text-time> and
<text-endtime>] [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      services01.it.c   3       ntpq
exit01            yes      time.tritn.com    2       ntpq
exit02            yes      time.tritn.com    2       ntpq
internet          no       -                 16      ntpq
leaf01            yes      services01.it.c   2       ntpq
leaf02            yes      services01.it.c   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 any changes were made around then. The syntax for the show command is:

```
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 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>] [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, *server11*. 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 leaf01
vrf vrf1 pretty
Number of Paths: 4
Number of Paths with Errors: 0
Number of Paths with Warnings: 0
Path MTU: 9152
 leaf01 vni: 34 uplink-2 -- downlink-5 spine02 downlink-2 -- uplink-2
vni: 34 leaf12 hostbond4 -- swp2 server11
               uplink-2 -- downlink-5 spine02 downlink-1 -- uplink-2
vni: 34 leaf11 hostbond4 -- swp1 server11
 leaf01 vni: 34 uplink-1 -- downlink-5 spine01 downlink-2 -- uplink-1
vni: 34 leaf12 hostbond4 -- swp2 server11
               uplink-1 -- downlink-5 spine01 downlink-1 -- uplink-1
vni: 34 leaf11 hostbond4 -- swp1 server11
```

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
10.0.0.8 vrf vrf1 pretty
Number of Paths: 4
Number of Paths with Errors: 0
Number of Paths with Warnings: 0
Path MTU: 9152
 server11 swp1 -- swp5 <vlan1000> tor-1 <vlan1001> vni: 34 uplink-2
-- downlink-5 spine02 downlink-2 -- uplink-2 vni: 34 <vlan1001>
leaf12 hostbond4 -- swp2 server11
                                                           uplink-2
-- downlink-5 spine02 downlink-1 -- uplink-2 vni: 34 <vlan1001>
leaf11 hostbond4 -- swp1 server11
          swp1 -- swp5 <vlan1000> tor-1 <vlan1001> vni: 34 uplink-1
-- downlink-5 spine01 downlink-2 -- uplink-1 vni: 34 <vlan1001>
leaf12 hostbond4 -- swp2 server11
                                                           uplink-1
-- downlink-5 spine01 downlink-1 -- uplink-1 vni: 34 <vlan1001>
leaf11 hostbond4 -- swp1 server11
```

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
10.0.0.8 vrf vrf1 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
server11
swp1            1000
    2   leaf01          swp5            1000
vlan1000        vrf1            vlan1001        vrf1            vni:
34               uplink-2
    3   spine02         downlink-5
downlink-5      default         downlink-2
default                               downlink-2
    4   leaf12          uplink-2               vni: 34
vlan1001
vrf1
hostbond4       1001
    5   server11        swp2
--- --- --------------- --------------- ------ ---------------------
--------------- --------------- --------------- ---------------
--------------------- --------------- -------
2   1
server11
swp1            1000
    2   leaf01          swp5            1000
vlan1000        vrf1            vlan1001        vrf1            vni:
34               uplink-2
    3   spine02         downlink-5
downlink-5      default         downlink-1
default                               downlink-1
    4   leaf11          uplink-2               vni: 34
vlan1001
vrf1
hostbond4       1001
    5   server11        swp1
--- --- --------------- --------------- ------ ---------------------
--------------- --------------- --------------- ---------------
--------------------- --------------- -------
```

```
3   1
server11
swp1            1000
    2   leaf01          swp5            1000
vlan1000        vrf1            vlan1001        vrf1            vni:
34               uplink-1
    3   spine01         downlink-5
downlink-5      default         downlink-2
default                               downlink-2
    4   leaf12          uplink-1               vni: 34
vlan1001
vrf1
hostbond4       1001
    5   server11        swp2
--- --- --------------- --------------- ------ ---------------------
--------------- --------------- --------------- ---------------
--------------------- --------------- -------
4   1
server11
swp1            1000
    2   leaf01          swp5            1000
vlan1000        vrf1            vlan1001        vrf1            vni:
34               uplink-1
    3   spine01         downlink-5
downlink-5      default         downlink-1
default                               downlink-1
    4   leaf11          uplink-1               vni: 34
vlan1001
vrf1
hostbond4       1001
    5   server11        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>] [around <text-time>] [json]
netq <hostname> show ip neighbors [<remote-interface>] [<ipv4>|<ipv4>
vrf <vrf>|vrf <vrf>] [<mac>] [around <text-time>] [json]
netq [<hostname>] show ip neighbors [<remote-interface>]
[<ipv4>|<ipv4> vrf <vrf>|vrf <vrf>] [<mac>] [around <text-time>]
[count] [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] [around <text-time>] [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>] [around <text-time>] [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>] [around <text-time>]
[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] [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 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
- 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         Thu Feb  7 18:30:53 2019
10.0.0.12/32              leaf02            lo
default         Thu Feb  7 18:30:53 2019
10.0.0.13/32              leaf03            lo
default         Thu Feb  7 18:30:53 2019
10.0.0.14/32              leaf04            lo
default         Thu Feb  7 18:30:53 2019
10.0.0.21/32              spine01           lo
default         Thu Feb  7 18:30:53 2019
10.0.0.22/32              spine02           lo
default         Thu Feb  7 18:30:53 2019
10.0.0.254/32             oob-mgmt-server   eth0
default         Thu Feb  7 18:30:53 2019
172.16.1.1/24             leaf01            br0
default         Thu Feb  7 18:30:53 2019
172.16.1.101/24           server01          eth1
default         Thu Feb  7 18:30:53 2019
172.16.2.1/24             leaf02            br0
default         Thu Feb  7 18:30:53 2019
172.16.2.101/24           server02          eth2
default         Thu Feb  7 18:30:53 2019
172.16.3.1/24             leaf03            br0
default         Thu Feb  7 18:30:53 2019
172.16.3.101/24           server03          eth1
default         Thu Feb  7 18:30:53 2019
172.16.4.1/24             leaf04            br0
default         Thu Feb  7 18:30:53 2019
172.16.4.101/24           server04          eth2
default         Thu Feb  7 18:30:53 2019
```

```
172.17.0.1/16             oob-mgmt-server   docker0
default         Thu Feb  7 18:30:53 2019
192.168.0.11/24           leaf01            eth0
default         Thu Feb  7 18:30:53 2019
192.168.0.12/24           leaf02            eth0
default         Thu Feb  7 18:30:53 2019
192.168.0.13/24           leaf03            eth0
default         Thu Feb  7 18:30:53 2019
192.168.0.14/24           leaf04            eth0
default         Thu Feb  7 18:30:53 2019
192.168.0.21/24           spine01           eth0
default         Thu Feb  7 18:30:53 2019
192.168.0.22/24           spine02           eth0
default         Thu Feb  7 18:30:53 2019
192.168.0.254/24          oob-mgmt-server   eth1
default         Thu Feb  7 18:30:53 2019
192.168.0.31/24           server01          eth0
default         Thu Feb  7 18:30:53 2019
192.168.0.32/24           server02          eth0
default         Thu Feb  7 18:30:53 2019
192.168.0.33/24           server03          eth0
default         Thu Feb  7 18:30:53 2019
192.168.0.34/24           server04          eth0
default         Thu Feb  7 18:30:53 2019
```

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         Thu Feb
7 18:30:53 2019
fe80::203:ff:fe22:2202/64
server02          eth2                      default         Thu Feb
7 18:30:53 2019
fe80::203:ff:fe33:3301/64
server03          eth1                      default         Thu Feb
7 18:30:53 2019
fe80::203:ff:fe44:4402/64
server04          eth2                      default         Thu Feb
7 18:30:53 2019
fe80::4638:39ff:fe00:18/6
leaf02            br0                       default         Thu Feb
7 18:30:53 2019
```

```
fe80::4638:39ff:fe00:1b/6
leaf03            swp52                     default         Thu Feb
7 18:30:53 2019
fe80::4638:39ff:fe00:1c/6
spine02           swp3                      default         Thu Feb
7 18:30:53 2019
fe80::4638:39ff:fe00:23/6
leaf03            br0                       default         Thu Feb
7 18:30:53 2019
fe80::4638:39ff:fe00:24/6
leaf01            swp52                     default         Thu Feb
7 18:30:53 2019
fe80::4638:39ff:fe00:25/6
spine02           swp1                      default         Thu Feb
7 18:30:53 2019
fe80::4638:39ff:fe00:28/6
leaf02            swp51                     default         Thu Feb
7 18:30:53 2019
fe80::4638:39ff:fe00:29/6
spine01           swp2                      default         Thu Feb
7 18:30:53 2019
fe80::4638:39ff:fe00:2c/6
leaf04            br0                       default         Thu Feb
7 18:30:53 2019
fe80::4638:39ff:fe00:3/64
leaf01            br0                       default         Thu Feb
7 18:30:53 2019
fe80::4638:39ff:fe00:3b/6
leaf04            swp51                     default         Thu Feb
7 18:30:53 2019
fe80::4638:39ff:fe00:3c/6
spine01           swp4                      default         Thu Feb
7 18:30:53 2019
fe80::4638:39ff:fe00:46/6
leaf04            swp52                     default         Thu Feb
7 18:30:53 2019
fe80::4638:39ff:fe00:47/6
spine02           swp4                      default         Thu Feb
7 18:30:53 2019
fe80::4638:39ff:fe00:4f/6
leaf03            swp51                     default         Thu Feb
7 18:30:53 2019
fe80::4638:39ff:fe00:50/6
spine01           swp3                      default         Thu Feb
7 18:30:53 2019
fe80::4638:39ff:fe00:53/6
leaf01            swp51                     default         Thu Feb
7 18:30:53 2019
fe80::4638:39ff:fe00:54/6
spine01           swp1                      default         Thu Feb
7 18:30:53 2019
```

```
fe80::4638:39ff:fe00:57/6 oob-mgmt-
server   eth1                      default         Thu Feb  7 18:30:
53 2019
fe80::4638:39ff:fe00:5d/6
leaf02            swp52                     default         Thu Feb
7 18:30:53 2019
fe80::4638:39ff:fe00:5e/6
spine02           swp2                      default         Thu Feb
7 18:30:53 2019
fe80::5054:ff:fe77:c277/6 oob-mgmt-
server   eth0                      default         Thu Feb  7 18:30:
53 2019
fe80::a200:ff:fe00:11
/64  leaf01            eth0                      default         Thu
Feb  7 18:30:53 2019
fe80::a200:ff:fe00:12
/64  leaf02            eth0                      default         Thu
Feb  7 18:30:53 2019
fe80::a200:ff:fe00:13
/64  leaf03            eth0                      default         Thu
Feb  7 18:30:53 2019
fe80::a200:ff:fe00:14
/64  leaf04            eth0                      default         Thu
Feb  7 18:30:53 2019
fe80::a200:ff:fe00:21
/64  spine01           eth0                      default         Thu
Feb  7 18:30:53 2019
fe80::a200:ff:fe00:22
/64  spine02           eth0                      default         Thu
Feb  7 18:30:53 2019
fe80::a200:ff:fe00:31
/64  server01          eth0                      default         Thu
Feb  7 18:30:53 2019
fe80::a200:ff:fe00:32
/64  server02          eth0                      default         Thu
Feb  7 18:30:53 2019
fe80::a200:ff:fe00:33
/64  server03          eth0                      default         Thu
Feb  7 18:30:53 2019
fe80::a200:ff:fe00:34
/64  server04          eth0                      default         Thu
Feb  7 18:30:53 2019
```

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         Thu Feb  7 18:30:53 2019
192.168.0.11/24           leaf01            eth0
default         Thu Feb  7 18:30:53 2019
192.168.0.12/24           leaf02            eth0
default         Thu Feb  7 18:30:53 2019
192.168.0.13/24           leaf03            eth0
default         Thu Feb  7 18:30:53 2019
192.168.0.14/24           leaf04            eth0
default         Thu Feb  7 18:30:53 2019
192.168.0.21/24           spine01           eth0
default         Thu Feb  7 18:30:53 2019
192.168.0.22/24           spine02           eth0
default         Thu Feb  7 18:30:53 2019
192.168.0.31/24           server01          eth0
default         Thu Feb  7 18:30:53 2019
192.168.0.32/24           server02          eth0
default         Thu Feb  7 18:30:53 2019
192.168.0.33/24           server03          eth0
default         Thu Feb  7 18:30:53 2019
192.168.0.34/24           server04          eth0
default         Thu Feb  7 18:30:53 2019
```

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
------------------------- ----------------- -------------------------
--------------- -------------------------
2001:c15c:d06:f00d::16/12 leaf01            lo
default         Fri Feb  8 00:35:07 2019
8
2001:cafe:babe:0:22::/128 leaf01            DataVrf1080
DataVrf1080     Fri Feb  8 00:35:07 2019
2001:cafe:babe:1:22::/128 leaf01            DataVrf1081
DataVrf1081     Fri Feb  8 00:35:07 2019
2001:cafe:babe:2:22::/128 leaf01            DataVrf1082
DataVrf1082     Fri Feb  8 00:35:07 2019
2001:fee1:600d:10::1/64   leaf01            VlanA-1.102
DataVrf1082     Fri Feb  8 00:35:07 2019
2001:fee1:600d:11::1/64   leaf01            VlanA-1.103
default         Fri Feb  8 00:35:07 2019
2001:fee1:600d:12::1/64   leaf01            VlanA-1.104
default         Fri Feb  8 00:35:07 2019
```

```
2001:fee1:600d:13::1/64   leaf01            VlanA-1.105
default         Fri Feb  8 00:35:07 2019
2001:fee1:600d:14::1/64   leaf01            VlanA-1.106
default         Fri Feb  8 00:35:07 2019
2001:fee1:600d:e::1/64    leaf01            VlanA-1.100
DataVrf1080     Fri Feb  8 00:35:07 2019
2001:fee1:600d:f::1/64    leaf01            VlanA-1.101
DataVrf1081     Fri Feb  8 00:35:07 2019
2001:fee1:d00d:1::1/64    leaf01            vlan1001-v0
vrf1            Fri Feb  8 00:35:07 2019
2001:fee1:d00d:1::2/64    leaf01            vlan1001
vrf1            Fri Feb  8 00:35:07 2019
2001:fee1:d00d:2::1/64    leaf01            vlan1002-v0
vrf1            Fri Feb  8 00:35:07 2019
```

Example: View Changes to IP Address Information

This example shows the IPv4 address information that changed for all devices around 1 day ago.

```
cumulus@switch:~$ netq show ip addresses around 1d
Matching address records:
Address                   Hostname          Interface
VRF             Last Changed
------------------------- ----------------- -------------------------
--------------- -------------------------
192.168.0.15/24           leaf01            eth0
mgmt            Thu Feb  7 22:49:26 2019
27.0.0.22/32              leaf01            lo
default         Thu Feb  7 22:49:26 2019
3.0.3.129/26              leaf01            VlanA-1.100
DataVrf1080     Thu Feb  7 22:49:26 2019
3.0.3.193/26              leaf01            VlanA-1.101
DataVrf1081     Thu Feb  7 22:49:26 2019
3.0.4.1/26                leaf01            VlanA-1.102
DataVrf1082     Thu Feb  7 22:49:26 2019
3.0.4.129/26              leaf01            VlanA-1.104
default         Thu Feb  7 22:49:26 2019
3.0.4.193/26              leaf01            VlanA-1.105
default         Thu Feb  7 22:49:26 2019
3.0.4.65/26               leaf01            VlanA-1.103
default         Thu Feb  7 22:49:26 2019
3.0.5.1/26                leaf01            VlanA-1.106
default         Thu Feb  7 22:49:26 2019
30.0.0.22/32              leaf01            DataVrf1080
DataVrf1080     Thu Feb  7 22:49:26 2019
30.0.1.22/32              leaf01            DataVrf1081
DataVrf1081     Thu Feb  7 22:49:26 2019
30.0.2.22/32              leaf01            DataVrf1082
DataVrf1082     Thu Feb  7 22:49:26 2019
45.0.0.13/26              leaf01            NetQBond-1
mgmt            Thu Feb  7 22:49:26 2019
```

```
6.0.0.1/26                leaf01            vlan1000-v0
vrf1            Thu Feb  7 22:49:26 2019
6.0.0.129/26              leaf01            vlan1002-v0
vrf1            Thu Feb  7 22:49:26 2019
```

Example: Obtain a Count of IP Addresses Used on a Node

This example shows the number of IPv4 and IPv6 addresses on the node leaf01. Note that you must specify a hostname to use the count option.

