CUMULUS

Cumulus NetQ 1.0.0

User Guide

Table of Contents

NetQ Components 5

Getting Started with NetQ 7
Contents 8
Install the NetQ Telemetry Server 8
Install the NetQ Agent 9
Installing on a Cumulus Linux Switch 10
Installing on an Ubuntu Server 10
Installing on a Red Hat or CentOS Server 10
Configuring the NetQ Agent on a Node 10
Configuring the Agent to Use a VRF 11
Configuring the Agent to Communicate over a Specific Port 12
Configuring NetQ Notifier on the Telemetry Server 12
Example /etc/netq/netq.yml Configuration 14

Getting to Know NetQ 18
Getting Information about Network Hardware 20
Using the NetQ Shell on the NetQ Telemetry Server 21
Using the netq resolve Command 22
Sample Commands for Various Components 25

Taking Preventative Steps with Your Network 26
Contents 27
netq check and netq show 27
netq show agents 29
Using NetQ with Automation 29
Using NetQ Virtual 30

Proactively Monitoring the Network Fabric 30
Contents 31
NetQ Notifier 31
Log Message Format 32
Supported Third-party Applications 32
Early Access Support 34
Exporting to ELK 35
Exporting to Splunk 35
Precisely Locating an Issue on the Network 36
Extending NetQ with Custom Services Using curl 37
Exporting NetQ Data 38
MLAG Troubleshooting with NetQ 39
Contents 39

All Nodes Are Up . . . 40
Dual-connected Bond Is Down . . . 41
VXLAN Active-active Device or Interface Is Down . . . 43
Remote-side clagd Stopped by systemctl Command . . . 45

Performing Network Diagnostics . . . 47
Contents . . . 48
Diagnosing an Event after It Occurs . . . 48
Using NetQ as a Time Machine . . . 50
How Far Back in Time Can You Travel? . . . 51
Using trace in a VRF . . . 52

NetQ Service Console . . . 52
Contents . . . 53
Connecting to the Service Console . . . 53
Configuring Users . . . 53
Other User Account Actions . . . 54
Accessing the NetQ Command Line . . . 54

Monitoring Linux Hosts with NetQ . . . 56

Monitoring Container Environments with NetQ . . . 58
Contents . . . 59
NetQ Container Support . . . 59
Telemetry Server Memory Requirement . . . 60
Configuring the Container Host . . . 60
Starting and Stopping Containers . . . 61
Showing Container Summary Information . . . 61
Identifying Containers on the Network . . . 62
Showing Container Adjacency . . . 67
Showing Container-specific Information . . . 71
Showing Containers with a Specific Image . . . 71
Showing Container Connectivity . . . 74
Checking Network Traffic over a Given Protocol . . . 77

Configuring the NetQ Virtual Environment . . . 78
Contents . . . 79
Setting up the Demo Environment with Vagrant and VirtualBox . . . 79

Restoring from Backups with NetQ . . . 80
Backup Locations . . . 81
Use Cases . . . 81
Restoring from a Snapshot . . . 81

Troubleshooting NetQ . . . 82

Checking Agent Health 83
Error Configuring the Telemetry Server on a Node 84
netq-support 85

Index 86

Cumulus NetQ is a fabric-wide, telemetry-based validation system, that enables organizations to validate network state, both during regular operations and for post-mortem diagnostic analysis. Running on Cumulus Linux switches and other certified systems — such as Ubuntu, Red Hat, and CentOS hosts — NetQ captures network data and other state information in real time, allowing cloud architects and network operations teams to operate with visibility over the entire network.

The system uses a three-pronged approach to validating networks:

- **Preventative**

  NetQ easily validates potential network configuration changes in a virtualized environment or lab using check, show and trace algorithms, eliminating the need to check nodes one by one and reducing manual errors before they are rolled into production (one of the main causes of network downtime).

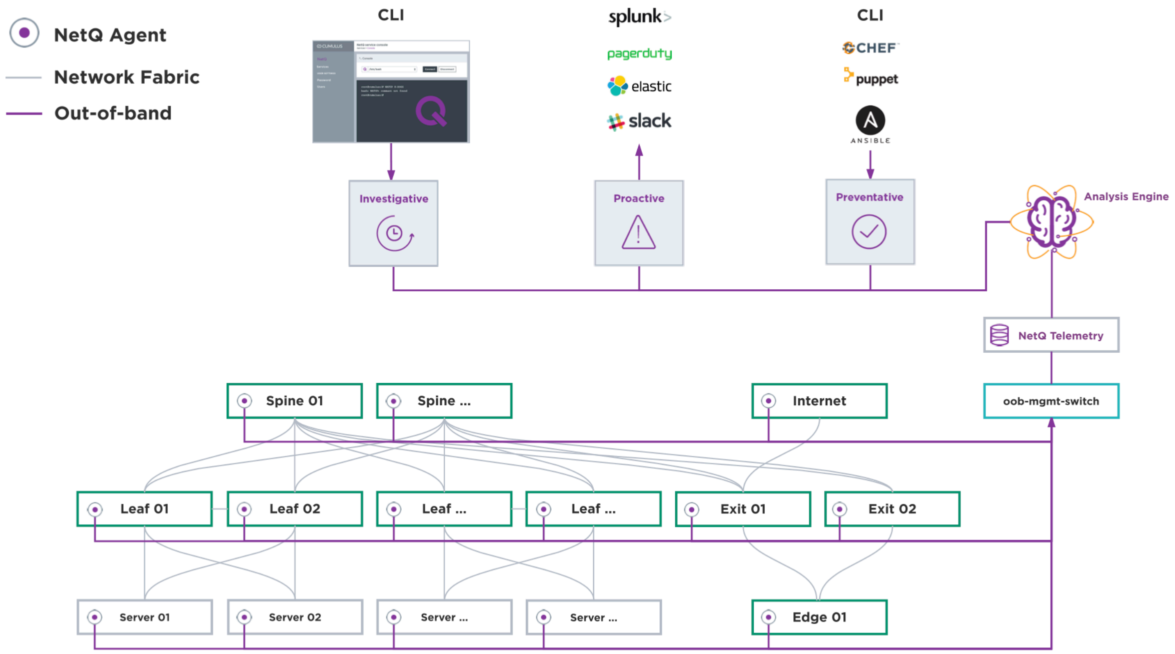
- **Proactive**

  NetQ detects faulty network states that can result in packet loss or connectivity issues, and alerts the user with precise fault location data to allow for faster remediation, greatly improving network agility, and reducing downtime costs.

- **Diagnostic**

  NetQ provides the ability to trace network paths, replay the network state at a time in the past, review fabric-wide event changelogs and diagnose the root cause of state deviations.

NetQ Components



NetQ comprises the following components:

- **NetQ Agent**

  The back-end Python agent installed on every monitored *node* in the network — including Cumulus Linux switches, Linux bare-metal hosts and virtual machines , or Docker containers. The agent pushes out data to the NetQ Telemetry Server periodically, and when specific `netlink` events occur. The agent monitors the following objects via `netlink`:

  - interfaces
  - address (IPv4 and IPv6)
  - route (IPv4 and IPv6)
  - link
  - bridge fdb
  - IP neighbor (IPv4 and IPv6)

Further, every 15 seconds, it gathers data for the following protocols:

- Bridging protocols (LLDP, STP, MLAG)
- Routing protocols (BGP, OSPF)
- Network virtualization (LNV, VXLAN data plane)
- Docker containers

It also listens to the Docker event stream to monitor Docker containers running on a host and gathers container networking information such as NAT translations, networks and container IP and MAC addresses.

- **NetQ Telemetry Server**

  The database/key-value store where all network information sent from NetQ Agents running on the network is collected, aggregated and queried from.
- **NetQ Analysis Engine**

  The NetQ Analysis Engine is the backend engine utilized when querying NetQ via the CLI, service console, or notifier. The engine has two parts:
  - The **NetQ Agent Command Line Interface**. The NetQ CLI can be used on every node and can be used on the NetQ Telemetry Server through `netq-shell`.
  - The **NetQ Notifier**. The notifier runs on the telemetry server. It responds to events pushed by the NetQ Agent, sending alerts to a configured channel, such as Slack, PagerDuty or `syslog`.
- **NetQ Service Console**

  The Service Console provides a browser-based window for accessing the NetQ CLI from anywhere.

Getting Started with NetQ

NetQ is comprised of two main install components: the NetQ Telemetry Server, and the `cumulus-netq` metapackage which gets installed on Cumulus Linux switches. Additionally, for host network visibility and containers, you can install host OS-specific metapackages.

This section walks through the basic install and setup steps for installing and running NetQ on the following supported operating systems:

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

✓ Before you get started, you should review the release notes for this version.

Contents

This chapter covers ...

- Install the NetQ Telemetry Server (see page 8)
- Install the NetQ Agent (see page 9)
  - Installing on a Cumulus Linux Switch (see page 10)
  - Installing on an Ubuntu Server (see page 10)
  - Installing on a Red Hat or CentOS Server (see page 10)
- Configuring the NetQ Agent on a Node (see page 10)
  - Configuring the Agent to Use a VRF (see page 11)
  - Configuring the Agent to Communicate over a Specific Port (see page 12)
- Configuring NetQ Notifier on the Telemetry Server (see page 12)
- Example /etc/netq/netq.yml Configuration (see page 14)

Install the NetQ Telemetry Server

The NetQ Telemetry Server is a VMware ESXi 6.5 virtual machine, comprising a set of individual Docker containers that each contain a separate service for the `redis` database, the Service Console, the NetQ CLI and NetQ Notifier.

> ⚠ Cumulus Networks recommends the telemetry server is installed on an out-of-band management network to ensure it can monitor in-band network issues without being affected itself. Ideally, you should run the telemetry server on a separate, powerful server for maximum usability and performance. For more information on system requirements, refer to the How Far Back in Time Can You Travel (see page 51) section.

> ⚠ The NetQ telemetry server containers are completely separate from any containers you may have on the hosts you are monitoring with NetQ. The NetQ containers will not overwrite the host containers and vice versa.

1. Download the NetQ Telemetry Server VMware virtual machine. Select *NetQ* from the **Product** menu on the Downloads page.
2. Import the virtual machine into your hypervisor.
3. Start the NetQ Telemetry Server. There are two default user accounts you can use to log in:
   - The primary username is *admin*, and the default password is *CumulusNetQ!*.
   - The alternate username is *cumulus*, and its password is *CumulusLinux!*.