```
cumulus@switch:~$ netq leaf01 show ip addresses count
Count of matching address records: 33

cumulus@switch:~$ netq leaf01 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
- 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     Thu Feb  7 22:49:26 2019
169.254.0.1               leaf01            swp51
44:38:39:00:00:54  default         no     Thu Feb  7 22:49:26 2019
169.254.0.1               leaf01            swp52
44:38:39:00:00:25  default         no     Thu Feb  7 22:49:26 2019
169.254.0.1               leaf02            swp51
44:38:39:00:00:29  default         no     Thu Feb  7 22:49:26 2019
169.254.0.1               leaf02            swp52
44:38:39:00:00:5e  default         no     Thu Feb  7 22:49:26 2019
169.254.0.1               leaf03            swp51
44:38:39:00:00:50  default         no     Thu Feb  7 22:49:26 2019
```

```
169.254.0.1               leaf03      swp52
44:38:39:00:00:1c  default         no    Thu Feb  7 22:49:26 2019
169.254.0.1               leaf04      swp51
44:38:39:00:00:3c  default         no    Thu Feb  7 22:49:26 2019
169.254.0.1               leaf04      swp52
44:38:39:00:00:47  default         no    Thu Feb  7 22:49:26 2019
169.254.0.1               spine01     swp1
44:38:39:00:00:53  default         no    Thu Feb  7 22:49:26 2019
169.254.0.1               spine01     swp2
44:38:39:00:00:28  default         no    Thu Feb  7 22:49:26 2019
169.254.0.1               spine01     swp3
44:38:39:00:00:4f  default         no    Thu Feb  7 22:49:26 2019
169.254.0.1               spine01     swp4
44:38:39:00:00:3b  default         no    Thu Feb  7 22:49:26 2019
169.254.0.1               spine02     swp1
44:38:39:00:00:24  default         no    Thu Feb  7 22:49:26 2019
169.254.0.1               spine02     swp2
44:38:39:00:00:5d  default         no    Thu Feb  7 22:49:26 2019
169.254.0.1               spine02     swp3
44:38:39:00:00:1b  default         no    Thu Feb  7 22:49:26 2019
169.254.0.1               spine02     swp4
44:38:39:00:00:46  default         no    Thu Feb  7 22:49:26 2019
192.168.0.11              oob-mgmt-server   eth1
a0:00:00:00:00:11  default         no    Thu Feb  7 22:49:26 2019
192.168.0.12              oob-mgmt-server   eth1
a0:00:00:00:00:12  default         no    Thu Feb  7 22:49:26 2019
192.168.0.13              oob-mgmt-server   eth1
a0:00:00:00:00:13  default         no    Thu Feb  7 22:49:26 2019
192.168.0.14              oob-mgmt-server   eth1
a0:00:00:00:00:14  default         no    Thu Feb  7 22:49:26 2019
192.168.0.21              oob-mgmt-server   eth1
a0:00:00:00:00:21  default         no    Thu Feb  7 22:49:26 2019
192.168.0.22              oob-mgmt-server   eth1
a0:00:00:00:00:22  default         no    Thu Feb  7 22:49:26 2019
192.168.0.253             oob-mgmt-server   eth1
a0:00:00:00:00:50  default         no    Thu Feb  7 22:49:26 2019
192.168.0.254             leaf01      eth0
44:38:39:00:00:57  default         no    Thu Feb  7 22:49:26 2019
192.168.0.254             leaf02      eth0
44:38:39:00:00:57  default         no    Thu Feb  7 22:49:26 2019
...
```

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      Thu Feb  7 22:49:26 2019
fe80::4638:39ff:fe00:29   leaf02            swp51
44:38:39:00:00:29  default         no      Thu Feb  7 22:49:26 2019
fe80::4638:39ff:fe00:4    leaf02            eth0
44:38:39:00:00:04  default         no      Thu Feb  7 22:49:26 2019
fe80::4638:39ff:fe00:5e   leaf02            swp52
44:38:39:00:00:5e  default         no      Thu Feb  7 22:49:26 2019
fe80::a200:ff:fe00:31     leaf02            eth0
a0:00:00:00:00:31  default         no      Thu Feb  7 22:49:26 2019
fe80::a200:ff:fe00:32     leaf02            eth0
a0:00:00:00:00:32  default         no      Thu Feb  7 22:49:26 2019
fe80::a200:ff:fe00:33     leaf02            eth0
a0:00:00:00:00:33  default         no      Thu Feb  7 22:49:26 2019
fe80::a200:ff:fe00:34     leaf02            eth0
a0:00:00:00:00:34  default         no      Thu Feb  7 22:49:26 2019
```

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
- 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
            Thu Feb  7 22:49:26 2019
yes    default         ::
/0                           server03          lo
            Thu Feb  7 22:49:26 2019
```

```
yes    default         ::
/0                           server01          lo
            Thu Feb  7 22:49:26 2019
yes    default         ::
/0                           server02          lo
            Thu Feb  7 22:49:26 2019

cumulus@switch:~$ netq show ip routes
Matching routes records:
Origin VRF             Prefix
Hostname          Nexthops                            Last Changed
------ --------------- ------------------------------
----------------- -----------------------------------
-------------------------
yes    DataVrf1080     3.0.3.128/26
leaf01            VlanA-1.100                         Fri Feb  8 00:
46:17 2019
yes    DataVrf1080     3.0.3.129/32
leaf01            VlanA-1.100                         Fri Feb  8 00:
46:17 2019
yes    DataVrf1080     30.0.0.22/32
leaf01            DataVrf1080                         Fri Feb  8 00:
46:17 2019
yes    DataVrf1081     3.0.3.192/26
leaf01            VlanA-1.101                         Fri Feb  8 00:
46:17 2019
yes    DataVrf1081     3.0.3.193/32
leaf01            VlanA-1.101                         Fri Feb  8 00:
46:17 2019
yes    DataVrf1081     30.0.1.22/32
leaf01            DataVrf1081                         Fri Feb  8 00:
46:17 2019
yes    DataVrf1082     3.0.4.0/26
leaf01            VlanA-1.102                         Fri Feb  8 00:
46:17 2019
yes    DataVrf1082     3.0.4.1/32
leaf01            VlanA-1.102                         Fri Feb  8 00:
46:17 2019
yes    DataVrf1082     30.0.2.22/32
leaf01            DataVrf1082                         Fri Feb  8 00:
46:17 2019
yes    default         27.0.0.22/32
leaf01            lo                                  Fri Feb  8 00:
46:17 2019
yes    default         3.0.4.128/26
leaf01            VlanA-1.104                         Fri Feb  8 00:
46:17 2019
yes    default         3.0.4.129/32
leaf01            VlanA-1.104                         Fri Feb  8 00:
46:17 2019
```

```
yes    default         3.0.4.192/26
leaf01            VlanA-1.105                         Fri Feb  8 00:
46:17 2019
yes    default         3.0.4.193/32
leaf01            VlanA-1.105                         Fri Feb  8 00:
46:17 2019
...
```

Example: View IP Routes for a Given IP Address

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

```
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  Fri Feb  8 00:
46:17 2019
no     default         10.0.0.12/32
leaf01            10.0.0.21: swp51, 10.0.0.22: swp52  Fri Feb  8 00:
46:17 2019
no     default         10.0.0.12/32
leaf04            10.0.0.21: swp51, 10.0.0.22: swp52  Fri Feb  8 00:
46:17 2019
no     default         10.0.0.12/32
spine02           10.0.0.12: swp2                     Fri Feb  8 00:
46:17 2019
no     default         10.0.0.12/32
spine01           10.0.0.12: swp2                     Fri Feb  8 00:
46:17 2019
yes    default         10.0.0.12/32
leaf02            lo                                  Fri Feb  8 00:
46:17 2019
```

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
      Fri Feb  8 00:46:17 2019
yes    default         192.168.0.0
/24                 spine01           eth0
    Fri Feb  8 00:46:17 2019
yes    default         192.168.0.21
/32                spine01           eth0
  Fri Feb  8 00:46:17 2019
```

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                   Fri Feb  8 00:46:17 2019
no     default         10.0.0.12
/32                   spine01           169.254.0.1:
swp2                   Fri Feb  8 00:46:17 2019
no     default         10.0.0.13
/32                   spine01           169.254.0.1:
swp3                   Fri Feb  8 00:46:17 2019
no     default         10.0.0.14
/32                   spine01           169.254.0.1:
swp4                   Fri Feb  8 00:46:17 2019
no     default         172.16.1.0
/24                  spine01           169.254.0.1:
swp1                   Fri Feb  8 00:46:17 2019
no     default         172.16.2.0
/24                  spine01           169.254.0.1:
swp2                   Fri Feb  8 00:46:17 2019
no     default         172.16.3.0
/24                  spine01           169.254.0.1:
swp3                   Fri Feb  8 00:46:17 2019
no     default         172.16.4.0
/24                  spine01           169.254.0.1:
swp4                   Fri Feb  8 00:46:17 2019
```

```
yes    default         10.0.0.21
/32                   spine01           lo
      Fri Feb  8 00:46:17 2019
yes    default         192.168.0.0
/24                 spine01           eth0
    Fri Feb  8 00:46:17 2019
yes    default         192.168.0.21
/32                spine01           eth0
  Fri Feb  8 00:46:17 2019
```

Example: View the Number of IP Routes on a Node

This example shows the total number of IP routes for all devices on a node.

```
cumulus@switch:~$ netq leaf01 show ip routes count
Count of matching routes records: 125

cumulus@switch:~$ netq leaf01 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
- 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 events [level info|level error|level
warning|level critical|level debug] type bgp [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
----------------- ---------------------------- ---------------
---------- ---------- ------------ -------------------------
exit-1            swp3(spine-1)                default
655537     655435     29/25/434    Thu Feb  7 18:19:50 2019
exit-1            swp3.2(spine-1)              DataVrf1080
655537     655435     15/13/0      Thu Feb  7 18:19:50 2019
exit-1            swp3.3(spine-1)              DataVrf1081
655537     655435     14/13/0      Thu Feb  7 18:19:50 2019
exit-1            swp3.4(spine-1)              DataVrf1082
655537     655435     16/13/0      Thu Feb  7 18:19:50 2019
exit-1            swp4(spine-2)                default
655537     655435     29/25/434    Thu Feb  7 18:19:50 2019
exit-1            swp4.2(spine-2)              DataVrf1080
655537     655435     16/13/0      Thu Feb  7 18:19:50 2019
exit-1            swp4.3(spine-2)              DataVrf1081
655537     655435     14/13/0      Thu Feb  7 18:19:50 2019
exit-1            swp4.4(spine-2)              DataVrf1082
655537     655435     16/13/0      Thu Feb  7 18:19:50 2019
exit-1            swp5(spine-3)                default
655537     655435     30/25/434    Thu Feb  7 18:19:50 2019
exit-1            swp5.2(spine-3)              DataVrf1080
655537     655435     15/13/0      Thu Feb  7 18:19:50 2019
exit-1            swp5.3(spine-3)              DataVrf1081
655537     655435     14/13/0      Thu Feb  7 18:19:50 2019
exit-1            swp5.4(spine-3)              DataVrf1082
655537     655435     16/13/0      Thu Feb  7 18:19:50 2019
exit-1            swp7                         default
655537     -          NotEstd      Thu Feb  7 18:31:44 2019
```

```
exit-1            swp7.2                       DataVrf1080
655537     -          NotEstd      Thu Feb   7 18:31:44 2019
exit-1            swp7.3                       DataVrf1081
655537     -          NotEstd      Thu Feb   7 18:31:44 2019
exit-1            swp7.4                       DataVrf1082
655537     -          NotEstd      Thu Feb   7 18:31:44 2019
exit-2            swp3(spine-1)                default
655538     655435     28/24/434    Thu Feb   7 18:19:50 2019
exit-2            swp3.2(spine-1)              DataVrf1080
655538     655435     14/12/0      Thu Feb   7 18:19:50 2019
exit-2            swp3.3(spine-1)              DataVrf1081
655538     655435     15/12/0      Thu Feb   7 18:19:50 2019
exit-2            swp3.4(spine-1)              DataVrf1082
655538     655435     15/12/0      Thu Feb   7 18:19:50 2019
exit-2            swp4(spine-2)                default
655538     655435     28/24/434    Thu Feb   7 18:19:50 2019
exit-2            swp4.2(spine-2)              DataVrf1080
655538     655435     14/12/0      Thu Feb   7 18:19:50 2019
exit-2            swp4.3(spine-2)              DataVrf1081
655538     655435     15/12/0      Thu Feb   7 18:19:50 2019
exit-2            swp4.4(spine-2)              DataVrf1082
655538     655435     15/12/0      Thu Feb   7 18:19:50 2019
exit-2            swp5(spine-3)                default
655538     655435     27/24/434    Thu Feb   7 18:19:50 2019
exit-2            swp5.2(spine-3)              DataVrf1080
655538     655435     15/12/0      Thu Feb   7 18:19:50 2019
exit-2            swp5.3(spine-3)              DataVrf1081
655538     655435     15/12/0      Thu Feb   7 18:19:50 2019
exit-2            swp5.4(spine-3)              DataVrf1082
655538     655435     15/12/0      Thu Feb   7 18:19:50 2019
exit-2            swp7                         default
655538     -          NotEstd      Thu Feb   7 18:31:49 2019
exit-2            swp7.2                       DataVrf1080
655538     -          NotEstd      Thu Feb   7 18:31:49 2019
exit-2            swp7.3                       DataVrf1081
655538     -          NotEstd      Thu Feb   7 18:31:49 2019
exit-2            swp7.4                       DataVrf1082
655538     -          NotEstd      Thu Feb   7 18:31:49 2019
spine-1           swp10(exit-2)                default
655435     655538     10/5/0       Thu Feb   7 18:19:50 2019
spine-1           swp10.2(exit-2)              DataVrf1080
655435     655538     10/5/0       Thu Feb   7 18:19:50 2019
spine-1           swp10.3(exit-2)              DataVrf1081
655435     655538     10/5/0       Thu Feb   7 18:19:50 2019
spine-1           swp10.4(exit-2)              DataVrf1082
655435     655538     10/5/0       Thu Feb   7 18:19:50 2019
spine-1           swp3(leaf-11)                default
655435     655559     19/6/94      Thu Feb   7 18:19:50 2019
spine-1           swp3.2(leaf-11)              DataVrf1080
655435     655559     14/2/0       Thu Feb   7 18:19:50 2019
spine-1           swp3.3(leaf-11)              DataVrf1081
655435     655559     14/2/0       Thu Feb   7 18:19:50 2019
```

```
spine-1           swp3.4(leaf-11)              DataVrf1082
655435     655559     14/2/0       Thu Feb  7 18:19:50 2019
spine-1           swp4(leaf-12)                default
655435     655560     19/6/64      Thu Feb  7 18:19:50 2019
spine-1           swp4.2(leaf-12)              DataVrf1080
655435     655560     14/2/0       Thu Feb  7 18:19:50 2019
spine-1           swp4.3(leaf-12)              DataVrf1081
655435     655560     14/2/0       Thu Feb  7 18:19:50 2019
spine-1           swp4.4(leaf-12)              DataVrf1082
655435     655560     14/2/0       Thu Feb  7 18:19:50 2019
spine-1           swp5(leaf-21)                default
655435     655561     19/6/50      Thu Feb  7 18:19:50 2019
spine-1           swp5.2(leaf-21)              DataVrf1080
655435     655561     14/2/0       Thu Feb  7 18:19:50 2019
spine-1           swp5.3(leaf-21)              DataVrf1081
655435     655561     14/2/0       Thu Feb  7 18:19:50 2019
spine-1           swp5.4(leaf-21)              DataVrf1082
655435     655561     14/2/0       Thu Feb  7 18:19:50 2019
spine-1           swp6(leaf-22)                default
655435     655562     19/6/62      Thu Feb  7 18:19:50 2019
spine-1           swp6.2(leaf-22)              DataVrf1080
655435     655562     14/2/0       Thu Feb  7 18:19:50 2019
spine-1           swp6.3(leaf-22)              DataVrf1081
655435     655562     14/2/0       Thu Feb  7 18:19:50 2019
spine-1           swp6.4(leaf-22)              DataVrf1082
655435     655562     14/2/0       Thu Feb  7 18:19:50 2019
spine-1           swp7(leaf-1)                 default
655435     655557     17/5/54      Thu Feb  7 18:19:50 2019
spine-1           swp7.2(leaf-1)               DataVrf1080
655435     655557     14/2/0       Thu Feb  7 18:19:50 2019
spine-1           swp7.3(leaf-1)               DataVrf1081
655435     655557     14/2/0       Thu Feb  7 18:19:50 2019
spine-1           swp7.4(leaf-1)               DataVrf1082
655435     655557     14/2/0       Thu Feb  7 18:19:50 2019
spine-1           swp8(leaf-2)                 default
655435     655558     17/5/54      Thu Feb  7 18:19:50 2019
spine-1           swp8.2(leaf-2)               DataVrf1080
655435     655558     14/2/0       Thu Feb  7 18:19:50 2019
spine-1           swp8.3(leaf-2)               DataVrf1081
655435     655558     14/2/0       Thu Feb  7 18:19:50 2019
spine-1           swp8.4(leaf-2)               DataVrf1082
655435     655558     14/2/0       Thu Feb  7 18:19:50 2019
spine-1           swp9(exit-1)                 default
655435     655537     19/5/0       Thu Feb  7 18:19:50 2019
spine-1           swp9.2(exit-1)               DataVrf1080
655435     655537     19/5/0       Thu Feb  7 18:19:50 2019
spine-1           swp9.3(exit-1)               DataVrf1081
655435     655537     19/5/0       Thu Feb  7 18:19:50 2019
spine-1           swp9.4(exit-1)               DataVrf1082
655435     655537     19/5/0       Thu Feb  7 18:19:50 2019
spine-2           swp10(exit-2)                default
655435     655538     10/5/0       Thu Feb  7 18:19:50 2019
```

```
spine-2           swp10.3(exit-2)              DataVrf1081
655435     655538     10/5/0       Thu Feb  7 18:19:50 2019
spine-2           swp10.4(exit-2)              DataVrf1082
655435     655538     10/5/0       Thu Feb  7 18:19:50 2019
spine-2           swp3.2(leaf-11)              DataVrf1080
655435     655559     14/2/0       Thu Feb  7 18:19:50 2019

...
```

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           swp3(spine01)                default
655557     655435     42/27/324    Thu Feb  7 18:19:50 2019
spine02           swp3.2(spine01)              DataVrf1080
655557     655435     31/18/0      Thu Feb  7 18:19:50 2019
spine02           swp3.3(spine01)              DataVrf1081
655557     655435     31/18/0      Thu Feb  7 18:19:50 2019
spine02           swp3.4(spine01)              DataVrf1082
655557     655435     29/18/0      Thu Feb  7 18:19:50 2019
spine02           swp5(spine03)                default
655557     655435     42/27/324    Thu Feb  7 18:19:50 2019
spine02           swp5.2(spine03)              DataVrf1080
655557     655435     31/18/0      Thu Feb  7 18:19:50 2019
spine02           swp5.3(spine03)              DataVrf1081
655557     655435     31/18/0      Thu Feb  7 18:19:50 2019
spine02           swp5.4(spine03)              DataVrf1082
655557     655435     29/18/0      Thu Feb  7 18:19:50 2019
```

Example: View BGP Configuration Information for a Given ASN

This example shows the BGP configuration information for ASN of *655557*. This ASN is associated with spine02 and so the results show the BGP neighbors for that switch.

```
cumulus@switch:~$ netq show bgp asn 655557
Matching bgp records:
Hostname          Neighbor                     VRF             ASN
    Peer ASN   PfxRx        Last Changed
----------------- ---------------------------- ---------------
---------- ---------- ------------ -------------------------
```

```
spine02           swp3(spine01)                 default
655557     655435     42/27/324    Thu Feb  7 18:19:50 2019
spine02           swp3.2(spine01)               DataVrf1080
655557     655435     31/18/0      Thu Feb  7 18:19:50 2019
spine02           swp3.3(spine01)               DataVrf1081
655557     655435     31/18/0      Thu Feb  7 18:19:50 2019
spine02           swp3.4(spine01)               DataVrf1082
655557     655435     29/18/0      Thu Feb  7 18:19:50 2019
spine02           swp5(spine03)                 default
655557     655435     42/27/324    Thu Feb  7 18:19:50 2019
spine02           swp5.2(spine03)               DataVrf1080
655557     655435     31/18/0      Thu Feb  7 18:19:50 2019
spine02           swp5.3(spine03)               DataVrf1081
655557     655435     31/18/0      Thu Feb  7 18:19:50 2019
spine02           swp5.4(spine03)               DataVrf1082
655557     655435     29/18/0      Thu Feb  7 18:19:50 2019
```

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
----------------- ---------------------------- ---------------
---------- ---------- ------------ -------------------------
exit01            swp3(spine01)                 default
655537     655435     29/25/434    Thu Feb  7 18:19:50 2019
exit01            swp3.2(spine01)               DataVrf1080
655537     655435     15/13/0      Thu Feb  7 18:19:50 2019
exit01            swp3.3(spine01)               DataVrf1081
655537     655435     14/13/0      Thu Feb  7 18:19:50 2019
exit01            swp3.4(spine01)               DataVrf1082
655537     655435     16/13/0      Thu Feb  7 18:19:50 2019
exit01            swp4(spine02)                 default
655537     655435     29/25/434    Thu Feb  7 18:19:50 2019
exit01            swp4.2(spine02)               DataVrf1080
655537     655435     16/13/0      Thu Feb  7 18:19:50 2019
exit01            swp4.3(spine02)               DataVrf1081
655537     655435     14/13/0      Thu Feb  7 18:19:50 2019
exit01            swp4.4(spine02)               DataVrf1082
655537     655435     16/13/0      Thu Feb  7 18:19:50 2019
exit01            swp5(spine03)                 default
655537     655435     30/25/434    Thu Feb  7 18:19:50 2019
exit01            swp5.2(spine03)               DataVrf1080
655537     655435     15/13/0      Thu Feb  7 18:19:50 2019
exit01            swp5.3(spine03)               DataVrf1081
655537     655435     14/13/0      Thu Feb  7 18:19:50 2019
exit01            swp5.4(spine03)               DataVrf1082
655537     655435     16/13/0      Thu Feb  7 18:19:50 2019
```

```
exit01            swp6(firewall01)             default
655537     655539     73/69/-      Thu Feb  7 18:26:30 2019
exit01            swp6.2(firewall01)           DataVrf1080
655537     655539     73/69/-      Thu Feb  7 18:26:30 2019
exit01            swp6.3(firewall01)           DataVrf1081
655537     655539     73/69/-      Thu Feb  7 18:26:30 2019
exit01            swp6.4(firewall01)           DataVrf1082
655537     655539     73/69/-      Thu Feb  7 18:26:30 2019
exit01            swp7                         default
655537     -          NotEstd      Thu Feb  7 18:31:44 2019
exit01            swp7.2                       DataVrf1080
655537     -          NotEstd      Thu Feb  7 18:31:44 2019
exit01            swp7.3                       DataVrf1081
655537     -          NotEstd      Thu Feb  7 18:31:44 2019
exit01            swp7.4                       DataVrf1082
655537     -          NotEstd      Thu Feb  7 18:31:44 2019
exit02            swp3(spine01)                default
655538     655435     28/24/434    Thu Feb  7 18:19:50 2019
exit02            swp3.2(spine01)              DataVrf1080
655538     655435     14/12/0      Thu Feb  7 18:19:50 2019
exit02            swp3.3(spine01)              DataVrf1081
655538     655435     15/12/0      Thu Feb  7 18:19:50 2019
exit02            swp3.4(spine01)              DataVrf1082
655538     655435     15/12/0      Thu Feb  7 18:19:50 2019
exit02            swp4(spine02)                default
655538     655435     28/24/434    Thu Feb  7 18:19:50 2019
exit02            swp4.2(spine02)              DataVrf1080
655538     655435     14/12/0      Thu Feb  7 18:19:50 2019
exit02            swp4.3(spine02)              DataVrf1081
655538     655435     15/12/0      Thu Feb  7 18:19:50 2019
exit02            swp4.4(spine02)              DataVrf1082
655538     655435     15/12/0      Thu Feb  7 18:19:50 2019
exit02            swp5(spine03)                default
655538     655435     27/24/434    Thu Feb  7 18:19:50 2019
exit02            swp5.2(spine03)              DataVrf1080
655538     655435     15/12/0      Thu Feb  7 18:19:50 2019
exit02            swp5.3(spine03)              DataVrf1081
655538     655435     15/12/0      Thu Feb  7 18:19:50 2019
exit02            swp5.4(spine03)              DataVrf1082
655538     655435     15/12/0      Thu Feb  7 18:19:50 2019
exit02            swp6(firewall01)             default
655538     655539     7/5/-        Thu Feb  7 18:26:30 2019
exit02            swp6.2(firewall01)           DataVrf1080
655538     655539     7/5/-        Thu Feb  7 18:26:30 2019
exit02            swp6.3(firewall01)           DataVrf1081
655538     655539     7/5/-        Thu Feb  7 18:26:30 2019
exit02            swp6.4(firewall01)           DataVrf1082
655538     655539     7/5/-        Thu Feb  7 18:26:30 2019
exit02            swp7                         default
655538     -          NotEstd      Thu Feb  7 18:31:49 2019
exit02            swp7.2                       DataVrf1080
655538     -          NotEstd      Thu Feb  7 18:31:49 2019
```

```
exit02            swp7.3                       DataVrf1081
655538     -          NotEstd      Thu Feb  7 18:31:49 2019
exit02            swp7.4                       DataVrf1082
655538     -          NotEstd      Thu Feb  7 18:31:49 2019
firewall01        swp3(exit01)                 default
655539     655537     29/27/-      Thu Feb  7 18:26:30 2019
firewall01        swp3.2(exit01)               default
655539     655537     15/15/-      Thu Feb  7 18:26:30 2019
firewall01        swp3.3(exit01)               default
655539     655537     15/15/-      Thu Feb  7 18:26:30 2019
firewall01        swp3.4(exit01)               default
655539     655537     15/15/-      Thu Feb  7 18:26:30 2019
firewall01        swp4(exit02)                 default
655539     655538     29/27/-      Thu Feb  7 18:26:30 2019
firewall01        swp4.2(exit02)               default
655539     655538     15/15/-      Thu Feb  7 18:26:30 2019
firewall01        swp4.3(exit02)               default
655539     655538     15/15/-      Thu Feb  7 18:26:30 2019
firewall01        swp4.4(exit02)               default
655539     655538     15/15/-      Thu Feb  7 18:26:30 2019
spine01           swp10(exit02)                default
655435     655538     10/5/0       Thu Feb  7 18:19:50 2019
spine01           swp10.2(exit02)              DataVrf1080
655435     655538     10/5/0       Thu Feb  7 18:19:50 2019
spine01           swp10.3(exit02)              DataVrf1081
655435     655538     10/5/0       Thu Feb  7 18:19:50 2019
spine01           swp10.4(exit02)              DataVrf1082
655435     655538     10/5/0       Thu Feb  7 18:19:50 2019
spine01           swp7(leaf01)                 default
655435     655557     17/5/54      Thu Feb  7 18:19:50 2019
spine01           swp7.2(leaf01)               DataVrf1080
655435     655557     14/2/0       Thu Feb  7 18:19:50 2019
spine01           swp7.3(leaf01)               DataVrf1081
655435     655557     14/2/0       Thu Feb  7 18:19:50 2019
spine01           swp7.4(leaf01)               DataVrf1082
655435     655557     14/2/0       Thu Feb  7 18:19:50 2019
spine01           swp8(leaf02)                 default
655435     655558     17/5/54      Thu Feb  7 18:19:50 2019
spine01           swp8.2(leaf02)               DataVrf1080
655435     655558     14/2/0       Thu Feb  7 18:19:50 2019
spine01           swp8.3(leaf02)               DataVrf1081
655435     655558     14/2/0       Thu Feb  7 18:19:50 2019
spine01           swp8.4(leaf02)               DataVrf1082
655435     655558     14/2/0       Thu Feb  7 18:19:50 2019
spine01           swp9(exit01)                 default
655435     655537     19/5/0       Thu Feb  7 18:19:50 2019
spine01           swp9.2(exit01)               DataVrf1080
655435     655537     19/5/0       Thu Feb  7 18:19:50 2019
spine01           swp9.3(exit01)               DataVrf1081
655435     655537     19/5/0       Thu Feb  7 18:19:50 2019
spine01           swp9.4(exit01)               DataVrf1082
655435     655537     19/5/0       Thu Feb  7 18:19:50 2019
```

```
spine02           swp10(exit02)                default
655435     655538     10/5/0       Thu Feb  7 18:19:50 2019
spine02           swp10.3(exit02)              DataVrf1081
655435     655538     10/5/0       Thu Feb  7 18:19:50 2019
spine02           swp10.4(exit02)              DataVrf1082
655435     655538     10/5/0       Thu Feb  7 18:19:50 2019
spine02           swp7(leaf01)                 default
655435     655557     17/5/62      Thu Feb  7 18:19:50 2019
spine02           swp7.2(leaf01)               DataVrf1080
655435     655557     14/2/0       Thu Feb  7 18:19:50 2019
spine02           swp7.3(leaf01)               DataVrf1081
655435     655557     14/2/0       Thu Feb  7 18:19:50 2019
spine02           swp7.4(leaf01)               DataVrf1082
655435     655557     14/2/0       Thu Feb  7 18:19:50 2019
spine02           swp8(leaf02)                 default
655435     655558     17/5/62      Thu Feb  7 18:19:50 2019
spine02           swp8.2(leaf02)               DataVrf1080
655435     655558     14/2/0       Thu Feb  7 18:19:50 2019
spine02           swp8.3(leaf02)               DataVrf1081
655435     655558     14/2/0       Thu Feb  7 18:19:50 2019
spine02           swp8.4(leaf02)               DataVrf1082
655435     655558     14/2/0       Thu Feb  7 18:19:50 2019
spine02           swp9(exit01)                 default
655435     655537     19/5/0       Thu Feb  7 18:19:50 2019
spine02           swp9.2(exit01)               DataVrf1080
655435     655537     19/5/0       Thu Feb  7 18:19:50 2019
spine02           swp9.4(exit01)               DataVrf1082
655435     655537     19/5/0       Thu Feb  7 18:19:50 2019
spine02           swp10.2(exit02)              DataVrf1080
655435     655538     10/5/0       Thu Feb  7 18:19:50 2019
spine02           swp9.3(exit01)               DataVrf1081
655435     655537     19/5/0       Thu Feb  7 18:19:50 2019
leaf01            swp3(spine01)                default
655557     655435     42/27/324    Thu Feb  7 18:19:50 2019
leaf01            swp3.2(spine01)              DataVrf1080
655557     655435     31/18/0      Thu Feb  7 18:19:50 2019
leaf01            swp3.3(spine01)              DataVrf1081
655557     655435     31/18/0      Thu Feb  7 18:19:50 2019
leaf01            swp3.4(spine01)              DataVrf1082
655557     655435     29/18/0      Thu Feb  7 18:19:50 2019
leaf01            swp4(spine02)                default
655557     655435     42/27/324    Thu Feb  7 18:19:50 2019
leaf01            swp4.2(spine02)              DataVrf1080
655557     655435     31/18/0      Thu Feb  7 18:19:50 2019
leaf01            swp4.3(spine02)              DataVrf1081
655557     655435     31/18/0      Thu Feb  7 18:19:50 2019
leaf01            swp4.4(spine02)              DataVrf1082
655557     655435     29/18/0      Thu Feb  7 18:19:50 2019
leaf01            swp5(spine03)                default
655557     655435     42/27/324    Thu Feb  7 18:19:50 2019
leaf01            swp5.2(spine03)              DataVrf1080
655557     655435     31/18/0      Thu Feb  7 18:19:50 2019
```

```
leaf01            swp5.3(spine03)             DataVrf1081
655557     655435     31/18/0     Thu Feb  7 18:19:50 2019
leaf01            swp5.4(spine03)             DataVrf1082
655557     655435     29/18/0     Thu Feb  7 18:19:50 2019
leaf02            swp3(spine01)               default
655558     655435     42/27/372   Thu Feb  7 18:19:50 2019
leaf02            swp3.2(spine01)             DataVrf1080
655558     655435     31/18/0     Thu Feb  7 18:19:50 2019
leaf02            swp3.3(spine01)             DataVrf1081
655558     655435     31/18/0     Thu Feb  7 18:19:50 2019
leaf02            swp3.4(spine01)             DataVrf1082
655558     655435     31/18/0     Thu Feb  7 18:19:50 2019
leaf02            swp4(spine02)               default
655558     655435     42/27/372   Thu Feb  7 18:19:50 2019
leaf02            swp4.2(spine02)             DataVrf1080
655558     655435     31/18/0     Thu Feb  7 18:19:50 2019
leaf02            swp4.3(spine02)             DataVrf1081
655558     655435     31/18/0     Thu Feb  7 18:19:50 2019
leaf02            swp4.4(spine02)             DataVrf1082
655558     655435     31/18/0     Thu Feb  7 18:19:50 2019
leaf02            swp5(spine03)               default
655558     655435     42/27/372   Thu Feb  7 18:19:50 2019
leaf02            swp5.2(spine03)             DataVrf1080
655558     655435     31/18/0     Thu Feb  7 18:19:50 2019
leaf02            swp5.3(spine03)             DataVrf1081
655558     655435     31/18/0     Thu Feb  7 18:19:50 2019
leaf02            swp5.4(spine03)             DataVrf1082
655558     655435     31/18/0     Thu Feb  7 18:19:50 2019
...
```

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 events type bgp between now and 5d

Matching bgp records:
Hostname          Message Type Severity
Message                             Timestamp
----------------- ------------ --------
----------------------------------- -------------------------
leaf01            bgp          info     BGP session with peer spine01
@desc 2h:10m:11s
                                        : state changed from failed
to esta
                                        blished
leaf01            bgp          info     BGP session with peer spine02
@desc 2h:10m:11s
                                        : state changed from failed
to esta
                                        blished
```

```
leaf01            bgp          info     BGP session with peer spine03
@desc 2h:10m:11s
                                        : state changed from failed
to esta
                                        blished
leaf01            bgp          info     BGP session with peer spine01
@desc 2h:10m:11s
                                        : state changed from failed
to esta
                                        blished
leaf01            bgp          info     BGP session with peer spine03
@desc 2h:10m:11s
                                        : state changed from failed
to esta
                                        blished
leaf01            bgp          info     BGP session with peer spine02
@desc 2h:10m:11s
                                        : state changed from failed
to esta
                                        blished
leaf01            bgp          info     BGP session with peer spine03
@desc 2h:10m:11s
                                        : state changed from failed
to esta
                                        blished
leaf01            bgp          info     BGP session with peer spine02
@desc 2h:10m:11s
                                        : state changed from failed
to esta
                                        blished
leaf01            bgp          info     BGP session with peer spine01
@desc 2h:10m:11s
                                        : state changed from failed
to esta
                                        blished

...
```