Once the NetQ Telemetry Server is installed, you need to configure NetQ Notifier.

In addition, if you intend to use NetQ with applications like PagerDuty or Slack, you need to configure those applications to receive notifications from NetQ Notifier.

> ✓ Note the external IP address of the host where the telemetry server is running, as you need this to correctly configure the NetQ Agent on every node you want to monitor. The telemetry server gets its IP address from DHCP; to get the IP address, run `ifconfig eth0` on the telemetry server.

Install the NetQ Agent

In order to manage a node with NetQ Agent and send notifications with NetQ Notifier, you need to install an OS-specific metapackage on each node. The node can be a:

- Cumulus Linux switch running version 3.3.0 or later
- Server running Red Hat RHEL 7.1, Ubuntu 16.04 or CentOS 7
- Linux virtual machine running one of the above Linux operating systems

The metapackage contains the NetQ Agent and the NetQ command line interface.

Install the metapackage on each node to monitor, then configure the NetQ Agent on the node.

Installing on a Cumulus Linux Switch

- Update the local `apt` repository, then install the metapackage on the switch:

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

Installing on an Ubuntu Server

- Reference and update the local `apt` repository, then install the metapackage on the server:

```
root@ubuntu:~# wget -O- https://hostapps3.cumulusnetworks.com
/setup/cumulus-host-ubuntu.pubkey | apt-key add -
root@ubuntu:~# wget -O- https://hostapps3.cumulusnetworks.com
/setup/cumulus-host-ubuntu-xenial.list > /etc/apt/sources.list.d
/cumulus-host-ubuntu-xenial.list
root@ubuntu:~# apt-get update ; apt-get install cumulus-netq
```

Installing on a Red Hat or CentOS Server

- Reference and update the local `yum` repository, then install the metapackage on the server:

```
root@rhel7:~# rpm --import https://hostapps3.cumulusnetworks.com
/setup/cumulus-host-el.pubkey
root@rhel7:~# wget -O- https://hostapps3.cumulusnetworks.com
/setup/cumulus-host-el.repo > /etc/yum.repos.d/cumulus-host-el.
repo
root@rhel7:~# yum install cumulus-netq
```

Configuring the NetQ Agent on a Node

Once you install the NetQ packages and configure the NetQ Telemetry Server, you need to configure NetQ on each node (Cumulus Linux switch or Linux host) to monitor that node on your network.

1. To ensure useful output, ensure that NTP is running.
2. On the host, after you install the NetQ metapackage, restart `rsyslog` so logs are sent to the correct destination:

```
cumulus@switch:~$ sudo systemctl restart rsyslog
```

3. **CentOS, RHEL or Ubuntu hosts only:** Enable and restart the `netqd` service:

```
cumulus@server01:~$ sudo systemctl enable netqd ; sudo systemctl
start netqd
```

4. Link the host to the telemetry server you configured above; in the following example, the IP address for the telemetry server host is *198.51.100.10*:

```
cumulus@switch:~$ netq add server 198.51.100.10
```

   This command updated the configuration in the `/etc/netq/netq.yml` file. It also enables the NetQ CLI.

5. **Container hosts only:** Enable Docker by adding the following three lines to the `netq.yml` file on the container host:

```
cumulus@server01:~$ sudo vi /etc/netq/netq.yml

...

docker:
  enable: true
  poll_period: 15
```

6. Restart the `netq` services.

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

> ⚠ If you see the following error, it means you haven't added the telemetry server or the server wasn't configured:
>
> ```
> cumulus@switch:~$ netq agent start
> Error: Please specify IP address of DB server
> ```

Configuring the Agent to Use a VRF

If you want the NetQ Agent to communicate with the telemetry server only via a VRF, including a management VRF, you need to specify the VRF name when configuring the NetQ Agent. For example, if the management VRF is configured and you want the agent to communicate with the telemetry server over it, configure the agent like this:

```
cumulus@switch:~$ netq add server 198.51.100.10 vrf mgmt
```

Configuring the Agent to Communicate over a Specific Port

By default, NetQ uses port 6379 for communication between the telemetry server and NetQ Agents. If you want the NetQ Agent to communicate with the telemetry server via a different port, you need to specify the port number when configuring the NetQ Agent like this:

```
cumulus@switch:~$ netq add server 198.51.100.10 port 7379
```

Configuring NetQ Notifier on the Telemetry Server

NetQ Notifier listens to events from the telemetry server database. When NetQ Notifier is running on the NetQ Telemetry Server, it sends out alerts. NetQ Notifier runs on the NetQ Telemetry Server only; the NetQ Agents on the nodes only communicate with it.

You need to configure two things for NetQ Notifier:

- The events for which you want to receive notifications/alerts, like sensors or BGP session notifications.
- The integrations for where to send those notifications; by default, they are `rsyslog`, PagerDuty and Slack.

NetQ Notifier sends out alerts based on the configured log level, which is one of the following:

- debug: Used for debugging-related messages.
- info: Used for informational, high-volume messages.
- warning: Used for warning conditions.
- error: Used for error conditions.

The default log level setting is *info*, so NetQ Notifier sends out alerts for info, warning and error conditions.

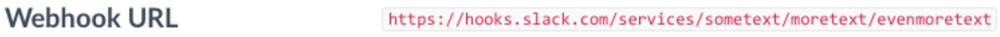
By default, all notifications/alerts are enabled, and logged in `/var/log/docker/netq/notifier_1/netq-notifier.log`. You only need to edit the notifications if there is something you don't want to monitor.

NetQ Notifier is already integrated with `rsyslog`. To integrate with PagerDuty or Slack, you need to specify some parameters.

To configure alerts and integrations on the NetQ Telemetry Server:

1. As the sudo user, open `/appliance/cfg/netq/netq.yml` in a text editor.

2. Configure the following in the `/appliance/cfg/netq/netq.yml` file:
    - Change the log level: If you want a more restrictive level than info.
    - Configure application notifications: To customize any notifications, uncomment the relevant section under **netq-notifier Configurations** and make changes accordingly.
    - Configure PagerDuty and Slack integrations. You can see where to input the information for these integrations in the example `netq.yml` file (see page 14) below.
        - For PagerDuty, enter the API access key (also called the authorization token) and the integration key (also called the service_key or routing_key).
        - For Slack, enter the webhook URL. To get the webhook URL, in the Slack dropdown menu, click **Apps & integrations**, then click **Manage** > **Custom Integrations** > **Incoming WebHooks** > select **Add Configuration** > select the channel to receive the notifications such as *#netq-notifier* in the **Post to Channel** dropdown > then click **Add Incoming WebHook integration** the URL produced by Slack will look something like the one pictured below:



Webhook URL https://hooks.slack.com/services/sometext/moretext/evenmoretext

Copy the URL from the Webhook URL field into the `/appliance/cfg/netq/netq.yml` file under the **Slack Notifications** section. Uncomment the Slack, enable, and webhook lines while adding the webhook URL value provided by Slack.

```
## Slack Notifications
##
## netq-notifier sends notifications to Slack using
Incoming Webhooks.
## The webhook for your channel can be found or
created on Slack at:
## Apps -> Custom Integrations -> Incoming Webhooks.
## Each webhook has a 'Webhook URL'. Please specify
that for webhook.
##
## enable: true or false
## webhook:
##
slack:
  enable: true
  webhook: https://hooks.slack.com/services/sometext
/moretext/evenmoretext
```

When you are finished editing the file, save and close it.

3. Stop then start the NetQ Notifier daemon to apply the new configuration:

```
cumulus@netq-appliance:~$ docker exec -it netq_netq-notifier_1
systemctl stop netq-notifier
cumulus@netq-appliance:~$ docker exec -it netq_netq-notifier_1
systemctl start netq-notifier
```