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 24 failed BGP sessions with a variety of reasons.

```
cumulus@switch:~$ netq check bgp
Total Nodes: 25, Failed Nodes: 3, Total Sessions: 220 , Failed
Sessions: 24,
Hostname          VRF             Peer Name         Peer Hostname
Reason                                        Last Changed
----------------- --------------- ----------------- -----------------
---------------------------------------------
-------------------------
exit-1            DataVrf1080     swp6.2            firewall-1
BGP session with peer firewall-1 swp6.2: AFI/ 1d:7h:56m:9s

SAFI evpn not activated on peer
exit-1            DataVrf1080     swp7.2            firewall-2
BGP session with peer firewall-2 (swp7.2 vrf  1d:7h:49m:31s

DataVrf1080) failed,

reason: Peer not configured
exit-1            DataVrf1081     swp6.3            firewall-1
BGP session with peer firewall-1 swp6.3: AFI/ 1d:7h:56m:9s

SAFI evpn not activated on peer
exit-1            DataVrf1081     swp7.3            firewall-2
BGP session with peer firewall-2 (swp7.3 vrf  1d:7h:49m:31s

DataVrf1081) failed,

reason: Peer not configured
exit-1            DataVrf1082     swp6.4            firewall-1
BGP session with peer firewall-1 swp6.4: AFI/ 1d:7h:56m:9s

SAFI evpn not activated on peer
exit-1            DataVrf1082     swp7.4            firewall-2
BGP session with peer firewall-2 (swp7.4 vrf  1d:7h:49m:31s

DataVrf1082) failed,

reason: Peer not configured
exit-1            default         swp6              firewall-1
BGP session with peer firewall-1 swp6: AFI/SA 1d:7h:56m:9s

FI evpn not activated on peer
exit-1            default         swp7              firewall-2
BGP session with peer firewall-2 (swp7 vrf de 1d:7h:49m:31s

fault) failed, reason: Peer not configured
exit-2            DataVrf1080     swp6.2            firewall-1
BGP session with peer firewall-1 swp6.2: AFI/ 1d:7h:56m:9s
```

```
SAFI evpn not activated on peer
exit-2            DataVrf1080     swp7.2            firewall-2
BGP session with peer firewall-2 (swp7.2 vrf  1d:7h:49m:26s

DataVrf1080) failed,

reason: Peer not configured
exit-2            DataVrf1081     swp6.3            firewall-1
BGP session with peer firewall-1 swp6.3: AFI/ 1d:7h:56m:9s

SAFI evpn not activated on peer
...
```

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
- 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 events [level info|level error|level
warning|level critical|level debug] type ospf [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                      Thu Feb  7
14:42:16 2019
leaf01            swp52                     0.0.0.0      Unnumbered
    Full       spine02           swp1                      Thu Feb  7
14:42:16 2019
leaf02            swp51                     0.0.0.0      Unnumbered
    Full       spine01           swp2                      Thu Feb  7
14:42:16 2019
leaf02            swp52                     0.0.0.0      Unnumbered
    Full       spine02           swp2                      Thu Feb  7
14:42:16 2019
leaf03            swp51                     0.0.0.0      Unnumbered
    Full       spine01           swp3                      Thu Feb  7
14:42:16 2019
leaf03            swp52                     0.0.0.0      Unnumbered
    Full       spine02           swp3                      Thu Feb  7
14:42:16 2019
leaf04            swp51                     0.0.0.0      Unnumbered
    Full       spine01           swp4                      Thu Feb  7
14:42:16 2019
leaf04            swp52                     0.0.0.0      Unnumbered
    Full       spine02           swp4                      Thu Feb  7
14:42:16 2019
spine01           swp1                      0.0.0.0      Unnumbered
    Full       leaf01            swp51                     Thu Feb  7
14:42:16 2019
spine01           swp2                      0.0.0.0      Unnumbered
    Full       leaf02            swp51                     Thu Feb  7
14:42:16 2019
```

```
spine01           swp3                      0.0.0.0      Unnumbered
    Full        leaf03           swp51                     Thu Feb  7
14:42:16 2019
spine01           swp4                      0.0.0.0      Unnumbered
    Full        leaf04           swp51                     Thu Feb  7
14:42:16 2019
spine02           swp1                      0.0.0.0      Unnumbered
    Full        leaf01           swp52                     Thu Feb  7
14:42:16 2019
spine02           swp2                      0.0.0.0      Unnumbered
    Full        leaf02           swp52                     Thu Feb  7
14:42:16 2019
spine02           swp3                      0.0.0.0      Unnumbered
    Full        leaf03           swp52                     Thu Feb  7
14:42:16 2019
spine02           swp4                      0.0.0.0      Unnumbered
    Full        leaf04           swp52                     Thu Feb  7
14:42:16 2019
```

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                      Thu Feb  7
14:42:16 2019
leaf01            swp52                     0.0.0.0      Unnumbered
    Full        spine02          swp1                      Thu Feb  7
14:42:16 2019
```

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                      Thu Feb  7
14:42:16 2019
leaf02            swp51                     0.0.0.0      Unnumbered
    Full       spine01          swp2                      Thu Feb  7
14:42:16 2019
leaf03            swp51                     0.0.0.0      Unnumbered
    Full       spine01          swp3                      Thu Feb  7
14:42:16 2019
leaf04            swp51                     0.0.0.0      Unnumbered
    Full       spine01          swp4                      Thu Feb  7
14:42:16 2019
```

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                      Thu Feb  7
14:42:16 2019
leaf01            swp52                     0.0.0.0      Unnumbered
    Full       spine02          swp1                      Thu Feb  7
14:42:16 2019
leaf02            swp51                     0.0.0.0      Unnumbered
    Full       spine01          swp2                      Thu Feb  7
14:42:16 2019
leaf02            swp52                     0.0.0.0      Unnumbered
    Full       spine02          swp2                      Thu Feb  7
14:42:16 2019
leaf03            swp51                     0.0.0.0      Unnumbered
    Full       spine01          swp3                      Thu Feb  7
14:42:16 2019
leaf03            swp52                     0.0.0.0      Unnumbered
    Full       spine02          swp3                      Thu Feb  7
14:42:16 2019
leaf04            swp51                     0.0.0.0      Unnumbered
    Full       spine01          swp4                      Thu Feb  7
14:42:16 2019
```

```
leaf04            swp52                     0.0.0.0      Unnumbered
    Full       spine02            swp4                        Thu Feb  7
14:42:16 2019
```

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
----------------- ------------------------- -------------------------
-------------------------
---------------------------------------------
-------------------------
spine03           swp6                      0.0.0.23
27.0.0.23                 mtu mismatch, mtu
mismatch                    Thu Feb  7 14:42:16 2019
leaf22            swp5                      0.0.0.17
27.0.0.17                 mtu mismatch, mtu
mismatch                    Thu Feb  7 14:42:16 2019
```

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 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>] [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         Fri Feb  8 01:35:49 2019
10.0.0.11/32              leaf01            swp51
default         Fri Feb  8 01:35:49 2019
10.0.0.11/32              leaf01            swp52
default         Fri Feb  8 01:35:49 2019
172.16.1.1/24             leaf01            br0
default         Fri Feb  8 01:35:49 2019
```

```
192.168.0.11/24           leaf01            eth0
default         Fri Feb  8 01:35:49 2019

cumulus@switch:~$ netq leaf03 show ip addresses
Matching address records:
Address                   Hostname          Interface
VRF             Last Changed
------------------------- ----------------- -------------------------
--------------- -------------------------
10.0.0.13/32              leaf03            lo
default         Thu Feb  7 18:31:29 2019
10.0.0.13/32              leaf03            swp51
default         Thu Feb  7 18:31:29 2019
10.0.0.13/32              leaf03            swp52
default         Thu Feb  7 18:31:29 2019
172.16.3.1/24             leaf03            br0
default         Thu Feb  7 18:31:29 2019
192.168.0.13/24           leaf03            eth0
default         Thu Feb  7 18:31:29 2019

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 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?

> ⓘ Lightweight network virtualization (LNV) was deprecated in Cumulus Linux 3.7.4 and will be removed in Cumulus Linux 4.0.0. Cumulus NetQ will continue to support and return LNV data as long as you are running a supported version of Cumulus Linux earlier than 4.0.0. For information on the support timeline, read this knowledge base article.

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 show interfaces type vxlan [state <remote-interface-state>]
[around <text-time>] [json]
netq <hostname> show interfaces type vxlan [state <remote-interface-
state>] [around <text-time>] [count] [json]
netq [<hostname>] show events [level info|level error|level
warning|level critical|level debug] type vxlan [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, 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                                       Fri Feb  8 01:35:49 2019
exit02            104001     EVPN   10.0.0.42
4001                                       Fri Feb  8 01:35:49 2019
leaf01            13         EVPN   10.0.0.112       13     10.0.0.134
(leaf04, leaf03)          Fri Feb  8 01:35:49 2019
leaf01            24         EVPN   10.0.0.112       24     10.0.0.134
(leaf04, leaf03)          Fri Feb  8 01:35:49 2019
leaf01            104001     EVPN   10.0.0.112
4001                                       Fri Feb  8 01:35:49 2019
```

```
leaf02            13         EVPN   10.0.0.112       13     10.0.0.134
(leaf04, leaf03)          Fri Feb  8 01:35:49 2019
leaf02            24         EVPN   10.0.0.112       24     10.0.0.134
(leaf04, leaf03)          Fri Feb  8 01:35:49 2019
leaf02            104001     EVPN   10.0.0.112
4001                                       Fri Feb  8 01:35:49 2019
leaf03            13         EVPN   10.0.0.134       13     10.0.0.112
(leaf02, leaf01)          Fri Feb  8 01:35:49 2019
leaf03            24         EVPN   10.0.0.134       24     10.0.0.112
(leaf02, leaf01)          Fri Feb  8 01:35:49 2019
leaf03            104001     EVPN   10.0.0.134
4001                                       Fri Feb  8 01:35:49 2019
leaf04            13         EVPN   10.0.0.134       13     10.0.0.112
(leaf02, leaf01)          Fri Feb  8 01:35:49 2019
leaf04            24         EVPN   10.0.0.134       24     10.0.0.112
(leaf02, leaf01)          Fri Feb  8 01:35:49 2019
leaf04            104001     EVPN   10.0.0.134
4001                                       Fri Feb  8 01:35:49 2019
```

This example shows the events and configuration changes that have occurred on the 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 events type vxlan 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        Fri Feb  8 01:
35:49 2019
exit02            104001     EVPN   10.0.0.42
4001                                       Add        Fri Feb  8 01:
35:49 2019
exit02            104001     EVPN   10.0.0.42
4001                                       Add        Fri Feb  8 01:
35:49 2019
exit02            104001     EVPN   10.0.0.42
4001                                       Add        Fri Feb  8 01:
35:49 2019
exit02            104001     EVPN   10.0.0.42
4001                                       Add        Fri Feb  8 01:
35:49 2019
exit02            104001     EVPN   10.0.0.42
4001                                       Add        Fri Feb  8 01:
35:49 2019
```

```
exit02            104001     EVPN   10.0.0.42
4001                                       Add        Fri Feb  8 01:
35:49 2019
exit01            104001     EVPN   10.0.0.41
4001                                       Add        Fri Feb  8 01:
35:49 2019
exit01            104001     EVPN   10.0.0.41
4001                                       Add        Fri Feb  8 01:
35:49 2019
exit01            104001     EVPN   10.0.0.41
4001                                       Add        Fri Feb  8 01:
35:49 2019
exit01            104001     EVPN   10.0.0.41
4001                                       Add        Fri Feb  8 01:
35:49 2019
exit01            104001     EVPN   10.0.0.41
4001                                       Add        Fri Feb  8 01:
35:49 2019
exit01            104001     EVPN   10.0.0.41
4001                                       Add        Fri Feb  8 01:
35:49 2019
exit01            104001     EVPN   10.0.0.41
4001                                       Add        Fri Feb  8 01:
35:49 2019
exit01            104001     EVPN   10.0.0.41
4001                                       Add        Fri Feb  8 01:
35:49 2019
leaf04            104001     EVPN   10.0.0.134
4001                                       Add        Fri Feb  8 01:
35:49 2019
leaf04            104001     EVPN   10.0.0.134
4001                                       Add        Fri Feb  8 01:
35:49 2019
leaf04            104001     EVPN   10.0.0.134
4001                                       Add        Fri Feb  8 01:
35:49 2019
leaf04            104001     EVPN   10.0.0.134
4001                                       Add        Fri Feb  8 01:
35:49 2019
leaf04            104001     EVPN   10.0.0.134
4001                                       Add        Fri Feb  8 01:
35:49 2019
leaf04            104001     EVPN   10.0.0.134
4001                                       Add        Fri Feb  8 01:
35:49 2019
leaf04            104001     EVPN   10.0.0.134
4001                                       Add        Fri Feb  8 01:
35:49 2019
leaf04            13         EVPN   10.0.0.134       13
10.0.0.112()                        Add        Fri Feb  8 01:35:49
2019
```

```
leaf04            13         EVPN   10.0.0.134       13
10.0.0.112()                        Add        Fri Feb  8 01:35:49
2019
leaf04            13         EVPN   10.0.0.134       13
10.0.0.112()                        Add        Fri Feb  8 01:35:49
2019
leaf04            13         EVPN   10.0.0.134       13
10.0.0.112()                        Add        Fri Feb  8 01:35:49
2019
leaf04            13         EVPN   10.0.0.134       13
10.0.0.112()                        Add        Fri Feb  8 01:35:49
2019
leaf04            13         EVPN   10.0.0.134       13
10.0.0.112()                        Add        Fri Feb  8 01:35:49
2019
leaf04            13         EVPN   10.0.0.134       13
10.0.0.112()                        Add        Fri Feb  8 01:35:49
2019
...
```

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)          Fri Feb  8 01:35:49 2019
leaf02            24         EVPN   10.0.0.112       24     10.0.0.134
(leaf04, leaf03)          Fri Feb  8 01:35:49 2019
leaf03            24         EVPN   10.0.0.134       24     10.0.0.112
(leaf02, leaf01)          Fri Feb  8 01:35:49 2019
leaf04            24         EVPN   10.0.0.134       24     10.0.0.112
(leaf02, leaf01)          Fri Feb  8 01:35:49 2019
```

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,  Fri
Feb  8 01:35:49 2019

VTEP: 10.0.0.112, MTU: 9000
leaf02            vni24                     vxlan
up         default         VNI: 24, PVID: 24, Master: bridge,  Fri
Feb  8 01:35:49 2019

VTEP: 10.0.0.112, MTU: 9000
leaf02            vxlan4001                 vxlan
up         default         VNI: 104001, PVID: 4001,            Fri
Feb  8 01:35:49 2019

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>] [around <text-time>]
[json]
```

```
netq [<hostname>] show events [level info|level error|level
warning|level critical|level debug] type vxlan [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.