Example /etc/netq/netq.yml Configuration

In the following sample `/etc/netq/netq.yml` file, notice that the NetQ Telemetry Server is on a server with the IP address 198.51.100.10.

```
cumulus@netq-appliance:~$ cat /etc/netq/netq.yml
## Netq configuration File.
## Configuration is also read from files in /etc/netq/config.d/ and
have
## precedence over config in /etc/netq/netq.yml.
## ----- Common configurations -----
## Backend Configuration for all netq agents and apps on this host.
##
backend:
  server: 198.51.100.10
#  port: 6379
## ----- netq-agent configurations -----
## Netq Agent Configuration
##
## log_level: Could be debug, info, warning or error. Default is info.
##
#netq-agent:
#  log_level: info
## Docker Agent Configuration
##
## docker_enable: Enable Docker monitoring. Default is True.
## docker_poll_period: Docker poll period in secs. Default is 15 secs.
##
#docker:
#  enable: true
#  poll_period: 15
## ----- netq configurations -----
## Netq configuration
##
## log_level: Could be debug, info, warning or error. Default is info.
##
#netqd:
#  log_level: info
## ----- netq-notifier Configurations -----
## Netq Notifier configuration
```

```
##
## log_level: Could be debug, info, warning or error. Default is info.
##
#netq-notifier:
#  log_level: info
## Slack Notifications
##
## netq-notifier sends notifications to Slack using Incoming Webhooks.
## The webhook for your channel can be found or created on Slack at:
## Apps -> Custom Integrations -> Incoming Webhooks.
## Each webhook has a 'Webhook URL'. Please specify that for webhook.
##
## enable: true or false
## webhook: <webhook link>
##
#slack:
#  enable: true
#  webhook: https://cumulusnetworks.slack.com/example/hook
## PagerDuty Notifications
##
## netq-notifier sends notifications to PagerDuty using the Events
API v2.
## To access the PagerDuty, we need a unique API Access Key which can
be created
## on your PagerDuty website at:
## Configuration -> API Access -> Create New API Key
## The Netq PagerDuty Integration needs to be identified by an
'Integration Key'
## that can be created/found on your PagerDuty website at:
## Configuration -> Services -> Add New Service -> New Integration ->
## Select Integration Type as 'Use our API directly: Events API v2'
##
#pagerduty:
#  enable: false
#  api_access_key:
#  api_integration_key:
## Agent State Notifications
##
## Notify when the agent goes Rotten or Fresh.
##
## enable: true or false
## include: list of node names to monitor. E.g. [cumulus-sw1, cumulus-
sw2]
## exclude: list of node names to ignore. E.g. [cumulus-sw1, cumulus-
sw2]
##          If 'include' is specified, this field is ignored.
#agents:
#  enable: true
#  exclude: []
#  include: [leaf01, leaf02, spine01, spine02]
## BGP Session Notifications
##
```

```
## Notify when BGP sessions go up or down
## enable: true or false
## include: list of node names or "nodename neighbor" to monitor
## exclude: list of node names or "nodename neighbor" to ignore.
##          E.g. [cumulus-sw1, cumulus-sw2 peerlink-3]
##          If 'include' is specified, this field is ignored.
#bgp:
#  enable: true
#  exclude: []
#  include: []
## OSPF Session Notifications
##
## Notify when OSPF sessions go up or down
## enable: true or false
## include: list of node names or "nodename neighbor" to monitor.
##            E.g. [cumulus-sw1, cumulus-sw2 peerlink-3]
## exclude: list of node names or "nodename neighbor" to ignore.
##          E.g. [cumulus-sw1, cumulus-sw2 peerlink-3]
##          If 'include' is specified, this field is ignored.
#ospf:
#  enable: true
#  exclude: []
#  include: [leaf01, leaf02, spine01, spine02]
## CLAG Node Notifications
##
## Notify when clag goes up or down.
## enable: true or false
## include: list of node names to monitor. E.g. [cumulus-sw1, cumulus-
sw2]
## exclude: list of node names to ignore. E.g. [cumulus-sw1, cumulus-
sw2]
##          If 'include' is specified, this field is ignored.
#clag:
#  enable: true
#  exclude: []
#  include: []
## Sensor Notifications
##
## Notify when sensors change state
## enable: true or false
## include: list of node names or "nodename sensor" to monitor
## exclude: list of node names or "nodename sensor" to ignore.
##          E.g. [cumulus-sw1, cumulus-sw2 temp1]
##          If 'include' is specified, this field is ignored.
#sensors:
#  enable: true
#  exclude: []
#  include: [leaf01, leaf02]
## Link Notifications
##
## Notify when links go up or down
## enable: true or false
## include: list of node names or "nodename interface" to monitor
## exclude: list of node names or "nodename interface" to ignore.
##          E.g. [cumulus-sw1, cumulus-sw2 swp1]
##          If 'include' is specified, this field is ignored.
#link:
#  enable: true
#  exclude: []
#  include: [leaf01, leaf02, spine01, spine02]
## VLAN notifications
##
## Notify when VLAN mismatch has occurred
## enable: true or false
## include: list of node names or "nodename interface" to monitor
## exclude: list of node names or "nodename interface" to ignore.
##          E.g. [cumulus-sw1, cumulus-sw2 swp1]
##          If 'include' is specified, this field is ignored.
#vlan:
#  enable: true
#  exclude: []
#  include: []
## License Notifications
##
## Notify when Cumulus Linux license is valid or not
## enable: true or false
## include: list of node names to monitor. E.g. [cumulus-sw1, cumulus-
sw2]
## exclude: list of node names to ignore. E.g. [cumulus-sw1, cumulus-
sw2]
##          If 'include' is specified, this field is ignored.
#license:
#  enable: true
#  exclude: []
#  include: []
## MTU notifications
##
## Notify when MTU mismatch has occurred
## enable: true or false
## include: list of node names or "nodename interface" to monitor
## exclude: list of node names or "nodename interface" to ignore.
##          E.g. [cumulus-sw1, cumulus-sw2 swp1]
##          If 'include' is specified, this field is ignored.
#mtu:
#  enable: true
#  exclude: []
#  include: []
## LNV notifications
##
## Notify when LNV error has occurred
## enable: true or false
## include: list of node names or "nodename interface" to monitor
## exclude: list of node names or "nodename interface" to ignore.
##          E.g. [cumulus-sw1, cumulus-sw2 swp1]
##          If 'include' is specified, this field is ignored.
#lnv:
#  enable: true
#  exclude: []
#  include: []
## VXLAN notifications
##
## Notify when VXLAN mismatch has occurred
#vxlan:
#  enable: true
#  exclude: []
#  include: []
```

Getting to Know NetQ

After you've installed NetQ, running `netq example` gives you some pointers as to how it helps you solve issues across your network.

```
cumulus@oob-mgmt-server:~$ netq example
    check                :  Perform fabric-wide checks
    find-duplicates      :  Find Duplicate IP or MAC
    find-origin          :  Find Origin of Route/MAC
    regexp               :  Using Regular Expressions
    resolve              :  Annotate input with names and interesting info
    startup              :  NetQ Quickstart
    trace                :  Control Path Trace

cumulus@switch:~$ netq example trace

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


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


Usage
=====
netq trace provides control path tracing (no real packets are sent)
from
a specified source to a specified destination. The trace covers
complete
end-to-end path tracing including bridged, routed and Vxlan overlay
paths.
ECMP is supported as well as checking for forwarding loops, MTU
consistency
across all paths, and VLAN consistency across all paths. The trace
also
covers that the path from dest to src also exists on each hop.


cumulus@torc-12:~$ netq trace 27.0.0.22 from 27.0.0.21
torc-12 -- torc-12:swp3 -- spine-1:swp5 -- torc-21:lo
        -- torc-12:swp4 -- spine-2:swp5 -- torc-21:lo
```

```
When tracing data, only the egress information is shown as this
information
is gathered by looking at the routing table. In this case, there are
two paths
(one through spine01 and one through spine02) because the environment
is
leveraging equal cost routing.


You can trace by MAC as well:
cumulus@leaf1:~$ netq trace 00:02:00:00:00:02 vlan 1009 from leaf1
leaf1 -- leaf1:sw_clag200 -- spine1:sw_clag300 -- edge2
                          -- spine1:sw_clag300 -- edge1:VlA-1
      -- leaf1:sw_clag200 -- spine2:sw_clag300 -- edge1:VlA-1
                          -- spine2:sw_clag300 -- edge2
cumulus@leaf1:~$


Legend
======
Any errors are shown in red. Bridged paths are always in WHITE,
routed paths
in GREEN, the VTEPs are shown in BLUE. A node in error is shown in
RED.
```

And `netq help` shows you information about specific commands.

```
cumulus@switch:~$ netq help show interfaces
Commands:
   netq <hostname> show docker container adjacent [interfaces <remote-
physical-interface>] [around <text-time>] [json]
   netq [<hostname>] show docker container name <container-name>
adjacent [interfaces <remote-physical-interface>] [around <text-
time>] [json]
   netq [<hostname>] show interfaces [around <text-time>] [count]
[json]
   netq <hostname> show interfaces <remote-interface> [around <text-
time>] [count] [json]
   netq [<hostname>] show interfaces type
(bond|bridge|eth|loopback|macvlan|swp|vlan|vrf|vxlan) [around <text-
time>] [count] [json]
   netq [<hostname>] show interfaces changes [between <text-time> and
<text-endtime>] [json]
   netq <hostname> show interfaces <remote-interface> changes
[between <text-time> and <text-endtime>] [json]
   netq [<hostname>] show interfaces type
(bond|bridge|eth|loopback|macvlan|swp|vlan|vrf|vxlan) changes
[between <text-time> and <text-endtime>] [json]
```

Getting Information about Network Hardware

You can get information about the hardware on the nodes in the network with `netq show inventory` command. You can get details about the ASIC, motherboard, CPU, license, memory, storage, operating system. To see a shorter summary, use the `brief` option:

```
netq@446c0319c06a:/$ netq show inventory brief
Node      Switch    OS             CPU     ASIC    Ports
--------  --------  -------------  ------  ------  -------
exit01    VX        Cumulus Linux  x86_64  N/A     N/A
exit02    VX        Cumulus Linux  x86_64  N/A     N/A
leaf01    VX        Cumulus Linux  x86_64  N/A     N/A
leaf02    VX        Cumulus Linux  x86_64  N/A     N/A
leaf03    VX        Cumulus Linux  x86_64  N/A     N/A
leaf04    VX        Cumulus Linux  x86_64  N/A     N/A
server01  N/A       Ubuntu         x86_64  N/A     N/A
server02  N/A       Ubuntu         x86_64  N/A     N/A
server03  N/A       Ubuntu         x86_64  N/A     N/A
server04  N/A       Ubuntu         x86_64  N/A     N/A
spine01   VX        Cumulus Linux  x86_64  N/A     N/A
spine02   VX        Cumulus Linux  x86_64  N/A     N/A
```

Using the NetQ Shell on the NetQ Telemetry Server

If you need to run `netq` commands from the telemetry server, use the NetQ shell. While most other Linux commands can work from this shell, Cumulus Networks recommends you only run `netq` commands here.

```
cumulus@netq-appliance:~$ netq-shell
[<Container: a017716433>]
Welcome to Cumulus (R) Linux (R)

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

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

TIP: Type `netq` to access NetQ CLI.
netq@017716433d5:/$ netq show agents
Node        Status    Sys Uptime    Agent Uptime
----------  --------  ------------  --------------
exit01      Fresh     3h ago        3h ago
exit02      Fresh     3h ago        3h ago
leaf01      Fresh     3h ago        3h ago
leaf02      Fresh     3h ago        3h ago
server01    Fresh     3h ago        3h ago
server02    Fresh     3h ago        3h ago
server03    Fresh     3h ago        3h ago
server04    Fresh     3h ago        3h ago

...
```

Using the netq resolve Command

Linux commands can be piped through NetQ with the `netq resolve` command, in order to provide more contextual information and colored highlights. For example, to show routes installed by the kernel, you would run the `ip route show proto kernel` command:

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

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

```
cumulus@leaf01:~$ ip route show proto kernel | netq resolve
3.0.2.128/26 (
server02
 :
torbond1.103
 ) dev VlanA-1.103  scope link  src 3.0.2.131 (
leaf02
 :
VlanA-1.103
 )
3.0.2.128/26 (
server02
 :
torbond1.103
 ) dev VlanA-1-103-v0  scope link  src 3.0.2.129 (
leaf02
 :
VlanA-1-103-v0
 )
3.0.2.192/26 (
leaf02
 :
VlanA-1-104-v0
 ) dev VlanA-1.104  scope link  src 3.0.2.195 (
leaf02
 :
VlanA-1.104
 )
3.0.2.192/26 (
leaf02
 :
VlanA-1-104-v0
 ) dev VlanA-1-104-v0  scope link  src 3.0.2.193 (
leaf02
 :
VlanA-1-104-v0
 )
3.0.3.0/26 (
server01
 :
torbond1.105
```

```
 ) dev VlanA-1.105  scope link  src 3.0.3.3 (
leaf02
 :
VlanA-1.105
 )
3.0.3.0/26 (
server01
 :
torbond1.105
 ) dev VlanA-1-105-v0  scope link  src 3.0.3.1 (
leaf02
 :
VlanA-1-105-v0
 )
3.0.3.64/26 (
server02
 :
torbond1.106
 ) dev VlanA-1.106  scope link  src 3.0.3.67 (
leaf02
 :
VlanA-1.106
 )
3.0.3.64/26 (
server02
 :
torbond1.106
 ) dev VlanA-1-106-v0  scope link  src 3.0.3.65 (
leaf01
 :
VlanA-1-106-v0
 )
169.254.0.8/30 (
leaf02
 :
peerlink-1.4094
 ) dev peerlink-1.4094  scope link  src 169.254.0.10 (
leaf02
 :
peerlink-1.4094
 )
192.168.0.0/24 (
server02
 :
eth0
```

```
  ) dev eth0  scope link  src 192.168.0.15 (
 leaf01
  :
 eth0
  )
```

Sample Commands for Various Components

NetQ provides network validation for the entire stack, providing algorithmic answers to many questions, both simple and intractable, that pertain to your network fabric.

Component	Problem	Solution
Host	Where is this container located? Open ports? What image is being used?	netq show docker container
Overlay	Is my overlay configured correctly? Can A reach B?	netq check vxlan netq check lnv netq trace
L3	Is OSPF working as expected? Is BGP working as expected? Can IP A reach IP B?	netq check ospf netq check/show bgp netq trace l3
L2	Is MLAG configured correctly? Is there a STP loop? Is VLAN or MTU misconfigured? How does MAC A reach B?	netq check clag netq show clag netq show stp topology netq check vlan netq check mtu netq trace L2
OS	Are all switches licensed correctly? Do all switches have NetQ agents running?	netq check license netq check agents netq show agents
Interfaces	Is my link down? Are all bond links up?	netq show interfaces netq show interfaces type bond

Component	Problem	Solution
Hardware	Have any components crashed? What switches do I have in the network?	netq check sensors netq show sensors all netq show inventory brief

Taking Preventative Steps with Your Network

NetQ provides quality assurance capabilities to detect erroneous or undesired network configurations before the changes are rolled into production. NetQ can be used to test existing or design topologies, validate configuration changes, and review the state of the network in real time, allowing it to integrate effectively with CI/CD environments. NetQ commands can also be run in an automation tool; depending on the outcome of the automation tests, the script can either continue the deployment, or roll back the changes until the issues are addressed.

In addition, NetQ Virtual (see page 78) provides users with a Cumulus VX topology to serve as a virtual representation of your production network; once the network is verified in NetQ Virtual, the topology can then be rolled into production.

Contents

This chapter covers ...

- netq check and netq show (see page 27)
  - netq show agents (see page 29)
- Using NetQ with Automation (see page 29)
- Using NetQ Virtual (see page 30)

netq check and netq show

The `netq check` and `netq show` commands validate network state before and after configuration changes. Based on results returned by NetQ, you or your automation script can either roll back the configuration change or continue deploying it:

```
cumulus@leaf01:~$ netq check
    agents   :  netq agent
    bgp      :  BGP info
    clag     :  Multi-chassis LAG (CLAG) info
    license  :  License
    lnv      :  Lightweight Network Virtualization info
    mtu      :  Link MTU
    ospf     :  OSPF info
    sensors  :  Temperature/Fan/PSU sensors
    vlan     :  VLAN
    vxlan    :  VxLAN dataplane info

Commands:
netq check (vlan|mtu) [unverified] [around <text-time>] [json]
netq check (agents|clag|ospf|vxlan|lnv|sensors|license) [around <text-
time>] [json]
netq check bgp [vrf <vrf>] [around <text-time>] [json]
```

Here are some example check commands:

```
cumulus@leaf01:~$ netq check agents around 10m
Checked nodes: 25, Rotten nodes: 0
```

NetQ check enables users to review the state of the network at specific moments in time by specifying the `around text-time` option.

```
cumulus@leaf01:~$ netq check bgp vrf DataVrf1081
Total Nodes: 25, Failed Nodes: 1, Total Sessions: 52 , Failed
Sessions: 2,
Node   Neighbor   Peer ID    Reason   Time
------ ---------- ---------- -------- ---------
exit-1 swp6.3     firewall-1 Idle     Yesterday
exit-1 swp7.3     firewall-2 Idle     Yesterday
```

```
cumulus@leaf01:~$ netq check lnv around 10m
Checked Nodes: 9, Warning Nodes: 0, Failed Nodes: 0
```

```
cumulus@leaf01:~$ netq check sensors around 14m
Total Nodes: 25, Failed Nodes: 0, Checked Sensors: 221, Failed
Sensors: 0
```

The `netq show` command displays a wide variety of content from the network:

```
cumulus@leaf01:~$ netq show
    agents      :  netq agent
    bgp         :  BGP info
    changes     :  How this infomation has changed with time
    clag        :  Multi-chassis LAG (CLAG) info
    docker      :  Docker Info
    interfaces  :  Network interface
    inventory   :  Inventory information
    ip          :  IPv4 related info
    ipv6        :  IPv6 related info
    lldp        :  LLDP based neighbor info
    lnv         :  Lightweight Network Virtualization info
    macs        :  Mac table entries
    sensors     :  Temperature/Fan/PSU sensors
    services    :  System services
```

netq show agents

To get the health 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 3 consecutive heartbeats are missed, its status changes to *Rotten*.

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

Node Name    Status   Connect Time         Last Connect
-----------  -------- -------------------  --------------
exit01       Fresh    2017-06-01 00:53:23  22s ago
exit02       Fresh    2017-06-01 00:53:24  20s ago
leaf01       Fresh    2017-06-01 00:53:22  23s ago
leaf02       Fresh    2017-06-01 00:53:23  22s ago
leaf03       Fresh    2017-06-01 00:53:22  21s ago
leaf04       Fresh    2017-06-01 00:53:22  21s ago
spine01      Fresh    2017-06-01 00:53:25  20s ago
spine02      Fresh    2017-06-01 00:53:25  19s ago
```

Using NetQ with Automation

Using NetQ for preventative care of your network pairs well with automation scripts and playbooks to prevent errors on your network before deploying the configuration to production.

NetQ works with Ansible, Chef and Puppet.

For example, you can use NetQ in your Ansible playbook to help you configure your network topology. The playbook could pull in BGP data in JSON format before it starts creating the topology:

```
- hosts: localhost leaf spine
  gather_facts: False
  tasks:
     - name: Gather BGP Adjanceny info in JSON format
       local_action: command netq show bgp json
       register: result
       #delegate_to: localhost
       run_once: true
```

Based on the outcome, the playbook can then respond appropriately. Later, it can check IP addresses to verify the connections:

```
#ipv6 address check
     - name: run ipv6check on broken_dict
       command: netq show ipv6 addresses {{item.key}} {{item.value}}
json
       with_dict: "{{broken_dict}}"
       register: command_outputs
       delegate_to: localhost
       run_once: true
```

Using NetQ Virtual

The NetQ Virtual environment provides another way for you to verify your network configuration before deploying it into production. For more information, see Configuring the NetQ Virtual Environment (see page 78).

Proactively Monitoring the Network Fabric

NetQ continually and algorithmically checks for these symptoms and sends real-time alerts via *NetQ Notifier* to notify users that a network state deviation has occurred. When alerted, you can determine precisely where the fault occurred so you can remediate quickly.

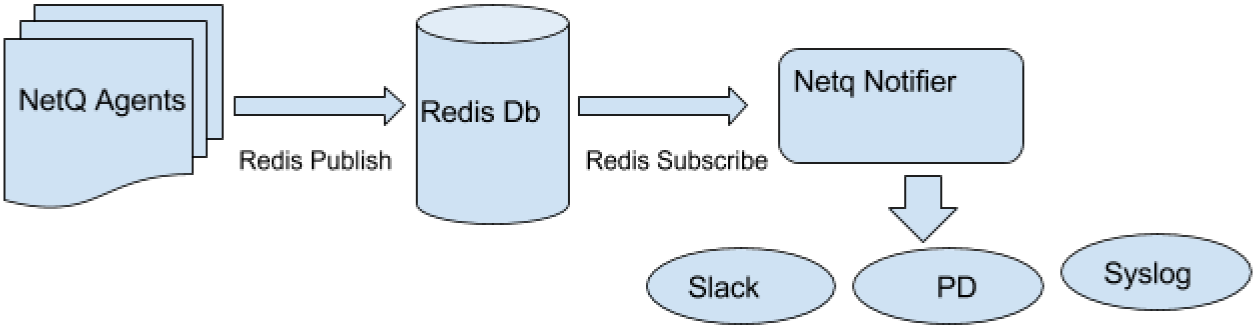
Contents

This chapter covers ...

- NetQ Notifier (see page 31)
    - Log Message Format (see page 32)
    - Supported Third-party Applications (see page 32)
    - Early Access Support (see page 34)
    - Exporting to ELK (see page 35)
    - Exporting to Splunk (see page 35)
    - Precisely Locating an Issue on the Network (see page 36)
- Extending NetQ with Custom Services Using curl (see page 37)
- Exporting NetQ Data (see page 38)

NetQ Notifier

The NetQ Notifier's role within the NetQ suite of applications is to deliver alerts to users through mediums such as Slack and syslog, informing users of network events.



Notifications can be provided for the following network events:

- BGP session failures
- Host sensor failures
- License failures
- Link up/down
- LNV failures

- MLAG node failures
- MTU mismatches
- NetQ Agent failures
- OSPF session failures
- VLAN mismatches
- VXLAN failures

When a notification arrives, what should you do next? Typically, you could run `netq check` commands; see Performing Network Diagnostics (see page 47) for more information. For a thorough example, read about troubleshooting MLAG node failures (see page 39).

Log Message Format

Messages have the following structure:

```
<level> <type>: <message>
```

For example:

```
INFO: AGENTS: All nodes are up.
```

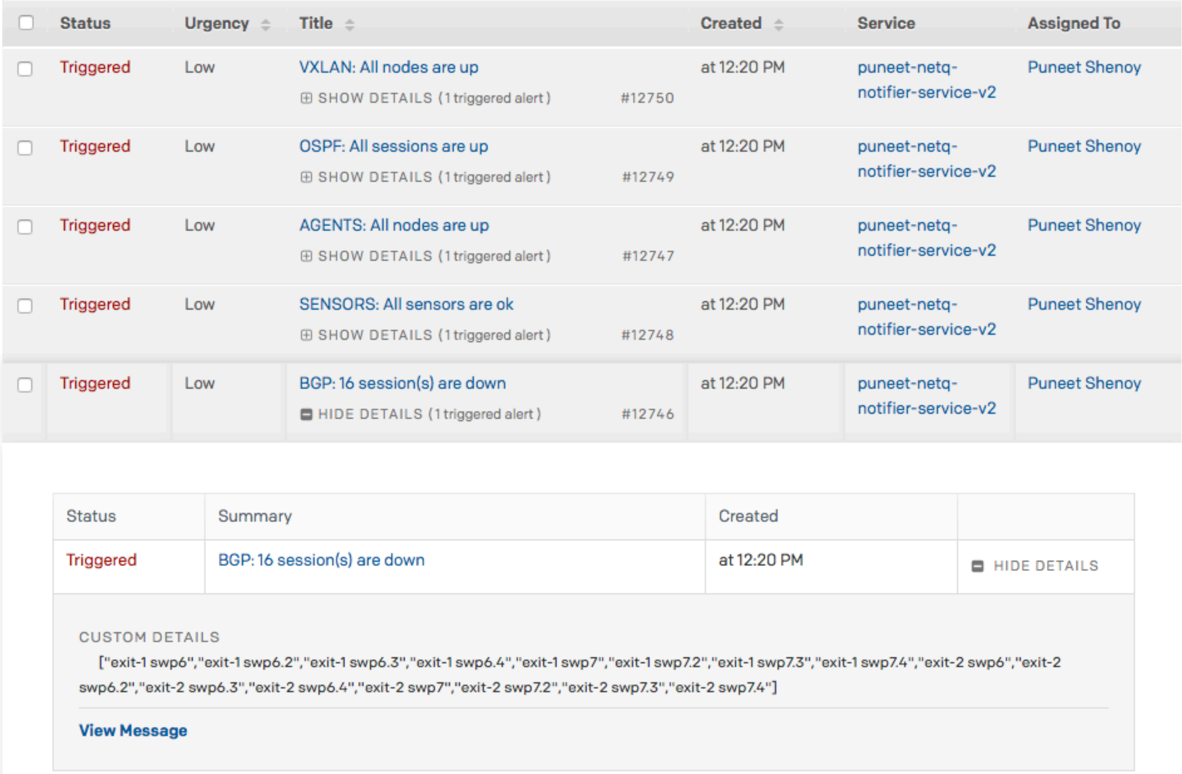
Enumerated lists are appended to the next line:

```
WARNING: VLAN: 3 mismatch(es) are found. They are: server01 torbond1,
server02 torbond1, server03 torbond1
```

Supported Third-party Applications

The following applications are supported by NetQ for notifications:

- PagerDuty: NetQ Notifier sends notifications to PagerDuty as PagerDuty events.



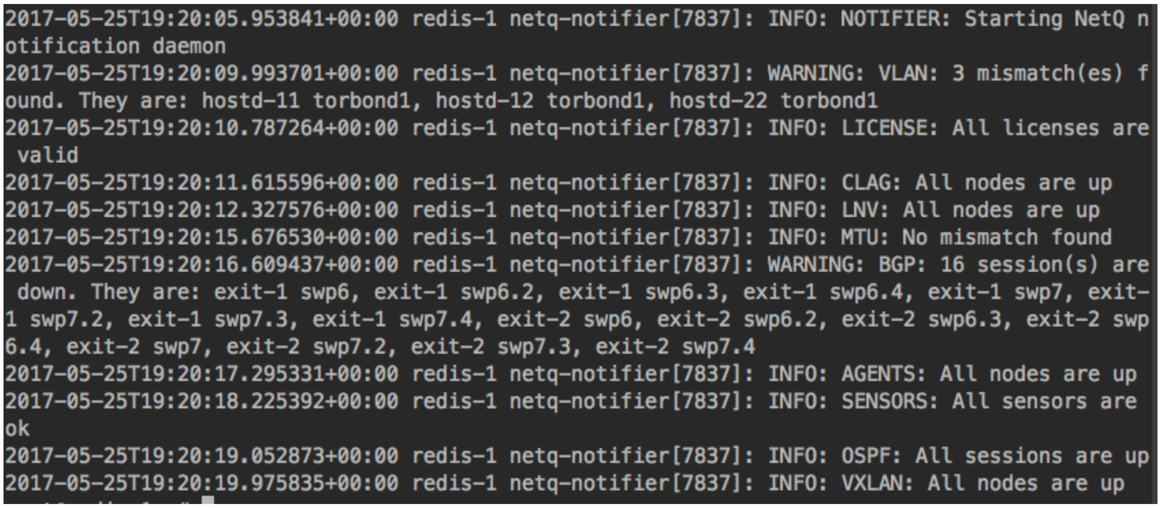
□	Status	Urgency	Title	Created	Service	Assigned To
□	Triggered	Low	VXLAN: All nodes are up ⊞ SHOW DETAILS (1 triggered alert) #12750	at 12:20 PM	puneet-netq-notifier-service-v2	Puneet Shenoy
□	Triggered	Low	OSPF: All sessions are up ⊞ SHOW DETAILS (1 triggered alert) #12749	at 12:20 PM	puneet-netq-notifier-service-v2	Puneet Shenoy
□	Triggered	Low	AGENTS: All nodes are up ⊞ SHOW DETAILS (1 triggered alert) #12747	at 12:20 PM	puneet-netq-notifier-service-v2	Puneet Shenoy
□	Triggered	Low	SENSORS: All sensors are ok ⊞ SHOW DETAILS (1 triggered alert) #12748	at 12:20 PM	puneet-netq-notifier-service-v2	Puneet Shenoy
□	Triggered	Low	BGP: 16 session(s) are down ⊟ HIDE DETAILS (1 triggered alert) #12746	at 12:20 PM	puneet-netq-notifier-service-v2	Puneet Shenoy

Status	Summary	Created	
Triggered	BGP: 16 session(s) are down	at 12:20 PM	⊟ HIDE DETAILS

CUSTOM DETAILS
["exit-1 swp6","exit-1 swp6.2","exit-1 swp6.3","exit-1 swp6.4","exit-1 swp7","exit-1 swp7.2","exit-1 swp7.3","exit-1 swp7.4","exit-2 swp6","exit-2 swp6.2","exit-2 swp6.3","exit-2 swp6.4","exit-2 swp7","exit-2 swp7.2","exit-2 swp7.3","exit-2 swp7.4"]

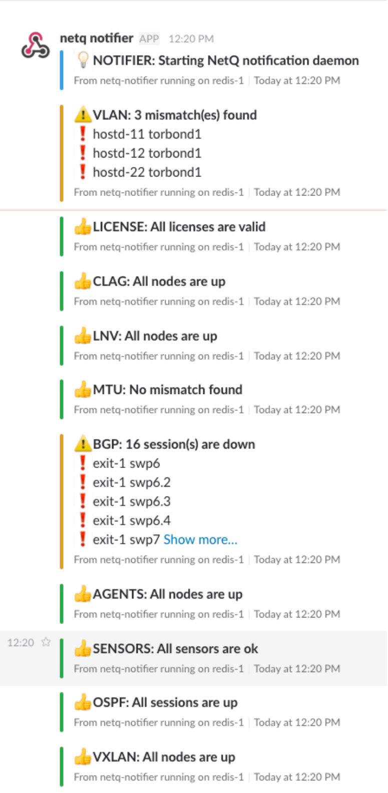
View Message

- rsyslog: Using `rsyslog`, NetQ Notifier sends alerts and events to the `/var/log/docker/netq/notifier_1/netq-notifier.log` file by default, but notifications can also be sent to ELK/Logstash or Splunk.