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
----------------- ---------- ---------------- ---------
---------------- ---------------- -------------------------
leaf01            33         27.0.0.22        yes       197:
33           197:33           Fri Feb  8 01:48:27 2019
leaf01            34         27.0.0.22        yes       197:
34           197:34           Fri Feb  8 01:48:27 2019
leaf01            35         27.0.0.22        yes       197:
35           197:35           Fri Feb  8 01:48:27 2019
leaf01            36         27.0.0.22        yes       197:
36           197:36           Fri Feb  8 01:48:27 2019
leaf01            37         27.0.0.22        yes       197:
37           197:37           Fri Feb  8 01:48:27 2019
leaf01            38         27.0.0.22        yes       197:
38           197:38           Fri Feb  8 01:48:27 2019
leaf01            39         27.0.0.22        yes       197:
39           197:39           Fri Feb  8 01:48:27 2019
leaf01            40         27.0.0.22        yes       197:
40           197:40           Fri Feb  8 01:48:27 2019
```

```
leaf01            41         27.0.0.22        yes       197:
41           197:41           Fri Feb  8 01:48:27 2019
leaf01            42         27.0.0.22        yes       197:
42           197:42           Fri Feb  8 01:48:27 2019
leaf02            33         27.0.0.23        yes       198:
33           198:33           Thu Feb  7 18:31:41 2019
leaf02            34         27.0.0.23        yes       198:
34           198:34           Thu Feb  7 18:31:41 2019
leaf02            35         27.0.0.23        yes       198:
35           198:35           Thu Feb  7 18:31:41 2019
leaf02            36         27.0.0.23        yes       198:
36           198:36           Thu Feb  7 18:31:41 2019
leaf02            37         27.0.0.23        yes       198:
37           198:37           Thu Feb  7 18:31:41 2019
leaf02            38         27.0.0.23        yes       198:
38           198:38           Thu Feb  7 18:31:41 2019
leaf02            39         27.0.0.23        yes       198:
39           198:39           Thu Feb  7 18:31:41 2019
leaf02            40         27.0.0.23        yes       198:
40           198:40           Thu Feb  7 18:31:41 2019
leaf02            41         27.0.0.23        yes       198:
41           198:41           Thu Feb  7 18:31:41 2019
leaf02            42         27.0.0.23        yes       198:
42           198:42           Thu Feb  7 18:31:41 2019
...
```

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 42.

```
cumulus@switch:~$ netq show evpn vni 42
Matching evpn records:
Hostname          VNI        VTEP IP          In Kernel Export
RT        Import RT        Last Changed
----------------- ---------- ---------------- ---------
---------------- ---------------- -------------------------
leaf01            42         27.0.0.22        yes       197:
42           197:42           Thu Feb 14 00:48:24 2019
leaf02            42         27.0.0.23        yes       198:
42           198:42           Wed Feb 13 18:14:49 2019
leaf11            42         36.0.0.24        yes       199:
42           199:42           Wed Feb 13 18:14:22 2019
leaf12            42         36.0.0.24        yes       200:
42           200:42           Wed Feb 13 18:14:27 2019
leaf21            42         36.0.0.26        yes       201:
42           201:42           Wed Feb 13 18:14:33 2019
leaf22            42         36.0.0.26        yes       202:
42           202:42           Wed Feb 13 18:14:37 2019
```

View EVPN Events

You can view status and configuration change events for the EVPN protocol service using the `netq show events` command. This example shows the events that have occurred in the last 48 hours.

```
cumulus@switch:/$ netq show events type evpn between now and 48h
Matching events records:
Hostname          Message Type Severity
Message                             Timestamp
----------------- ------------ --------
----------------------------------- -------------------------
torc-21           evpn         info     VNI 33 state changed from
down to u 1d:8h:16m:29s
                                        p
torc-12           evpn         info     VNI 41 state changed from
down to u 1d:8h:16m:35s
                                        p
torc-11           evpn         info     VNI 39 state changed from
down to u 1d:8h:16m:41s
                                        p
tor-1             evpn         info     VNI 37 state changed from
down to u 1d:8h:16m:47s
                                        p
tor-2             evpn         info     VNI 42 state changed from
down to u 1d:8h:16m:51s
                                        p
torc-22           evpn         info     VNI 39 state changed from
down to u 1d:8h:17m:40s
                                        p
...
```

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 events [level info|level error|level
warning|level critical|level debug] type lnv [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
----------------- ---------- -------- ---------- ------ ------
-------------------------
spine01           SND        HER      up         3      6      Thu
Feb  7 18:31:31 2019
spine02           SND        HER      up         3      6      Thu
Feb  7 18:31:31 2019
spine03           SND        HER      up         3      6      Thu
Feb  7 18:31:31 2019
leaf01            RD         HER      up         4      6      Thu
Feb  7 18:31:31 2019
leaf02            RD         HER      up         4      6      Thu
Feb  7 18:31:31 2019
leaf11            RD         HER      up         0      0      Thu
Feb  7 18:31:31 2019
leaf12            RD         HER      up         4      6      Thu
Feb  7 18:31:31 2019
leaf21            RD         HER      up         4      6      Thu
Feb  7 18:31:31 2019
leaf22            RD         HER      up         4      6      Thu
Feb  7 18:31:31 2019
```

View LNV Status in the Past

You can view the status in the past using the `around` keyword. 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
----------------- ---------- -------- ---------- ------ ------
-------------------------
spine01           SND        HER      up         3      6      Thu
Feb  7 18:31:31 2019
spine02           SND        HER      up         3      6      Thu
Feb  7 18:31:31 2019
spine03           SND        HER      up         3      6      Thu
Feb  7 18:31:31 2019
```

```
leaf01            RD         HER      up         4      6      Thu
Feb  7 18:31:31 2019
leaf02            RD         HER      up         4      6      Thu
Feb  7 18:31:31 2019
leaf11            RD         HER      up         4      6      Thu
Feb  7 18:31:31 2019
leaf12            RD         HER      up         4      6      Thu
Feb  7 18:31:31 2019
leaf21            RD         HER      up         4      6      Thu
Feb  7 18:31:31 2019
leaf22            RD         HER      up         4      6      Thu
Feb  7 18:31:31 2019
```

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

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 156) 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                      Fri Feb  8 01:50:59 2019
server01          eth1                      leaf01
swp1                      Fri Feb  8 01:50:59 2019
server01          eth2                      leaf02
swp1                      Fri Feb  8 01:50:59 2019
```

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                      Thu Feb  7 18:31:26 2019
```

```
leaf01            swp1                      server01
eth1                      Thu Feb  7 18:31:26 2019
leaf01            swp2                      server02
eth1                      Thu Feb  7 18:31:26 2019
leaf01            swp49                     leaf02
swp49                     Thu Feb  7 18:31:26 2019
leaf01            swp50                     leaf02
swp50                     Thu Feb  7 18:31:26 2019
leaf01            swp51                     spine01
swp1                      Thu Feb  7 18:31:26 2019
leaf01            swp52                     spine02
swp1                      Thu Feb  7 18:31:26 2019
```

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                    Fri Feb  8 01:
50:49 2019
no     default         172.16.1.0/24
server01          10.1.3.1: uplink                    Fri Feb  8 01:
50:49 2019
yes    default         10.1.3.0/24
server01          uplink                              Fri Feb  8 01:
50:49 2019
yes    default         10.1.3.101/32
server01          uplink                              Fri Feb  8 01:
50:49 2019
yes    default         192.168.0.0/24
server01          eth0                                Fri Feb  8 01:
50:49 2019
yes    default         192.168.0.31/32
server01          eth0                                Fri Feb  8 01:
50:49 2019
```

Monitor Container Environments

The NetQ Agent monitors container environments the same way it monitors physical servers (see page 154) . 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 the Kubernetes container orchestrator.

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...

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 Platform on a host with at least 64G RAM.

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. Run `netq show kubernetes help` to see all the possible commands:

```
netq [<hostname>] show kubernetes cluster [name <kube-cluster-name>]
[around <text-time>] [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 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 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 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 replication-controller [name <kube-
rc-name>] [cluster <kube-cluster-name>] [namespace <namespace>]
[label <kube-rc-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>] [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 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 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]
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 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]
netq config add agent kubernetes-monitor [poll-period <text-duration-
period>]
netq config del agent kubernetes-monitor
netq config show agent kubernetes-monitor [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 NetQ Platform by its IP address. See the Install NetQ (2.1.0 version) 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@switch:~$ netq show kubernetes cluster
Matching kube_cluster records:
Master                   Cluster Name     Controller Status
Scheduler Status Nodes
------------------------ ---------------- --------------------
---------------- --------------------
server11:3.0.0.68        default          Healthy
Healthy          server11 server13 se

rver22 server11 serv

er12 server23 server

24
server12:3.0.0.69        default          Healthy
Healthy          server12 server21 se

rver23 server13 serv

er14 server21 server

22
```

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

```
cumulus@switch:~$ netq server11 show kubernetes cluster
Matching kube_cluster records:
Master                   Cluster Name     Controller Status
Scheduler Status Nodes
------------------------ ---------------- --------------------
---------------- --------------------
server11:3.0.0.68        default          Healthy
Healthy          server11 server13 se

rver22 server11 serv

er12 server23 server

24
```

Optionally, you can output the results in JSON format:

```
cumulus@server11:~$ netq show kubernetes cluster json
{
    "kube_cluster":[
        {
            "clusterName":"default",
            "schedulerStatus":"Healthy",
            "master":"server12:3.0.0.69",
            "nodes":"server12 server21 server23 server13 server14
server21 server22",
            "controllerStatus":"Healthy"
        },
        {
            "clusterName":"default",
            "schedulerStatus":"Healthy",
            "master":"server11:3.0.0.68",
            "nodes":"server11 server13 server22 server11 server12
server23 server24",
            "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 *around* keyword. This example shows the changes that have been made in the last hour.

```
cumulus@server11:~$ netq show kubernetes cluster around 1h
Matching kube_cluster records:
Master                   Cluster Name     Controller Status
Scheduler Status Nodes                                    DBState
Last changed
------------------------ ---------------- --------------------
---------------- ---------------------------------------- --------
-------------------------
server11:3.0.0.68        default          Healthy
Healthy          server11 server13 server22 server11 serv Add
Fri Feb  8 01:50:50 2019

er12 server23 server24
server12:3.0.0.69        default          Healthy
Healthy          server12 server21 server23 server13 serv Add
Fri Feb  8 01:50:50 2019

er14 server21 server22
```

```
server12:3.0.0.69        default          Healthy
Healthy          server12 server21 server23 server13      Add
Fri Feb  8 01:50:50 2019
server11:3.0.0.68        default          Healthy
Healthy          server11                                 Add
Fri Feb  8 01:50:50 2019
server12:3.0.0.69        default          Healthy
Healthy          server12                                 Add
Fri Feb  8 01:50:50 2019
```

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@server11:~$ netq show kubernetes pod
Matching kube_pod records:
Master                   Namespace    Name
IP               Node         Labels               Status
Containers               Last Changed
------------------------ ------------ --------------------
---------------- ------------ -------------------- --------
------------------------ ----------------
server11:3.0.0.68        default      cumulus-frr-8vssx
3.0.0.70         server13     pod-template-generat Running  cumulus-
frr:f8cac70bb217 Fri Feb  8 01:50:50 2019

ion:1 name:cumulus-f

rr controller-revisi

on-hash:3710533951
server11:3.0.0.68        default      cumulus-frr-dkkgp
3.0.5.135        server24     pod-template-generat Running  cumulus-
frr:577a60d5f40c Fri Feb  8 01:50:50 2019

ion:1 name:cumulus-f

rr controller-revisi

on-hash:3710533951
server11:3.0.0.68        default      cumulus-frr-f4bgx
3.0.3.196        server11     pod-template-generat Running  cumulus-
frr:1bc73154a9f5 Fri Feb  8 01:50:50 2019

ion:1 name:cumulus-f

rr controller-revisi
```

```
on-hash:3710533951
server11:3.0.0.68        default      cumulus-frr-gqqxn
3.0.2.5          server22     pod-template-generat Running  cumulus-
frr:3ee0396d126a Fri Feb  8 01:50:50 2019

ion:1 name:cumulus-f

rr controller-revisi

on-hash:3710533951
server11:3.0.0.68        default      cumulus-frr-kdh9f
3.0.3.197        server12     pod-template-generat Running  cumulus-
frr:94b6329ecb50 Fri Feb  8 01:50:50 2019

ion:1 name:cumulus-f

rr controller-revisi

on-hash:3710533951
server11:3.0.0.68        default      cumulus-frr-mvv8m
3.0.5.134        server23     pod-template-generat Running  cumulus-
frr:b5845299ce3c Fri Feb  8 01:50:50 2019

ion:1 name:cumulus-f

rr controller-revisi

on-hash:3710533951
server11:3.0.0.68        default      httpd-5456469bfd-bq9
10.244.49.65     server22     app:httpd            Running  httpd:
79b7f532be2d       Fri Feb  8 01:50:50 2019
                                      zm
server11:3.0.0.68        default      influxdb-6cdb566dd-8
10.244.162.128   server13     app:influx           Running  influxdb:
15dce703cdec    Fri Feb  8 01:50:50 2019
                                      9lwn
server11:3.0.0.68        default      nginx-8586cf59-26pj5
10.244.9.193     server24     run:nginx            Running  nginx:
6e2b65070c86       Fri Feb  8 01:50:50 2019
server11:3.0.0.68        default      nginx-8586cf59-c82ns
10.244.40.128    server12     run:nginx            Running  nginx:
01b017c26725       Fri Feb  8 01:50:50 2019
server11:3.0.0.68        default      nginx-8586cf59-wjwgp
10.244.49.64     server22     run:nginx            Running  nginx:
ed2b4254e328       Fri Feb  8 01:50:50 2019
server11:3.0.0.68        kube-system  calico-etcd-pfg9r
3.0.0.68         server11     k8s-app:calico-etcd  Running  calico-
etcd:f95f44b745a7 Fri Feb  8 01:50:50 2019

pod-template-generat
```

```
ion:1 controller-rev

ision-hash:142071906

5
server11:3.0.0.68        kube-system  calico-kube-controll
3.0.2.5          server22     k8s-app:calico-kube- Running  calico-
kube-controllers: Fri Feb  8 01:50:50 2019
                                      ers-d669cc78f-
4r5t2                                controllers
3688b0c5e9c5
server11:3.0.0.68        kube-system  calico-node-4px69
3.0.2.5          server22     k8s-app:calico-node  Running  calico-
node:1d01648ebba4 Fri Feb  8 01:50:50 2019

pod-template-generat          install-cni:da350802a3d2

ion:1 controller-rev

ision-hash:324404111

9
server11:3.0.0.68        kube-system  calico-node-bt8w6
3.0.3.196        server11     k8s-app:calico-node  Running  calico-
node:9b3358a07e5e Fri Feb  8 01:50:50 2019

pod-template-generat          install-cni:d38713e6fdd8

ion:1 controller-rev

ision-hash:324404111

9
server11:3.0.0.68        kube-system  calico-node-gtmkv
3.0.3.197        server12     k8s-app:calico-node  Running  calico-
node:48fcc6c40a6b Fri Feb  8 01:50:50 2019

pod-template-generat          install-cni:f0838a313eff

ion:1 controller-rev

ision-hash:324404111

9
server11:3.0.0.68        kube-system  calico-node-mvslq
3.0.5.134        server23     k8s-app:calico-node  Running  calico-
node:7b361aece76c Fri Feb  8 01:50:50 2019

pod-template-generat          install-cni:f2da6bc36bf8

ion:1 controller-rev
```

```
ision-hash:324404111

9
server11:3.0.0.68        kube-system  calico-node-sjj2s
3.0.5.135        server24     k8s-app:calico-node  Running  calico-
node:6e13b2b73031 Fri Feb  8 01:50:50 2019

pod-template-generat          install-cni:fa4b2b17fba9

ion:1 controller-rev

ision-hash:324404111

9
server11:3.0.0.68        kube-system  calico-node-vdkk5
3.0.0.70         server13     k8s-app:calico-node  Running  calico-
node:fb3ec9429281 Fri Feb  8 01:50:50 2019

pod-template-generat          install-cni:b56980da7294

ion:1 controller-rev

ision-hash:324404111

9
server11:3.0.0.68        kube-system  calico-node-zzfkr
3.0.0.68         server11     k8s-app:calico-node  Running  calico-
node:c1ac399dd862 Fri Feb  8 01:50:50 2019

pod-template-generat          install-cni:60a779fdc47a

ion:1 controller-rev

ision-hash:324404111

9
server11:3.0.0.68        kube-system  etcd-server11
3.0.0.68         server11     tier:control-plane c Running  etcd:
dde63d44a2f5        Fri Feb  8 01:50:50 2019

omponent:etcd
server11:3.0.0.68        kube-system  kube-apiserver-hostd
3.0.0.68         server11     tier:control-plane c Running  kube-
apiserver:0cd557bbf Fri Feb  8 01:50:50 2019

-11                                                omponent:kube-
apiser          2fe

ver
```

```
server11:3.0.0.68        kube-system  kube-controller-mana
3.0.0.68         server11     tier:control-plane c Running  kube-
controller-manager: Fri Feb  8 01:50:50 2019
                                      ger-
server11                                       omponent:kube-
contro          89b2323d09b2

ller-manager
server11:3.0.0.68        kube-system  kube-dns-6f4fd4bdf-p
10.244.34.64     server23     k8s-app:kube-dns     Running  dnsmasq:
284d9d363999 kub Fri Feb  8 01:50:50 2019

lv7p
edns:bd8bdc49b950 sideca

r:fe10820ffb19
server11:3.0.0.68        kube-system  kube-proxy-4cx2t
3.0.3.197        server12     k8s-app:kube-proxy p Running  kube-
proxy:49b0936a4212  Fri Feb  8 01:50:50 2019

od-template-generati

on:1 controller-revi

sion-hash:3953509896
server11:3.0.0.68        kube-system  kube-proxy-7674k
3.0.3.196        server11     k8s-app:kube-proxy p Running  kube-
proxy:5dc2f5fe0fad  Fri Feb  8 01:50:50 2019

od-template-generati

on:1 controller-revi

sion-hash:3953509896
server11:3.0.0.68        kube-system  kube-proxy-ck5cn
3.0.2.5          server22     k8s-app:kube-proxy p Running  kube-
proxy:6944f7ff8c18  Fri Feb  8 01:50:50 2019

od-template-generati

on:1 controller-revi

sion-hash:3953509896
server11:3.0.0.68        kube-system  kube-proxy-f9dt8
3.0.0.68         server11     k8s-app:kube-proxy p Running  kube-
proxy:032cc82ef3f8  Fri Feb  8 01:50:50 2019

od-template-generati

on:1 controller-revi

sion-hash:3953509896
```

```
server11:3.0.0.68        kube-system  kube-proxy-j6qw6
3.0.5.135        server24     k8s-app:kube-proxy p Running  kube-
proxy:10544e43212e  Fri Feb  8 01:50:50 2019

od-template-generati

on:1 controller-revi

sion-hash:3953509896
server11:3.0.0.68        kube-system  kube-proxy-lq8zz
3.0.5.134        server23     k8s-app:kube-proxy p Running  kube-
proxy:1bcfa09bb186  Fri Feb  8 01:50:50 2019

od-template-generati

on:1 controller-revi

sion-hash:3953509896
server11:3.0.0.68        kube-system  kube-proxy-vg7kj
3.0.0.70         server13     k8s-app:kube-proxy p Running  kube-
proxy:8fed384b68e5  Fri Feb  8 01:50:50 2019

od-template-generati

on:1 controller-revi

sion-hash:3953509896
server11:3.0.0.68        kube-system  kube-scheduler-hostd
3.0.0.68         server11     tier:control-plane c Running  kube-
scheduler:c262a8071 Fri Feb  8 01:50:50 2019

-11                                                omponent:kube-
schedu          3cb

ler
server12:3.0.0.69        default      cumulus-frr-2gkdv
3.0.2.4          server21     pod-template-generat Running  cumulus-
frr:25d1109f8898 Fri Feb  8 01:50:50 2019

ion:1 name:cumulus-f

rr controller-revisi

on-hash:3710533951
server12:3.0.0.69        default      cumulus-frr-b9dm5
3.0.3.199        server14     pod-template-generat Running  cumulus-
frr:45063f9a095f Fri Feb  8 01:50:50 2019

ion:1 name:cumulus-f

rr controller-revisi
```

```
on-hash:3710533951
server12:3.0.0.69        default      cumulus-frr-rtqhv
3.0.2.6          server23     pod-template-generat Running  cumulus-
frr:63e802a52ea2 Fri Feb  8 01:50:50 2019

ion:1 name:cumulus-f

rr controller-revisi

on-hash:3710533951
server12:3.0.0.69        default      cumulus-frr-tddrg
3.0.5.133        server22     pod-template-generat Running  cumulus-
frr:52dd54e4ac9f Fri Feb  8 01:50:50 2019

ion:1 name:cumulus-f

rr controller-revisi

on-hash:3710533951
server12:3.0.0.69        default      cumulus-frr-vx7jp
3.0.5.132        server21     pod-template-generat Running  cumulus-
frr:1c20addfcbd3 Fri Feb  8 01:50:50 2019

ion:1 name:cumulus-f

rr controller-revisi

on-hash:3710533951
server12:3.0.0.69        default      cumulus-frr-x7ft5
3.0.3.198        server13     pod-template-generat Running  cumulus-
frr:b0f63792732e Fri Feb  8 01:50:50 2019

ion:1 name:cumulus-f

rr controller-revisi

on-hash:3710533951
server12:3.0.0.69        kube-system  calico-etcd-btqgt
3.0.0.69         server12     k8s-app:calico-etcd  Running  calico-
etcd:72b1a16968fb Fri Feb  8 01:50:50 2019

pod-template-generat

ion:1 controller-rev

ision-hash:142071906

5
server12:3.0.0.69        kube-system  calico-kube-controll
3.0.5.132        server21     k8s-app:calico-kube- Running  calico-
kube-controllers: Fri Feb  8 01:50:50 2019
```

```
                                      ers-d669cc78f-
bdnzk                                controllers
6821bf04696f
server12:3.0.0.69        kube-system  calico-node-4g6vd
3.0.3.198        server13     k8s-app:calico-node  Running  calico-
node:1046b559a50c Fri Feb  8 01:50:50 2019

pod-template-generat          install-cni:0a136851da17

ion:1 controller-rev

ision-hash:490828062
server12:3.0.0.69        kube-system  calico-node-4hg6l
3.0.0.69         server12     k8s-app:calico-node  Running  calico-
node:4e7acc83f8e8 Fri Feb  8 01:50:50 2019

pod-template-generat          install-cni:a26e76de289e

ion:1 controller-rev

ision-hash:490828062
server12:3.0.0.69        kube-system  calico-node-4p66v
3.0.2.6          server23     k8s-app:calico-node  Running  calico-
node:a7a44072e4e2 Fri Feb  8 01:50:50 2019

pod-template-generat          install-cni:9a19da2b2308

ion:1 controller-rev

ision-hash:490828062
server12:3.0.0.69        kube-system  calico-node-5z7k4
3.0.5.133        server22     k8s-app:calico-node  Running  calico-
node:9878b0606158 Fri Feb  8 01:50:50 2019

pod-template-generat          install-cni:489f8f326cf9

ion:1 controller-rev

ision-hash:490828062
...
```