```
2017-05-25T19:20:05.953841+00:00 redis-1 netq-notifier[7837]: INFO: NOTIFIER: Starting NetQ n
otification daemon
2017-05-25T19:20:09.993701+00:00 redis-1 netq-notifier[7837]: WARNING: VLAN: 3 mismatch(es) f
ound. They are: hostd-11 torbond1, hostd-12 torbond1, hostd-22 torbond1
2017-05-25T19:20:10.787264+00:00 redis-1 netq-notifier[7837]: INFO: LICENSE: All licenses are
 valid
2017-05-25T19:20:11.615596+00:00 redis-1 netq-notifier[7837]: INFO: CLAG: All nodes are up
2017-05-25T19:20:12.327576+00:00 redis-1 netq-notifier[7837]: INFO: LNV: All nodes are up
2017-05-25T19:20:15.676530+00:00 redis-1 netq-notifier[7837]: INFO: MTU: No mismatch found
2017-05-25T19:20:16.609437+00:00 redis-1 netq-notifier[7837]: WARNING: BGP: 16 session(s) are
 down. They are: exit-1 swp6, exit-1 swp6.2, exit-1 swp6.3, exit-1 swp6.4, exit-1 swp7, exit-
1 swp7.2, exit-1 swp7.3, exit-1 swp7.4, exit-2 swp6, exit-2 swp6.2, exit-2 swp6.3, exit-2 swp
6.4, exit-2 swp7, exit-2 swp7.2, exit-2 swp7.3, exit-2 swp7.4
2017-05-25T19:20:17.295331+00:00 redis-1 netq-notifier[7837]: INFO: AGENTS: All nodes are up
2017-05-25T19:20:18.225392+00:00 redis-1 netq-notifier[7837]: INFO: SENSORS: All sensors are
ok
2017-05-25T19:20:19.052873+00:00 redis-1 netq-notifier[7837]: INFO: OSPF: All sessions are up
2017-05-25T19:20:19.975835+00:00 redis-1 netq-notifier[7837]: INFO: VXLAN: All nodes are up
```

- Slack: NetQ Notifier sends notifications to Slack as incoming webhooks for a Slack channel you configure. For example:



netq notifier APP 12:20 PM
💡NOTIFIER: Starting NetQ notification daemon
From netq-notifier running on redis-1 | Today at 12:20 PM

⚠VLAN: 3 mismatch(es) found
❗ hostd-11 torbond1
❗ hostd-12 torbond1
❗ hostd-22 torbond1
From netq-notifier running on redis-1 | Today at 12:20 PM

👍LICENSE: All licenses are valid
From netq-notifier running on redis-1 | Today at 12:20 PM

👍CLAG: All nodes are up
From netq-notifier running on redis-1 | Today at 12:20 PM

👍LNV: All nodes are up
From netq-notifier running on redis-1 | Today at 12:20 PM

👍MTU: No mismatch found
From netq-notifier running on redis-1 | Today at 12:20 PM

⚠BGP: 16 session(s) are down
❗ exit-1 swp6
❗ exit-1 swp6.2
❗ exit-1 swp6.3
❗ exit-1 swp6.4
❗ exit-1 swp7 Show more...
From netq-notifier running on redis-1 | Today at 12:20 PM

👍AGENTS: All nodes are up
From netq-notifier running on redis-1 | Today at 12:20 PM

12:20 ☆
👍SENSORS: All sensors are ok
From netq-notifier running on redis-1 | Today at 12:20 PM

👍OSPF: All sessions are up
From netq-notifier running on redis-1 | Today at 12:20 PM

👍VXLAN: All nodes are up
From netq-notifier running on redis-1 | Today at 12:20 PM

Early Access Support

Early access features include NetQ Notifier integration with:

- ELK
- Splunk

In addition, you can export NetQ Notifier data to the following applications:

- ELK/Logstash
- PagerDuty
- Slack
- Splunk
- syslog

NetQ integrates with ELK and Splunk using `rsyslog`, a standard mechanism to capture log files in Linux. Both ELK and Splunk provide plugins to handle `rsyslog` inputs.

To configure PagerDuty, Slack or `syslog`, you need to edit the NetQ configuration file `/etc/netq/netq.yml`.

Exporting to ELK

To export NetQ Notifier data to ELK via Logstash, on the host running the NetQ Telemetry Server and NetQ Notifier, configure the notifier to send the logs to a Logstash instance. In the following example, Logstash is on a host with the IP address 192.168.50.30, using port 51414:

```
# rsyslog - logstash configuration
sed -i '/$netq_notifier_log/a if $programname == "netq-notifier" then
@@192.168.50.30:51414' /etc/rsyslog.d\
/50-netq-notifier.conf
```

Then restart `rsyslog`:

```
root@ts_host:~# systemctl restart rsyslog
```

On the server running Logstash, create a file in `/etc/logstash/conf.d/` called `notifier_logstash.conf`, and paste in the following text, using the IP address and port you specified earlier:

```
root@ts_host:~# vi /etc/logstash/conf.d/notifier_logstash.conf

input {
    syslog {
        type => syslog
        port =>
51414
    }
}
output {
    file {
        path => "/tmp/logstash_notifier.
log"
    }
}
```

Then restart Logstash:

```
root@logstash_host:~# systemctl restart logstash
```

NetQ Notifier logs now appear in `/tmp/logstash_notifier.log` on the Logstash host.

Exporting to Splunk

To export NetQ Notifier data to Splunk, on the host running the NetQ Telemetry Server and NetQ Notifier, configure the notifier to send the logs to Splunk. In the following example, Splunk is on a host with the IP address 192.168.50.30, using port 51414:

```
# rsyslog - splunk configuration
sed -i ‘/$netq_notifier_log/a if $programname == “netq-notifier” then
@@192.168.50.30:51415’ /etc/rsyslog.d\
/50-netq-notifier.conf
```

Then restart `rsyslog`:

```
root@ts_host:~# systemctl restart rsyslog
```

To configure Splunk, do the following:

1. In Splunk in a browser, choose **Add Data** > **monitor** > **TCP** > **Port**, and set it to *51415*.
2. Click **Next**, then choose **Source Type (syslog)** > **Review** > **Done**.

NetQ Notifier messages now appear in Splunk.

Precisely Locating an Issue on the Network

NetQ helps you locate exactly where you have an issue on your network. Use `netq check` or `netq trace` to locate a fault, then run `netq show changes` to see what could have caused it.

For example, checking the state of the VLANs on your network, you can see where some nodes have mismatched VLANs with their peers:

```
cumulus@leaf01:~$ netq check vlan
Checked Nodes: 25, Checked Links: 775, Failed Nodes: 3, Failed Links:
6
Vlan and/or PVID mismatch found on following links
Node      Interface    Vlans              Peer     Peer Interface
Peer Vlans        Error
--------  -----------  -----------------  -------  ----------------
-----------------  -----------------
server01  torbond1     103-106,1000-1005  leaf02   hostbond2
101-106,1000-1005  VLAN set Mismatch
server01  torbond1     103-106,1000-1005  leaf01   hostbond2
101-106,1000-1005  VLAN set Mismatch
server02  torbond1     102-106,1000-1005  leaf02   hostbond3
101-106,1000-1005  VLAN set Mismatch
server02  torbond1     102-106,1000-1005  leaf01   hostbond3
101-106,1000-1005  VLAN set Mismatch
server03  torbond1     102-106,1000-1005  leaf04   hostbond2
101-106,1000-1005  VLAN set Mismatch
server03  torbond1     102-106,1000-1005  leaf03   hostbond2
101-106,1000-1005  VLAN set Mismatch
```

Extending NetQ with Custom Services Using curl

You can extend NetQ to monitor parameters beyond what it monitors by default. For example, you can create a service that runs a series of pings to a known host or between two known hosts to ensure that connectivity is valid. Or you can create a service that curls a URL and sends the output to `/dev/null`. This method works with the NetQ time machine (see page 50) capability regarding `netq show services`.

1. As the sudo user on a node running the NetQ agent, edit the `/etc/netq/config.d/netq-agent-commands.yml` file.
2. Create the custom service. In the example below, the new service is called *web*. You need to specify:
    - The *period* in seconds.
    - The *key* that identifies the name of the service.
    - The command will *run* always. If you do not specify *always* here, you must enable the service manually using `systemctl`.
    - The *command* to run. In this case we are using `curl` to ping a web server.

```
cumulus@leaf01:~$ sudo vi /etc/netq/config.d/netq-agent-commands.
yml

user-commands:
  - service: 'misc'
    commands:
      - period: "60"
        key: "config-interfaces"
        command: "/bin/cat /etc/network/interfaces"
      - period: "60"
        key: "config-ntp"
        command: "/bin/cat /etc/ntp.conf"
  - service: "zebra"
    commands:
      - period: "60"
        key: "config-quagga"
        command: ["/usr/bin/vtysh", "-c", "show running-config"]

  - service: "web"
    commands:
      - period: "60"
        key: "webping"
        run: "always"
        command: ['/usr/bin/curl https://cumulusnetworks.com/ -o
/dev/null']
```

3. After you save and close the file, restart the NetQ agent:

```
cumulus@leaf01: netq agent restart
```

4. You can verify the command is running by checking the `/var/run/netq-agent-running.json` file:

```
cumulus@leaf01: sudo cat /var/run/netq-agent-running.json
cumulus@leaf01:mgmt-vrf:~$ cat /var/run/netq-agent-running.json
{"commands": [{"callback": null, "service": "web", "command": "
/usr/bin/curl https://cumulusnetworks.com/ -o /dev/null",
"period": 60, "key": "webping"},  #this is the output

{"service": "smond", "always": false, "period": 30, "callback":
{}, "command": "/usr/sbin/smonctl -j", "key": "smonctl-json"},
{"service": "zebra", "always": false, "period": 60, "callback":
null, "command": ["/usr/bin/vtysh", "-c", "show running-
config"], "key": "config-quagga"}, {"service": "clagd",
"always": false, "period": 15, "callback": {}, "command": "/usr
/bin/clagctl -j", "key": "clagctl-json"}, {"service": "bgpd",
"always": false, "period": 15, "callback": {}, "command": ["/usr
/bin/vtysh", "-c", "show ip bgp vrf all neighbors json"], "key":
"bgp-neighbors"}, {"service": "misc", "always": false, "period":
30, "callback": {}, "command": "/usr/sbin/switchd -lic", "key":
"cl-license"}, {"service": "misc", "always": false, "period":
60, "callback": null, "command": "/bin/cat /etc/network
/interfaces", "key": "config-interfaces"}, {"service": "misc",
"always": false, "period": 60, "callback": null, "command": "/bin
/cat /etc/ntp.conf", "key": "config-ntp"}, {"service": "lldpd",
"always": false, "period": 30, "callback": {}, "command": "/usr
/sbin/lldpctl -f json", "key": "lldp-neighbor-json"},
{"service": "mstpd", "always": false, "period": 15, "callback":
{}, "command": "/sbin/mstpctl showall json", "key": "mstpctl-
bridge-json"}], "backend": {"server": "192.168.0.254", "vrf":
"mgmt", "port": 6379}}
cumulus@leaf01:mgmt-vrf:~$
```

5. And you can see the service is running on the host when you run netq show services:

```
cumulus@leaf01: netq show services web
```

Exporting NetQ Data

Data from the NetQ Telemetry Server can be exported in a number of ways. First, you can use the `json` option to output check and show commands to JSON format for parsing in other applications.

For example, you can check the state of BGP on your network with `netq check bgp`:

```
cumulus@leaf01:~$ netq check bgp
```

```
Total Nodes: 25, Failed Nodes: 2, Total Sessions: 228 , Failed
Sessions: 2,
Node         Neighbor     Peer ID      Reason     Time
----------   ----------   ----------   --------   -------
exit01       swp6.2       spine01      Idle       15h ago
spine01      swp3.2       exit01       Idle       15h ago
```

When you show the output in JSON format, this same command looks like this:

```
cumulus@leaf01:~$ netq check bgp json
{
    "failedNodes": [
        {
            "node": "exit-1",
            "reason": "Idle",
            "peerId": "firewall-1",
            "neighbor": "swp6.2",
            "time": "15h ago"
        },
        {
            "node": "firewall-1",
            "reason": "Idle",
            "peerId": "exit-1",
            "neighbor": "swp3.2",
            "time": "15h ago"
        }
    ],
    "summary": {
        "checkedNodeCount": 25,
        "failedSessionCount": 2,
        "failedNodeCount": 2,
        "totalSessionCount": 228
    }
}
```

MLAG Troubleshooting with NetQ

This chapter 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 of MLAG state.

Contents

This chapter covers ...

- All Nodes Are Up (see page 40)
- Dual-connected Bond Is Down (see page 41)
- VXLAN Active-active Device or Interface Is Down (see page 43)
- Remote-side clagd Stopped by systemctl Command (see page 45)

All Nodes Are Up

When the MLAG configuration is running smoothly, NetQ Notifier sends out a message that all nodes are up:

```
2017-05-22T23:13:09.683429+00:00 noc-pr netq-notifier[5501]: INFO:
CLAG: All nodes are up
```

Running `netq show clag` confirms this:

```
cumulus@noc-pr:~$ netq show clag
Matching CLAG session records are:
Node             Peer             SysMac            State Backup
#Bonds #Dual Last Changed
---------------- ---------------- ----------------- ----- ------
------ ----- --------------
mlx-2700-03      torc-11(P)       44:38:39:ff:ff:01 up    up
8      8     26s ago
noc-pr(P)        noc-se           00:01:01:10:00:01 up    up
9      9     39m ago
noc-se           noc-pr(P)        00:01:01:10:00:01 up    up
9      9     40m ago
torc-11(P)       mlx-2700-03      44:38:39:ff:ff:01 up    up
8      8     27s ago
torc-21(P)       torc-22          44:38:39:ff:ff:02 up    up
8      8     2h ago
torc-22          torc-21(P)       44:38:39:ff:ff:02 up    up
8      8     2h ago
```

You can also verify a specific node is up:

```
cumulus@noc-pr:~$ netq mlx-2700-03 show clag
Matching CLAG session records are:
Node             Peer             SysMac            State Backup
#Bonds #Dual Last Changed
---------------- ---------------- ----------------- ----- ------
------ ----- --------------
mlx-2700-03      torc-11(P)       44:38:39:ff:ff:01 up    up
8      8     45s ago
```

Similarly, checking the MLAG state with NetQ also confirms this:

```
cumulus@noc-pr:~$ netq check clag
Checked Nodes: 6, Failed Nodes: 0
```

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

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

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

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

Dual-connected Bond Is Down

When dual connectivity is lost in an MLAG configuration, you'll receive messages from NetQ Notifier similar to the following:

```
2017-05-22T23:14:40.290918+00:00 noc-pr netq-notifier[5501]: WARNING:
LINK: 1 link(s) are down. They are: mlx-2700-03 hostbond5
2017-05-22T23:14:53.081480+00:00 noc-pr netq-notifier[5501]: WARNING:
CLAG: 1 node(s) have failures. They are: mlx-2700-03
2017-05-22T23:14:58.161267+00:00 noc-pr netq-notifier[5501]: WARNING:
CLAG: 2 node(s) have failures. They are: mlx-2700-03, torc-11
```

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

```
cumulus@noc-pr:~$ netq mlx-2700-03 show service clagd

Matching services records are:
Node        Service   PID   VRF     Enabled   Active   Monitored
Status   Up Time   Last Changed
----------- --------- ----- ------- --------- -------- -----------
-------- --------- --------------
```

```
mlx-2700-03 clagd     5802  default yes       yes      yes
warning  1h ago    2m ago
```

Checking the MLAG status provides the reason for the failure:

```
cumulus@noc-pr:~$ netq check clag
Checked Nodes: 6, Warning Nodes: 2
Node             Reason
----------------
----------------------------------------------------------------------
----
mlx-2700-03      Link Down: hostbond5
torc-11          Singly Attached Bonds: hostbond5
```

You can retrieve the output in JSON format for importing the output into another tool:

```
cumulus@noc-pr:~$ netq check clag json
{
"warningNodes": [
{ "node": "mlx-2700-03", "reason": "Link Down: hostbond5" }
,
{ "node": "torc-11", "reason": "Singly Attached Bonds: hostbond5" }
],
"failedNodes": [],
"summary":
{ "checkedNodeCount": 6, "failedNodeCount": 0, "warningNodeCount": 2 }
}
```

After you fix the issue, you can show the MLAG state to see if all the nodes are up. The notifications from NetQ Notifier indicate all nodes are UP, and the `netq check` flag also indicates there are no failures.

```
cumulus@noc-pr:~$ netq show clag
Matching CLAG session records are:
Node             Peer             SysMac            State Backup
#Bonds #Dual Last Changed
---------------- ---------------- ----------------- ----- ------
------ ----- --------------
mlx-2700-03      torc-11(P)       44:38:39:ff:ff:01 up    up
8      7     52s ago
noc-pr(P)        noc-se           00:01:01:10:00:01 up    up
9      9     27m ago
noc-se           noc-pr(P)        00:01:01:10:00:01 up    up
9      9     27m ago
torc-11(P)       mlx-2700-03      44:38:39:ff:ff:01 up    up
8      7     50s ago
torc-21(P)       torc-22          44:38:39:ff:ff:02 up    up
8      8     1h ago
```

```
torc-22          torc-21(P)       44:38:39:ff:ff:02 up    up
8      8     1h ago
```

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

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

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

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

VXLAN Active-active Device or Interface Is Down

When a VXLAN active-active device or interface in an MLAG configuration is down, log messages also include VXLAN and LNV checks.

```
2017-05-22T23:16:51.517522+00:00 noc-pr netq-notifier[5501]: WARNING:
VXLAN: 2 node(s) have failures. They are: mlx-2700-03, torc-11
2017-05-22T23:16:51.525403+00:00 noc-pr netq-notifier[5501]: WARNING:
LINK: 2 link(s) are down. They are: torc-11 vx-37, mlx-2700-03 vx-37
2017-05-22T23:16:54.194681+00:00 noc-pr netq-notifier[5501]: WARNING:
LNV: 1 node(s) have failures. They are: torc-22
2017-05-22T23:16:59.448755+00:00 noc-pr netq-notifier[5501]: WARNING:
LNV: 3 node(s) have failures. They are: tor-2, torc-21, torc-22
2017-05-22T23:17:04.703044+00:00 noc-pr netq-notifier[5501]: WARNING:
CLAG: 2 node(s) have failures. They are: mlx-2700-03, torc-11
```

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

```
cumulus@noc-pr:~$ netq mlx-2700-03 show service clagd
Matching services records are:
Node        Service   PID   VRF      Enabled   Active   Monitored
Status   Up Time   Last Changed
----------- --------- ----- ------- --------- -------- -----------
-------- --------- --------------
mlx-2700-03 clagd     5802  default yes       yes      yes
error    2h ago    3m ago
```

Checking the MLAG status provides the reason for the failure:

```
cumulus@noc-pr:~$ netq check clag
Checked Nodes: 6, Warning Nodes: 2, Failed Nodes: 2
Node             Reason
----------------
----------------------------------------------------------------------
----
mlx-2700-03      Protodown Bonds: vx-37:vxlan-single
torc-11          Protodown Bonds: vx-37:vxlan-single
```

You can retrieve the output in JSON format for importing the output into another tool:

```
cumulus@noc-pr:~$ netq check clag json
{
"failedNodes": [
{ "node": "mlx-2700-03", "reason": "Protodown Bonds: vx-37:vxlan-
single" }
,
{ "node": "torc-11", "reason": "Protodown Bonds: vx-37:vxlan-single" }
],
"summary":
{ "checkedNodeCount": 6, "failedNodeCount": 2, "warningNodeCount": 2 }
}
```

After you fix the issue, you can show the MLAG state to see if all the nodes are up:

```
cumulus@noc-pr:~$ netq show clag
Matching CLAG session records are:
Node             Peer             SysMac            State Backup
#Bonds #Dual Last Changed
---------------- ---------------- ----------------- ----- ------
------ ----- --------------
mlx-2700-03      torc-11(P)       44:38:39:ff:ff:01 up    up
8      7     52s ago
noc-pr(P)        noc-se           00:01:01:10:00:01 up    up
9      9     27m ago
noc-se           noc-pr(P)        00:01:01:10:00:01 up    up
9      9     27m ago
```

```
torc-11(P)       mlx-2700-03      44:38:39:ff:ff:01 up    up
8      7     50s ago
torc-21(P)       torc-22          44:38:39:ff:ff:02 up    up
8      8     1h ago
torc-22          torc-21(P)       44:38:39:ff:ff:02 up    up
8      8     1h ago
```

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

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

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

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

Remote-side clagd Stopped by systemctl Command

In the event the `clagd` service is stopped via the `systemctl` command, NetQ Notifier sends messages similar to the following:

```
2017-05-22T23:51:19.539033+00:00 noc-pr netq-notifier[5501]: WARNING:
VXLAN: 1 node(s) have failures. They are: torc-11
2017-05-22T23:51:19.622379+00:00 noc-pr netq-notifier[5501]: WARNING:
LINK: 2 link(s) flapped and are down. They are: torc-11 hostbond5,
torc-11 hostbond4
2017-05-22T23:51:19.622922+00:00 noc-pr netq-notifier[5501]: WARNING:
LINK: 23 link(s) are down. They are: torc-11 VlanA-1-104-v0, torc-11
VlanA-1-101-v0, torc-11 VlanA-1, torc-11 vx-33, torc-11 vx-36, torc-
11 vx-37, torc-11 vx-34, torc-11 vx-35, torc-11 swp7, torc-11 VlanA-1-
```