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

```
cumulus@server11:~$ netq show kubernetes pod node server11
Matching kube_pod records:
Master                   Namespace    Name
IP               Node         Labels               Status
Containers               Last Changed
```

```
------------------------ ------------ --------------------
---------------- ------------ -------------------- --------
------------------------ ----------------
server11:3.0.0.68        kube-system  calico-etcd-pfg9r
3.0.0.68         server11     k8s-app:calico-etcd  Running  calico-
etcd:f95f44b745a7 2d:14h:0m:59s

pod-template-generat

ion:1 controller-rev

ision-hash:142071906

5
server11:3.0.0.68        kube-system  calico-node-zzfkr
3.0.0.68         server11     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
server11:3.0.0.68        kube-system  etcd-server11
3.0.0.68         server11     tier:control-plane c Running  etcd:
dde63d44a2f5        2d:14h:1m:44s

omponent:etcd
server11:3.0.0.68        kube-system  kube-apiserver-serve
3.0.0.68         server11     tier:control-plane c Running  kube-
apiserver:0cd557bbf 2d:14h:1m:44s

r11                                                omponent:kube-
apiser          2fe

ver
server11:3.0.0.68        kube-system  kube-controller-mana
3.0.0.68         server11     tier:control-plane c Running  kube-
controller-manager: 2d:14h:1m:44s
                                      ger-
server11                                       omponent:kube-
contro          89b2323d09b2

ller-manager
server11:3.0.0.68        kube-system  kube-proxy-f9dt8
3.0.0.68         server11     k8s-app:kube-proxy p Running  kube-
proxy:032cc82ef3f8  2d:14h:0m:59s

od-template-generati
```

```
on:1 controller-revi

sion-hash:3953509896
server11:3.0.0.68        kube-system  kube-scheduler-serve
3.0.0.68        server11     tier:control-plane c Running  kube-
scheduler:c262a8071 2d:14h:1m:44s

r11                                                 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 server11 show kubernetes node
Matching kube_cluster records:
Master                   Cluster Name     Node Name
Role       Status           Labels               Pod
CIDR                 Last Changed
------------------------ ---------------- --------------------
---------- ---------------- --------------------
------------------------ ----------------
server11:3.0.0.68        default          server11
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
server11:3.0.0.68        default          server13
worker     KubeletReady     kubernetes.io/hostna 10.224.3.0
/24            14h:19m:56s

me:server13 beta.kub

ernetes.io/arch:amd6

4 beta.kubernetes.io
```

```
/os:linux
server11:3.0.0.68        default          server22
worker     KubeletReady     kubernetes.io/hostna 10.224.1.0
/24            14h:24m:31s

me:server22 beta.kub

ernetes.io/arch:amd6

4 beta.kubernetes.io

/os:linux
server11:3.0.0.68        default          server11
worker     KubeletReady     kubernetes.io/hostna 10.224.2.0
/24            14h:24m:16s

me:server11 beta.kub

ernetes.io/arch:amd6

4 beta.kubernetes.io

/os:linux
server11:3.0.0.68        default          server12
worker     KubeletReady     kubernetes.io/hostna 10.224.4.0
/24            14h:24m:16s

me:server12 beta.kub

ernetes.io/arch:amd6

4 beta.kubernetes.io

/os:linux
server11:3.0.0.68        default          server23
worker     KubeletReady     kubernetes.io/hostna 10.224.5.0
/24            14h:24m:16s

me:server23 beta.kub

ernetes.io/arch:amd6

4 beta.kubernetes.io

/os:linux
server11:3.0.0.68        default          server24
worker     KubeletReady     kubernetes.io/hostna 10.224.6.0
/24            14h:24m:1s

me:server24 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 — *server11* in this case — was included in the query.

```
cumulus@server11:~$ netq server11 show kubernetes node components
Matching kube_cluster records:
                         Master           Cluster Name         Node
Name    Kubelet      KubeProxy         Container Runt

ime
------------------------ ---------------- --------------------
------------ ------------ ----------------- --------------
server11:3.0.0.68        default          server11             v1.
9.2       v1.9.2       docker://17.3.2   KubeletReady
server11:3.0.0.68        default          server13             v1.
9.2       v1.9.2       docker://17.3.2   KubeletReady
server11:3.0.0.68        default          server22             v1.
9.2       v1.9.2       docker://17.3.2   KubeletReady
server11:3.0.0.68        default          server11             v1.
9.2       v1.9.2       docker://17.3.2   KubeletReady
server11:3.0.0.68        default          server12             v1.
9.2       v1.9.2       docker://17.3.2   KubeletReady
server11:3.0.0.68        default          server23             v1.
9.2       v1.9.2       docker://17.3.2   KubeletReady
server11:3.0.0.68        default          server24             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@server11:~$ netq server11 show kubernetes node components
name server13
Matching kube_cluster records:
                         Master           Cluster Name         Node
Name    Kubelet      KubeProxy         Container Runt

ime
------------------------ ---------------- --------------------
------------ ------------ ----------------- --------------
server11:3.0.0.68        default          server13             v1.
9.2       v1.9.2       docker://17.3.2   KubeletReady
```

You can view information about the replica set:

```
cumulus@server11:~$ netq server11 show kubernetes replica-set
Matching kube_replica records:
Master                   Cluster Name Namespace        Replication
Name               Labels
Replicas                           Ready Replicas Last Changed
------------------------ ------------ ----------------
------------------------------ --------------------
---------------------------------- -------------- ----------------
server11:3.0.0.68        default      default          influxdb-
6cdb566dd             app:influx
1                                  1              14h:19m:28s
server11:3.0.0.68        default      default          nginx-
8586cf59                 run:nginx
3                                  3              14h:24m:39s
server11:3.0.0.68        default      default          httpd-
5456469bfd               app:httpd
1                                  1              14h:19m:28s
server11:3.0.0.68        default      kube-system      kube-dns-
6f4fd4bdf             k8s-app:kube-dns
1                                  1              14h:27m:9s
server11:3.0.0.68        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@server11:~$ netq server11 show kubernetes daemon-set
namespace default
Matching kube_daemonset records:
Master                   Cluster Name Namespace        Daemon Set
Name                Labels               Desired Count Ready Count
Last Changed
------------------------ ------------ ----------------
------------------------------ -------------------- -------------
----------- ----------------
server11:3.0.0.68        default      default          cumulus-
frr                    k8s-app:cumulus-frr   6             6
14h:25m:37s
```

You can view information about the pod:

```
cumulus@server11:~$ netq server11 show kubernetes pod namespace
default label nginx
Matching kube_pod records:
```

```
Master                   Namespace    Name
IP                Node         Labels               Status
Containers               Last Changed
------------------------ ------------ --------------------
---------------- ------------ -------------------- --------
------------------------ ----------------
server11:3.0.0.68        default      nginx-8586cf59-26pj5
10.244.9.193      server24     run:nginx            Running   nginx:
6e2b65070c86       14h:25m:24s
server11:3.0.0.68        default      nginx-8586cf59-c82ns
10.244.40.128     server12     run:nginx            Running   nginx:
01b017c26725       14h:25m:24s
server11:3.0.0.68        default      nginx-8586cf59-wjwgp
10.244.49.64      server22     run:nginx            Running   nginx:
ed2b4254e328       14h:25m:24s

cumulus@server11:~$ netq server11 show kubernetes pod namespace
default label app
Matching kube_pod records:
Master                   Namespace    Name
IP                Node         Labels               Status
Containers               Last Changed
------------------------ ------------ --------------------
---------------- ------------ -------------------- --------
------------------------ ----------------
server11:3.0.0.68        default      httpd-5456469bfd-bq9
10.244.49.65      server22     app:httpd            Running   httpd:
79b7f532be2d       14h:20m:34s
                                      zm
server11:3.0.0.68        default      influxdb-6cdb566dd-8
10.244.162.128    server13     app:influx           Running   influxdb:
15dce703cdec    14h:20m:34s
                                      9lwn
```

You can view information about the replication controller:

```
cumulus@server11:~$ netq server11 show kubernetes replication-
controller
No matching kube_replica records found
```

You can view information about a deployment:

```
cumulus@server11:~$ netq server11 show kubernetes deployment name
nginx
Matching kube_deployment records:
Master                   Namespace       Name
Replicas                           Ready Replicas
Labels                         Last Changed
```

```
------------------------ --------------- --------------------
---------------------------------- --------------
------------------------------ ----------------
server11:3.0.0.68        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@server11:~$ netq server11 show kubernetes replica-set label
kube
Matching kube_replica records:
Master                   Cluster Name Namespace        Replication
Name               Labels
Replicas                           Ready Replicas Last Changed
------------------------ ------------ ----------------
------------------------------ --------------------
---------------------------------- -------------- ----------------
server11:3.0.0.68        default      kube-system      kube-dns-
6f4fd4bdf             k8s-app:kube-dns
1                                  1              14h:30m:41s
server11:3.0.0.68        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@server11:~$ netq server11 show kubernetes deployment name
nginx connectivity
nginx -- nginx-8586cf59-wjwgp -- server22:swp1:torbond1 -- swp7:
hostbond3:torc-21
                              -- server22:swp2:torbond1 -- swp7:
hostbond3:torc-22
                              -- server22:swp3:NetQBond-2 -- swp20:
NetQBond-20:edge01
                              -- server22:swp4:NetQBond-2 -- swp20:
NetQBond-20:edge02
      -- nginx-8586cf59-c82ns -- server12:swp2:NetQBond-1 -- swp23:
NetQBond-23:edge01
```

```
                              -- server12:swp3:NetQBond-1 -- swp23:
NetQBond-23:edge02
                              -- server12:swp1:swp1 -- swp6:VlanA-1:
tor-1
      -- nginx-8586cf59-26pj5 -- server24:swp2:NetQBond-1 -- swp29:
NetQBond-29:edge01
                              -- server24:swp3:NetQBond-1 -- swp29:
NetQBond-29:edge02
                              -- server24:swp1:swp1 -- swp8:VlanA-1:
tor-2
```

View Kubernetes Service Connectivity and Impact

You can show the Kubernetes services in a cluster:

```
cumulus@server11:~$ netq show kubernetes service
Matching kube_service records:
Master                   Namespace        Service Name
Labels       Type       Cluster IP       External IP
Ports                               Last Changed
------------------------ ---------------- --------------------
------------ ---------- ---------------- ----------------
----------------------------------- ----------------
server11:3.0.0.68        default
kubernetes                        ClusterIP
10.96.0.1                         TCP:443
2d:13h:45m:30s
server11:3.0.0.68        kube-system      calico-etcd          k8s-
app:cali ClusterIP  10.96.232.136                    TCP:
6666                            2d:13h:45m:27s
                                                               co-etcd
server11:3.0.0.68        kube-system      kube-dns             k8s-
app:kube ClusterIP  10.96.0.10                       UDP:53 TCP:
53                       2d:13h:45m:28s
                                                               -dns
server12:3.0.0.69        default
kubernetes                        ClusterIP
10.96.0.1                         TCP:443
2d:13h:46m:24s
server12:3.0.0.69        kube-system      calico-etcd          k8s-
app:cali ClusterIP  10.96.232.136                    TCP:
6666                            2d:13h:46m:20s
                                                               co-etcd
server12:3.0.0.69        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@server11:~$ netq show kubernetes service name calico-etcd
Matching kube_service records:
Master                   Namespace        Service Name
Labels       Type       Cluster IP       External IP
Ports                               Last Changed
------------------------ ---------------- --------------------
------------ ---------- ---------------- ----------------
----------------------------------- ----------------
server11:3.0.0.68        kube-system      calico-etcd          k8s-
app:cali ClusterIP  10.96.232.136                 TCP:
6666                            2d:13h:48m:10s
                                                               co-etcd
server12:3.0.0.69        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@server11:~$ netq show kubernetes service name calico-etcd
connectivity
calico-etcd -- calico-etcd-pfg9r -- server11:swp1:torbond1 -- swp6:
hostbond2:torc-11
                                 -- server11:swp2:torbond1 -- swp6:
hostbond2:torc-12
                                 -- server11:swp3:NetQBond-2 -- swp16:
NetQBond-16:edge01
                                 -- server11:swp4:NetQBond-2 -- swp16:
NetQBond-16:edge02
calico-etcd -- calico-etcd-btqgt -- server12:swp1:torbond1 -- swp7:
hostbond3:torc-11
                                 -- server12:swp2:torbond1 -- swp7:
hostbond3:torc-12
                                 -- server12:swp3:NetQBond-2 -- swp17:
NetQBond-17:edge01
                                 -- server12:swp4:NetQBond-2 -- swp17:
NetQBond-17:edge02
```

To see the impact of a given Kubernetes service, run:

```
cumulus@server11:~$ netq server11 show impact kubernetes service name
calico-etcd
calico-etcd -- calico-etcd-pfg9r -- server11:swp1:torbond1 -- swp6:
hostbond2:torc-11
                                 -- server11:swp2:torbond1 -- swp6:
hostbond2:torc-12
                                 -- server11:swp3:NetQBond-2 -- swp16:
NetQBond-16:edge01
```

```
                                 -- server11:swp4:NetQBond-2 -- swp16:
NetQBond-16:edge02
```

View Kubernetes Cluster Configuration in the Past

You can use the "time machine" features (see page 187) of NetQ on a Kubernetes cluster, using the `around` keyword 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 *server23*. server23 is there because the node *server22* went down and Kubernetes spun up a third replica on a different host to satisfy the deployment requirement.

```
cumulus@redis-1:~$ netq server11 show kubernetes deployment name
nginx connectivity
nginx -- nginx-8586cf59-fqtnj -- server12:swp2:NetQBond-1 -- swp23:
NetQBond-23:edge01
                              -- server12:swp3:NetQBond-1 -- swp23:
NetQBond-23:edge02
                              -- server12:swp1:swp1 -- swp6:VlanA-1:
tor-1
      -- nginx-8586cf59-8g487 -- server24:swp2:NetQBond-1 -- swp29:
NetQBond-29:edge01
                              -- server24:swp3:NetQBond-1 -- swp29:
NetQBond-29:edge02
                              -- server24:swp1:swp1 -- swp8:VlanA-1:
tor-2
      -- nginx-8586cf59-2hb8t -- server23:swp1:swp1 -- swp7:VlanA-1:
tor-2
                              -- server23:swp2:NetQBond-1 -- swp28:
NetQBond-28:edge01
                              -- server23:swp3:NetQBond-1 -- swp28:
NetQBond-28:edge02
```

You can see this by going back in time 10 minutes. *server23* was not present, whereas *server22* **was** present:

```
cumulus@redis-1:~$ netq server11 show kubernetes deployment name
nginx connectivity around 10m
nginx -- nginx-8586cf59-fqtnj -- server12:swp2:NetQBond-1 -- swp23:
NetQBond-23:edge01
                              -- server12:swp3:NetQBond-1 -- swp23:
NetQBond-23:edge02
                              -- server12:swp1:swp1 -- swp6:VlanA-1:
tor-1
      -- nginx-8586cf59-2xxs4 -- server22:swp1:torbond1 -- swp7:
hostbond3:torc-21
                              -- server22:swp2:torbond1 -- swp7:
hostbond3:torc-22
```

```
                              -- server22:swp3:NetQBond-2 -- swp20:
NetQBond-20:edge01
                              -- server22:swp4:NetQBond-2 -- swp20:
NetQBond-20:edge02
      -- nginx-8586cf59-8g487 -- server24:swp2:NetQBond-1 -- swp29:
NetQBond-29:edge01
                              -- server24:swp3:NetQBond-1 -- swp29:
NetQBond-29:edge02
                              -- server24: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@server11:~$ netq torc-21 show impact kubernetes deployment
name nginx
nginx -- nginx-8586cf59-wjwgp -- server22:swp1:torbond1 -- swp7:
hostbond3:torc-21
                              -- server22:swp2:torbond1 -- swp7:
hostbond3:torc-22
                              -- server22:swp3:NetQBond-2 -- swp20:
NetQBond-20:edge01
                              -- server22:swp4:NetQBond-2 -- swp20:
NetQBond-20:edge02
      -- nginx-8586cf59-c82ns -- server12:swp2:NetQBond-1 -- swp23:
NetQBond-23:edge01
                              -- server12:swp3:NetQBond-1 -- swp23:
NetQBond-23:edge02
                              -- server12:swp1:swp1 -- swp6:VlanA-1:
tor-1
      -- nginx-8586cf59-26pj5 -- server24:swp2:NetQBond-1 -- swp29:
NetQBond-29:edge01
                              -- server24:swp3:NetQBond-1 -- swp29:
NetQBond-29:edge02
                              -- server24:swp1:swp1 -- swp8:VlanA-1:
tor-2
cumulus@server11:~$ netq server12 show impact kubernetes deployment
name nginx
nginx -- nginx-8586cf59-wjwgp -- server22:swp1:torbond1 -- swp7:
hostbond3:torc-21
                              -- server22:swp2:torbond1 -- swp7:
hostbond3:torc-22
                              -- server22:swp3:NetQBond-2 -- swp20:
NetQBond-20:edge01
                              -- server22:swp4:NetQBond-2 -- swp20:
NetQBond-20:edge02
      -- nginx-8586cf59-c82ns -- server12:swp2:NetQBond-1 -- swp23:
NetQBond-23:edge01
```

```
                              -- server12:swp3:NetQBond-1 -- swp23:
NetQBond-23:edge02
                              -- server12:swp1:swp1 -- swp6:VlanA-1:
tor-1
      -- nginx-8586cf59-26pj5 -- server24:swp2:NetQBond-1 -- swp29:
NetQBond-29:edge01
                              -- server24:swp3:NetQBond-1 -- swp29:
NetQBond-29:edge02
```

Manage NetQ Agents

At various points in time, you might want to change which network nodes are being monitored by NetQ or look more closely at a network node for troubleshooting purposes. Adding the NetQ Agent to a switch or host is described in Install NetQ. Disabling an Agent is described here and managing NetQ Agent logging is also presented.

Contents

This topic describes how to...

.

Modify the Configuration of the NetQ Agent on a Node

The agent configuration commands enable you to add and remove agents from switches and hosts, start and stop agent operations, add and remove Kubernetes container monitoring, add or remove sensors, debug the agent, and add or remove FRR (FRRouting).

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

This example shows how to specify the IP address and optionally a specific port on the NetQ Platform where agents should send their data.

```
cumulus@switch~:$ netq config add agent server 10.0.0.23
```

This example shows how to configure the agent to send sensor data.

```
cumulus@switch~:$ netq config add agent sensors
```

This example shows how to start monitoring with Kubernetes.

```
cumulus@switch:~$ netq config add kubernetes-monitor
```

> ⓘ After making configuration changes to your agents, you must restart the agent for the changes to take effect. Use the `netq config restart agent` command.

Disable the NetQ Agent on a Node

You can temporarily disable NetQ Agent on a node. Disabling the agent maintains the activity history in the NetQ database.

To disable NetQ Agent on a node, run the following command from the node:

```
cumulus@switch:~$ netq config stop agent
```

Remove the NetQ Agent from a Node

You can decommission a NetQ Agent on a given node. You might need to do this when you:

- RMA the switch or host being monitored
- Change the hostname of the switch or host being monitored
- Move the switch or host being monitored from one data center to another

> ⚠ Decommissioning the node removes the agent server settings from the local configuration file.

To decommission a node from the NetQ database:

1. On the given node, stop and disable the NetQ Agent service.

    ```
    cumulus@switch:~$ sudo systemctl stop netq-agent
    cumulus@switch:~$ sudo systemctl disable netq-agent
    ```

2. On the NetQ Appliance or Platform, decommission the node.

```
cumulus@netq-appliance:~$ netq decommission <hostname>
```

Configure Logging for a NetQ Agent

The logging level used for a NetQ Agent determines what types of events are logged about the NetQ Agent on the switch or host.

First, you need to decide what level of logging you want to configure. You can configure the logging level to be the same for every NetQ Agent, or selectively increase or decrease the logging level for a NetQ Agent on a problematic node.

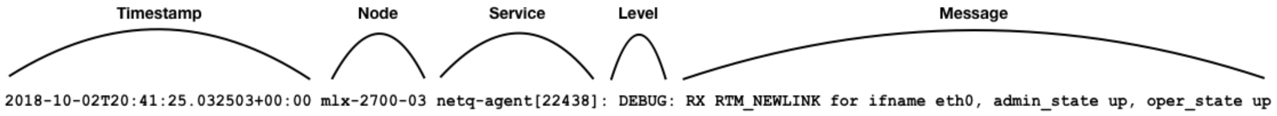
Logging Level	Description
debug	Sends notifications for all debugging-related, informational, warning, and error messages.
info	Sends notifications for informational, warning, and error messages (default).
warning	Sends notifications for warning and error messages.
error	Sends notifications for errors messages.

You can view the NetQ Agent log directly. Messages have the following structure:

`<timestamp> <node> <service>[PID]: <level>: <message>`

Element	Description
timestamp	Date and time event occurred in UTC format
node	Hostname of network node where event occurred
service [PID]	Service and Process IDentifier that generated the event
level	Logging level in which the given event is classified; *debug*, *error*, *info*, or *warning*
message	Text description of event, including the node where the event occurred

For example:



```
2018-10-02T20:41:25.032503+00:00 mlx-2700-03 netq-agent[22438]: DEBUG: RX RTM_NEWLINK for ifname eth0, admin_state up, oper_state up
```

This example shows a portion of a NetQ Agent log with debug level logging.

```
...
2019-02-16T18:45:53.951124+00:00 spine-1 netq-agent[8600]: INFO: OPTA
Discovery exhibit url hydra-09.cumulusnetworks.com port 4786
2019-02-16T18:45:53.952035+00:00 spine-1 netq-agent[8600]: INFO: OPTA
Discovery Agent ID spine-1
2019-02-16T18:45:53.960152+00:00 spine-1 netq-agent[8600]: INFO:
Received Discovery Response 0
2019-02-16T18:46:54.054160+00:00 spine-1 netq-agent[8600]: INFO: OPTA
Discovery exhibit url hydra-09.cumulusnetworks.com port 4786
2019-02-16T18:46:54.054509+00:00 spine-1 netq-agent[8600]: INFO: OPTA
Discovery Agent ID spine-1
2019-02-16T18:46:54.057273+00:00 spine-1 netq-agent[8600]: INFO:
Received Discovery Response 0
2019-02-16T18:47:54.157985+00:00 spine-1 netq-agent[8600]: INFO: OPTA
Discovery exhibit url hydra-09.cumulusnetworks.com port 4786
2019-02-16T18:47:54.158857+00:00 spine-1 netq-agent[8600]: INFO: OPTA
Discovery Agent ID spine-1
2019-02-16T18:47:54.171170+00:00 spine-1 netq-agent[8600]: INFO:
Received Discovery Response 0
2019-02-16T18:48:54.260903+00:00 spine-1 netq-agent[8600]: INFO: OPTA
Discovery exhibit url hydra-09.cumulusnetworks.com port 4786
...
```

Example: Configure debug-level logging

1. Set the logging level to *debug*.

   ```
   cumulus@switch:~$ netq config add agent loglevel debug
   ```

2. Restart the NetQ Agent.

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

3. Optionally, verify connection to the NetQ platform by viewing the `netq-agent.log` messages.

Example: Configure warning-level logging

```
cumulus@switch:~$ netq config add agent loglevel warning
cumulus@switch:~$ netq config restart agent
```

Example: Disable Agent Logging

If you have set the logging level to *debug* for troubleshooting, it is recommended that you either change the logging level to a less heavy mode or completely disable agent logging altogether when you are finished troubleshooting.

To change the logging level, run the following command and restart the agent service:

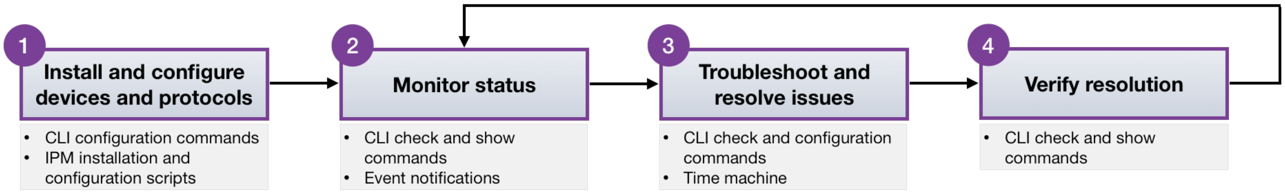
```
cumulus@switch:~$ netq config add agent loglevel <LOG_LEVEL>
cumulus@switch:~$ netq config restart agent
```

To disable all logging:

```
cumulus@switch:~$ netq config del agent loglevel
cumulus@switch:~$ netq config restart agent
```

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

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

NetQ provides a number of commands for diagnosing past events.

NetQ records network events and stores them in its database. You can view the events through a third-party notification application like PagerDuty or Slack or use `netq show events` 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@switch:~$ netq check bgp
```

```
Total Nodes: 25, Failed Nodes: 3, Total Sessions: 220 , Failed
Sessions: 24,
Hostname          VRF             Peer Name         Peer Hostname
Reason                                        Last Changed
----------------- --------------- ----------------- -----------------
---------------------------------------------
-------------------------
exit-1            DataVrf1080     swp6.2            firewall-1
BGP session with peer firewall-1 swp6.2: AFI/ 1d:7h:56m:9s

SAFI evpn not activated on peer
exit-1            DataVrf1080     swp7.2            firewall-2
BGP session with peer firewall-2 (swp7.2 vrf  1d:7h:49m:31s

DataVrf1080) failed,

reason: Peer not configured
exit-1            DataVrf1081     swp6.3            firewall-1
BGP session with peer firewall-1 swp6.3: AFI/ 1d:7h:56m:9s

SAFI evpn not activated on peer
exit-1            DataVrf1081     swp7.3            firewall-2
BGP session with peer firewall-2 (swp7.3 vrf  1d:7h:49m:31s

DataVrf1081) failed,

reason: Peer not configured
```

You can run a trace from spine01 to leaf02, which has the IP address 10.1.20.252:

```
cumulus@switch:~$ 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.

```
cumulus@switch:~$ netq show events type bgp
Matching events records:
Hostname          Message Type Severity
Message                             Timestamp
----------------- ------------ --------
----------------------------------- -------------------------
leaf21            bgp          info     BGP session with peer spine-1
swp3. 1d:8h:35m:19s
                                        3 vrf DataVrf1081 state
changed fro
                                        m failed to Established
leaf21            bgp          info     BGP session with peer spine-2
swp4. 1d:8h:35m:19s
```

```
                                        3 vrf DataVrf1081 state
changed fro
                                        m failed to Established
leaf21            bgp          info     BGP session with peer spine-3
swp5. 1d:8h:35m:19s
                                        3 vrf DataVrf1081 state
changed fro
                                        m failed to Established
leaf21            bgp          info     BGP session with peer spine-1
swp3. 1d:8h:35m:19s
                                        2 vrf DataVrf1080 state
changed fro
                                        m failed to Established
leaf21            bgp          info     BGP session with peer spine-3
swp5. 1d:8h:35m:19s
                                        2 vrf DataVrf1080 state
changed fro
                                        m failed to Established
...
```

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@switch:/$ netq show events between now and 48h
Matching events records:
Hostname          Message Type Severity
Message                             Timestamp
----------------- ------------ --------
----------------------------------- -------------------------
```

```
leaf21            configdiff   info     leaf21 config file ptm was
modified 1d:8h:38m:6s
leaf21            configdiff   info     leaf21 config file lldpd was
modifi 1d:8h:38m:6s
                                        ed
leaf21            configdiff   info     leaf21 config file interfaces
was m 1d:8h:38m:6s
                                        odified
leaf21            configdiff   info     leaf21 config file frr was
modified 1d:8h:38m:6s
leaf12            configdiff   info     leaf12 config file ptm was
modified 1d:8h:38m:11s
leaf12            configdiff   info     leaf12 config file lldpd was
modifi 1d:8h:38m:11s
                                        ed
leaf12            configdiff   info     leaf12 config file interfaces
was m 1d:8h:38m:11s
                                        odified
leaf12            configdiff   info     leaf12 config file frr was
modified 1d:8h:38m:11s
leaf11            configdiff   info     leaf11 config file ptm was
modified 1d:8h:38m:22s
...
```

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
```

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.

<table>
<tr><th>Component</th><th>Problem</th><th>Solution</th></tr>
<tr><td>Host</td><td>Where is this container located?
Open ports? What image is being used?
Which containers are part of this service? How are they connected?</td><td>netq show docker container
netq show docker container service</td></tr>
<tr><td>Overlay</td><td>Is my overlay configured correctly?
Can A reach B?</td><td>netq check|show vxlan
netq check evpn|Inv</td></tr>
<tr><td>L3</td><td>Is OSPF working as expected?
Is BGP working as expected?
Can IP A reach IP B?</td><td>netq check|show ospf
netq check|show bgp</td></tr>
<tr><td>L2</td><td>Is MLAG configured correctly?
Is there an STP loop?
Is VLAN or MTU misconfigured?
How does MAC A reach B?</td><td>netq check|show clag
netq show stp
netq check|show vlan
netq check mtu</td></tr>
<tr><td>OS</td><td>Are all switches licensed correctly?
Do all switches have NetQ agents running?</td><td>netq check license
netq check|show agents</td></tr>
<tr><td>Interfaces</td><td>Is my link down? Are all bond links up?
What optics am I using? What's the peer for this port?
Which ports are empty? Is there a link mismatch? Are links flapping?</td><td>netq show|check interfaces</td></tr>
<tr><td>Hardware</td><td>Have any components crashed?
What switches do I have in the network?</td><td>netq check sensors
netq show sensors all
netq show inventory brief</td></tr>
</table>

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 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 records:
Hostname          Peer              SysMac             State
Backup #Bond #Dual Last Changed

s
----------------- ----------------- ------------------ ----------
------ ----- ----- -------------------------
spine01(P)        spine02           00:01:01:10:00:01  up
up      24    24    Thu Feb  7 18:30:49 2019
spine02           spine01(P)        00:01:01:10:00:01  up
up      24    24    Thu Feb  7 18:30:53 2019
leaf01(P)         leaf02            44:38:39:ff:ff:01  up
up      12    12    Thu Feb  7 18:31:15 2019
leaf02            leaf01(P)         44:38:39:ff:ff:01  up
up      12    12    Thu Feb  7 18:31:20 2019
```

```
leaf03(P)         leaf04            44:38:39:ff:ff:02  up
up     12    12    Thu Feb  7 18:31:26 2019
leaf04            leaf03(P)         44:38:39:ff:ff:02  up
up     12    12    Thu Feb  7 18:31:30 2019
```

You can also verify a specific node is up:

```
cumulus@switch:~$ netq spine01 show clag
Matching clag records:
Hostname          Peer              SysMac             State
Backup #Bond #Dual Last Changed

s
----------------- ----------------- ------------------ ----------
------ ----- ----- -------------------------
spine01(P)        spine02           00:01:01:10:00:01  up
up     24    24    Thu Feb  7 18:30:49 2019
```

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@switch:/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 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: spine01 hostbond5
2017-05-22T23:14:53.081480+00:00 noc-pr netq-notifier[5501]: WARNING:
CLAG: 1 node(s) have failures. They are: spine01
2017-05-22T23:14:58.161267+00:00 noc-pr netq-notifier[5501]: WARNING:
CLAG: 2 node(s) have failures. They are: spine01, leaf01
```

To begin your investigation, show the status of the `clagd` service:

```
cumulus@switch:~$ netq spine01 show services clagd

Matching services records:
Hostname          Service              PID   VRF             Enabled
Active Monitored Status           Uptime                    Last
Changed
----------------- -------------------- ----- --------------- -------
------ --------- ---------------- -------------------------
-------------------------
spine01           clagd                2678  default         yes
yes    yes       ok               23h:57m:16s               Thu Feb
7 18:30:49 2019
```

Checking the MLAG status provides the reason for the failure:

```
cumulus@switch:~$ netq check clag
Checked Nodes: 6, Warning Nodes: 2
Node             Reason
----------------
----------------------------------------------------------------------
----
spine01          Link Down: hostbond5
leaf01           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": "spine01",
            "reason": "Link Down: hostbond5"
        }
        ,
        {
            "node": "lea01",
            "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 indicate all nodes are UP, and the `netq check` flag also indicates there are no failures.

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

Matching clag records:
Hostname          Peer              SysMac             State
Backup #Bond #Dual Last Changed

s
----------------- ----------------- ------------------ ----------
------ ----- ----- -------------------------
spine01(P)        spine02           00:01:01:10:00:01  up
up     24    24     Thu Feb  7 18:30:49 2019
spine02           spine01(P)        00:01:01:10:00:01  up
up     24    24     Thu Feb  7 18:30:53 2019
leaf01(P)         leaf02            44:38:39:ff:ff:01  up
up     12    12     Thu Feb  7 18:31:15 2019
leaf02            leaf01(P)         44:38:39:ff:ff:01  up
up     12    12     Thu Feb  7 18:31:20 2019
leaf03(P)         leaf04            44:38:39:ff:ff:02  up
up     12    12     Thu Feb  7 18:31:26 2019
leaf04            leaf03(P)         44:38:39:ff:ff:02  up
up     12    12     Thu Feb  7 18:31:30 2019
```

When you are logged directly into a switch, you can run `clagctl` to get the state:

```
cumulus@switch:/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: spine01, leaf01
2017-05-22T23:16:51.525403+00:00 noc-pr netq-notifier[5501]: WARNING:
LINK: 2 link(s) are down. They are: leaf01 vx-37, spine01 vx-37
2017-05-22T23:16:54.194681+00:00 noc-pr netq-notifier[5501]: WARNING:
LNV: 1 node(s) have failures. They are: leaf02
2017-05-22T23:16:59.448755+00:00 noc-pr netq-notifier[5501]: WARNING:
LNV: 3 node(s) have failures. They are: leaf01, leaf03, leaf04
2017-05-22T23:17:04.703044+00:00 noc-pr netq-notifier[5501]: WARNING:
CLAG: 2 node(s) have failures. They are: spine01, leaf01
```

To begin your investigation, show the status of the `clagd` service:

```
cumulus@switch:~$ netq spine01 show services clagd

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

```
spine01           clagd                2678  default         yes
yes    yes       error            23h:57m:16s               Thu Feb
7 18:30:49 2019
```

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
----------------
----------------------------------------------------------------------
----
spine01          Protodown Bonds: vx-37:vxlan-single
leaf01           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": "spine01",
            "reason": "Protodown Bonds: vx-37:vxlan-single"
        }
        ,
        {
            "node": "leaf01",
            "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:
Hostname          Peer              SysMac             State
Backup #Bond #Dual Last Changed

s
----------------- ----------------- ------------------ ----------
------ ----- ----- -------------------------
```

```
spine01(P)        spine02           00:01:01:10:00:01  up
up     24     24    Thu Feb  7 18:30:49 2019
spine02           spine01(P)        00:01:01:10:00:01  up
up     24     24    Thu Feb  7 18:30:53 2019
leaf01(P)         leaf02            44:38:39:ff:ff:01  up
up     12     12    Thu Feb  7 18:31:15 2019
leaf02            leaf01(P)         44:38:39:ff:ff:01  up
up     12     12    Thu Feb  7 18:31:20 2019
leaf03(P)         leaf04            44:38:39:ff:ff:02  up
up     12     12    Thu Feb  7 18:31:26 2019
leaf04            leaf03(P)         44:38:39:ff:ff:02  up
up     12     12    Thu Feb  7 18:31:30 2019
```

When you are logged directly into a switch, you can run `clagctl` to get the state:

```
cumulus@switch:/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: leaf01
```

```
2017-05-22T23:51:19.622379+00:00 noc-pr netq-notifier[5501]: WARNING:
LINK: 2 link(s) flapped and are down. They are: leaf01 hostbond5,
leaf01 hostbond4
2017-05-22T23:51:19.622922+00:00 noc-pr netq-notifier[5501]: WARNING:
LINK: 23 link(s) are down. They are: leaf01 VlanA-1-104-v0, leaf01
VlanA-1-101-v0, leaf01 VlanA-1, leaf01 vx-33, leaf01 vx-36, leaf01 vx-
37, leaf01 vx-34, leaf01 vx-35, leaf01 swp7, leaf01 VlanA-1-102-v0,
leaf01 VlanA-1-103-v0, leaf01 VlanA-1-100-v0, leaf01 VlanA-1-106-v0,
leaf01 swp8, leaf01 VlanA-1.106, leaf01 VlanA-1.105, leaf01 VlanA-
1.104, leaf01 VlanA-1.103, leaf01 VlanA-1.102, leaf01 VlanA-1.101,
leaf01 VlanA-1.100, leaf01 VlanA-1-105-v0, leaf01 vx-38
2017-05-22T23:51:27.696572+00:00 noc-pr netq-notifier[5501]: INFO:
LINK: 15 link(s) are up. They are: leaf01 VlanA-1.106, leaf01 VlanA-1-
104-v0, leaf01 VlanA-1.104, leaf01 VlanA-1.103, leaf01 VlanA-1.101,
leaf01 VlanA-1-100-v0, leaf01 VlanA-1.100, leaf01 VlanA-1.102, leaf01
VlanA-1-101-v0, leaf01 VlanA-1-102-v0, leaf01 VlanA-1.105, leaf01
VlanA-1-103-v0, leaf01 VlanA-1-106-v0, leaf01 VlanA-1, leaf01 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: leaf01
2017-05-22T23:51:36.156708+00:00 noc-pr netq-notifier[5501]: WARNING:
CLAG: 2 node(s) have failures. They are: spine01, leaf01
2017-05-22T23:51:36.183638+00:00 noc-pr netq-notifier[5501]: WARNING:
LNV: 2 node(s) have failures. They are: spine02, leaf01
2017-05-22T23:51:41.444670+00:00 noc-pr netq-notifier[5501]: WARNING:
LNV: 1 node(s) have failures. They are: leaf01
```

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
---------------- ---------------- ----------------- ----- ------
------ ----- -------------------------
spine01(P)       spine02           00:01:01:10:00:01 up    up
9      9     Thu Feb  7 18:30:53 2019
spine02          spine01(P)        00:01:01:10:00:01 up    up
9      9     Thu Feb  7 18:31:04 2019
leaf01                             44:38:39:ff:ff:01 down  n/a
0      0     Thu Feb  7 18:31:13 2019
leaf03(P)        leaf04            44:38:39:ff:ff:02 up    up
8      8     Thu Feb  7 18:31:19 2019
leaf04           leaf03(P)         44:38:39:ff:ff:02 up    up
8      8     Thu Feb  7 18:31:25 2019
```

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
----------------
----------------------------------------------------------------------
----
spine01              Peer Connectivity failed
leaf01               Peer Connectivity failed
```

You can retrieve the output in JSON format for export to another tool:

```
cumulus@switch:~$ netq check clag json
{
    "failedNodes": [
        {
            "node": "spine01",
            "reason": "Peer Connectivity failed"
        }
        ,
        {
            "node": "leaf01",
            "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:

```
cumulus@switch:~$ sudo 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 paths you can take to locate and investigate issues that occur in the NetQ software itself, including viewing configuration and log files, verifying NetQ Agent health, and verifying NetQ Platform 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 the following configuration and log files that can provide insight into the root cause of the issue:

File	Description
`/etc/netq/netq.yml`	The NetQ configuration file. This file appears only if you installed either the **`netq-apps`** package or the NetQ Agent on the system.
`/var/log/netqd.log`	The NetQ daemon log file for the NetQ CLI. This log file appears only if you installed the **`netq-apps`** package on the system.
`/var/log/netq-agent.log`	The NetQ Agent log file. This log file appears only if you installed the NetQ Agent on the system.

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:

```
cumulus@switch:$ 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
```

Generate a Support File

The `opta-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 NetQ Platform 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:

```
cumulus@switch:~$ opta-support
```

Early Access Features

NetQ has early access features that provide advanced access to new functionality before it becomes generally available. The ability to view physical interface statistics collected by the NetQ Agent is the only early access feature available in NetQ 2.1.2.

This feature is bundled into the `netq-apps` package; there is no specific EA package like there typically is with Cumulus Linux.

Contents

This topic describes how to...

Enable Early Access Features

You enable early access features by running the `netq config add experimental` command on any node running NetQ.

```
cumulus@switch:~$ netq config add experimental
Experimental config added
```

View Interface Statistics

NetQ Agents collect performance statistics every 30 seconds for the physical interfaces on switches and hosts in your network. The NetQ Agent does not collect statistics for non-physical interfaces, such as bonds, bridges, and VXLANs. After enabling the feature, the NetQ Agent collects the following 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

These can be viewed using the following NetQ CLI command:

```
netq [<hostname>] show interface-stats [errors | all] [<physical-
port>] [around <text-time>] [json]
```

Use the *hostname* option to limit the output to a particular switch. Use the *errors* option to view only the transmit and receive errors found on the designated interfaces. Use the *physical-port* option to limit the output to a particular port. Use the *around* option to view the data at a time in the past.

In this example, we view the interface statistics for all switches and all of their physical interfaces.

```
cumulus@switch:~$ netq show interface-stats
```

```
Matching proc_dev_stats records:
Hostname          Interface                 Duration         RX
Bytes             RX Drop              RX Errors           TX
Bytes             TX Drop              TX Errors           Last
Changed
----------------- ------------------------- ----------------
-------------------- -------------------- --------------------
-------------------- -------------------- --------------------
-------------------------
edge01            eth0                      30
2278                 0                    16
4007                 0                    0                    Mon
Jun  3 23:03:14 2019
edge01            lo                        30
864                  0                    0
864                  0                    0                    Mon
Jun  3 23:03:14 2019
exit01            bridge                    60
336                  0                    0
1176                 0                    0                    Mon
Jun  3 23:02:27 2019
exit01            eth0                      30
3424                 0                    0
6965                 0                    0                    Mon
Jun  3 23:02:58 2019
exit01            mgmt                      30
2682                 0                    0
7488                 0                    0                    Mon
Jun  3 23:02:58 2019
exit01            swp44                     30
2457                 0                    0
2457                 0                    0                    Mon
Jun  3 23:02:58 2019
exit01            swp51                     30
2462                 0                    0
1769                 0                    0                    Mon
Jun  3 23:02:58 2019
exit01            swp52                     30
2634                 0                    0
2629                 0                    0                    Mon
Jun  3 23:02:58 2019
exit01            vlan4001                  50
336                  0                    0
1176                 0                    0                    Mon
Jun  3 23:02:27 2019
exit01            vrf1                      60
1344                 0                    0
0                    0                    0                    Mon
Jun  3 23:02:27 2019
```

```
exit01            vxlan4001                 50
336                  0                    0
1368                 0                    0                   Mon
Jun  3 23:02:27 2019
exit02            bridge                    61
1008                 0                    0
392                  0                    0                   Mon
Jun  3 23:03:07 2019
exit02            eth0                      20
2711                 0                    0
4983                 0                    0                   Mon
Jun  3 23:03:07 2019
exit02            mgmt                      30
2162                 0                    0
5506                 0                    0                   Mon
Jun  3 23:03:07 2019
exit02            swp44                     20
3040                 0                    0
3824                 0                    0                   Mon
Jun  3 23:03:07 2019
...
```

Disable Early Access Features

You disable the early access features by running the `netq config del experimental` command on any node running NetQ.

```
cumulus@switch:~$ netq config del experimental
Experimental config deleted
```