```
102-v0, torc-11 VlanA-1-103-v0, torc-11 VlanA-1-100-v0, torc-11 VlanA-
1-106-v0, torc-11 swp8, torc-11 VlanA-1.106, torc-11 VlanA-1.105,
torc-11 VlanA-1.104, torc-11 VlanA-1.103, torc-11 VlanA-1.102, torc-
11 VlanA-1.101, torc-11 VlanA-1.100, torc-11 VlanA-1-105-v0, torc-11
vx-38
2017-05-22T23:51:27.696572+00:00 noc-pr netq-notifier[5501]: INFO:
LINK: 15 link(s) are up. They are: torc-11 VlanA-1.106, torc-11 VlanA-
1-104-v0, torc-11 VlanA-1.104, torc-11 VlanA-1.103, torc-11 VlanA-
1.101, torc-11 VlanA-1-100-v0, torc-11 VlanA-1.100, torc-11 VlanA-
1.102, torc-11 VlanA-1-101-v0, torc-11 VlanA-1-102-v0, torc-11 VlanA-
1.105, torc-11 VlanA-1-103-v0, torc-11 VlanA-1-106-v0, torc-11 VlanA-
1, torc-11 VlanA-1-105-v0
2017-05-22T23:51:30.863789+00:00 noc-pr netq-notifier[5501]: WARNING:
LNV: 1 node(s) have failures. They are: torc-11
2017-05-22T23:51:36.156708+00:00 noc-pr netq-notifier[5501]: WARNING:
CLAG: 2 node(s) have failures. They are: mlx-2700-03, torc-11
2017-05-22T23:51:36.183638+00:00 noc-pr netq-notifier[5501]: WARNING:
LNV: 2 node(s) have failures. They are: spine-2, torc-11
2017-05-22T23:51:41.444670+00:00 noc-pr netq-notifier[5501]: WARNING:
LNV: 1 node(s) have failures. They are: torc-11
```

Showing the MLAG state reveals which nodes are down:

```
cumulus@noc-pr:~$ netq show clag
Matching CLAG session records are:
Node             Peer             SysMac            State Backup
#Bonds #Dual Last Changed
---------------- ---------------- ----------------- ----- ------
------ ----- --------------
mlx-2700-03                       44:38:39:ff:ff:01 down  down
8      0     33s ago
noc-pr(P)        noc-se           00:01:01:10:00:01 up    up
9      9     1h ago
noc-se           noc-pr(P)        00:01:01:10:00:01 up    up
9      9     1h ago
torc-11                           44:38:39:ff:ff:01 down  n/a
0      0     32s ago
torc-21(P)       torc-22          44:38:39:ff:ff:02 up    up
8      8     2h ago
torc-22          torc-21(P)       44:38:39:ff:ff:02 up    up
8      8     2h ago
```

Checking the MLAG status provides the reason for the failure:

```
cumulus@noc-pr:~$ netq check clag
Checked Nodes: 6, Warning Nodes: 1, Failed Nodes: 2
Node             Reason
```

```
----------------
----------------------------------------------------------------------
----
mlx-2700-03      Peer Connectivity failed
torc-11          Peer Connectivity failed
```

You can retrieve the output in JSON format for importing the output into another tool:

```
cumulus@noc-pr:~$ netq check clag json
{
"failedNodes": [
{ "node": "mlx-2700-03", "reason": "Peer Connectivity failed" }
,
{ "node": "torc-11", "reason": "Peer Connectivity failed" }
],
"summary":
{ "checkedNodeCount": 6, "failedNodeCount": 2, "warningNodeCount": 1 }
}
```

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

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


The peer is not alive
Our Priority, ID, and Role: 8192 44:38:39:00:a5:38 primary
Peer Interface and IP: peerlink-3.4094 169.254.0.9
VxLAN Anycast IP: 36.0.0.20
Backup IP: 27.0.0.20 (inactive)
System MAC: 44:38:39:ff:ff:01


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

Performing Network Diagnostics

NetQ provides users with the ability to go back in time to replay the network state, see fabric-wide event changelogs 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 chapter covers ...

- Diagnosing an Event after It Occurs (see page 48)
- Using NetQ as a Time Machine (see page 50)
  - How Far Back in Time Can You Travel? (see page 51)
- Using trace in a VRF (see page 52)

Diagnosing an Event after It Occurs

NetQ provides a number of commands to enable you to diagnose past events.

NetQ Notifier records network events and sends them to `syslog`, or another third-party service like PagerDuty or Slack. You can use `netq show changes` to look for any changes made to the runtime configuration that may have triggered the alert, then use `netq trace` to track the connection between the nodes.

The `netq trace` command traces the route of an IP or MAC address from one endpoint to another. It works across bridged, routed and VXLAN connections, computing the path using available data instead of sending real traffic — this way, it can be run from anywhere. It performs MTU and VLAN consistency checks for every link along the path.

For example, say you get an alert about a BGP session failure. You can quickly run `netq check bgp` to determine what sessions failed:

```
cumulus@leaf01:~$ netq check bgp
Total Nodes: 25, Failed Nodes: 4, Total Sessions: 228 , Failed
Sessions: 6,
Node        Neighbor    Peer ID     Reason    Time
----------  ----------  ----------  --------  -------
exit01      swp7.2      spine02     Idle      53m ago
exit01      swp7.3      spine02     Idle      53m ago
exit02      swp6.4      spine01     Idle      53m ago
spine01     swp4.4      exit02      Idle      53m ago
spine02     swp3.2      exit01      Idle      53m ago
spine02     swp3.3      exit01      Idle      53m ago
```

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

```
cumulus@leaf01:~$ netq trace 10.1.20.252 from spine01 around 5m
spine01 -- spine01:swp1 -- leaf01:vlan20
        -- spine01:swp2 -- leaf02:vlan20
```

Then you can check what's changed on the network to help you identify the problem. Notice the nodes in a *Failed* state filter to the top of the list:

```
cumulus@leaf01:~$ netq show bgp changes
Matching BGP Session records are:
Node             Neighbor                     VRF
ASN        Peer ASN   State  PfxRx        DbState  Last Changed
---------------- ---------------------------- ----------------
---------- ---------- ------ ------------ -------- ------------
leaf04           swp52(spine02)               default
64516      65000      Estd   6            Add      5h ago
leaf03           swp52(spine02)               default
64515      65000      Estd   5            Add      5h ago
leaf01           swp52(spine02)               default
64513      65000      Estd   5            Add      5h ago
leaf02           swp52(spine02)               default
64514      65000      Estd   6            Add      5h ago
spine02          swp2(leaf02)                 default
65000      64514      Estd   2            Add      5h ago
spine02          swp3(leaf03)                 default
65000      64515      Estd   2            Add      5h ago
spine02          swp1(leaf01)                 default
65000      64513      Estd   2            Add      5h ago
spine02          swp4(leaf04)                 default
65000      64516      Estd   2            Add      5h ago
leaf04           swp51(spine01)               default
64516      65000      Estd   6            Add      5h ago
spine01          swp2(leaf02)                 default
65000      64514      Estd   2            Add      5h ago
leaf02           swp51(spine01)               default
64514      65000      Estd   6            Add      5h ago
leaf01           swp51(spine01)               default
64513      65000      Estd   5            Add      5h ago
spine01          swp1(leaf01)                 default
65000      64513      Estd   2            Add      5h ago
spine01          swp4(leaf04)                 default
65000      64516      Estd   2            Add      5h ago
leaf03           swp51(spine01)               default
64515      65000      Estd   5            Add      5h ago
spine01          swp3(leaf03)                 default
65000      64515      Estd   2            Add      5h ago
```

Using NetQ as a Time Machine

With NetQ, you can travel back to a specific point in time or a range of times to help you isolate errors and issues.

For example, if you think you had an issue with your sensors last night, you can check the sensors on all your nodes around the time you think the issue occurred:

```
cumulus@leaf01:~$ netq check sensors around 12h
Total Nodes: 25, Failed Nodes: 0, Checked Sensors: 221, Failed Sensors:
0
```

Or you can specify a range of times using the `between` option. The units of time you can specify are second (*s*), minutes (*m*), hours (*h*) and days (*d*). Always specify the most recent time first, then the more distant time. For example, to see the changes made to the network between the past minute and 5 minutes ago, you'd run:

```
cumulus@leaf01:~$ netq show changes between 1m and 5m
No changes to specified interfaces found
No changes to interface addresses found
Matching MAC table records are:
Origin MAC                  VLAN     Node Name        Egress
Port      DbState Last Changed
------ -------------------- -------- ----------------
---------------- ------- --------------
1      44:38:39:00:00:17    20       leaf02           bond-
swp1        Add     3m ago
1      44:38:39:00:00:17    20       leaf01           bond-
swp1        Add     3m ago
1      44:38:39:00:00:32    20       leaf03           bond-
swp2        Add     4m ago
1      44:38:39:00:00:32    20       leaf04           bond-
swp2        Add     4m ago
1      44:38:39:00:00:15    20       leaf01           bond-
swp2        Del     4m ago
1      44:38:39:00:00:15    20       leaf02           bond-
swp2        Del     4m ago
1      44:38:39:00:00:32    20       leaf03           bond-
swp2        Del     4m ago
1      44:38:39:00:00:32    20       leaf04           bond-
swp2        Del     4m ago
1      44:38:39:00:00:17    20       leaf02           bond-
swp1        Del     4m ago
```

```
1      44:38:39:00:00:17    20        leaf01           bond-
swp1        Del     4m ago
Matching IP route records are:
Origin Table            IP
Node             Nexthops                  DbState        Last Changed
------ ---------------- --------------------------------
---------------- ------------------------ -------------- ------------
0      default          ff02::1:ff00:5c/128
spine01          swp1                     Del            3m ago
0      default          ff02::1:ff00:12/128
leaf02           eth0                     Del            3m ago
No changes to IP neighbor table found
No changes to BGP sessions found
No changes to CLAG session found
No changes to LNV session found
```

You can travel back in time 5 minutes and run a trace from spine02 to exit01, which has the IP address 27.0.0.1:

```
cumulus@leaf01:~$ netq trace 27.0.0.1 from spine02 around 5m
Detected Routing Loop. Node exit01 (now via Local Node exit01 and
Ports swp6 <==> Remote  Node/s spine01 and Ports swp3) visited twice.
Detected Routing Loop. Node spine02 (now via mac:00:02:00:00:00:15)
visited twice.
spine02 -- spine02:swp3 -- exit01:swp6.4 -- exit01:swp3 -- exit01
                        -- spine02:swp7  -- spine02
```

How Far Back in Time Can You Travel?

The NetQ Telemetry Server stores an amount of data limited by a few factors:

- The size of the network: The larger the network, the more complex it is because of the number of routes and nodes.
- The amount of memory in the telemetry server. The more memory, the more data you can retrieve.
- The types of nodes you are monitoring with NetQ. You can monitor just network switches, or switches and hosts, or switches, hosts and containers.
- The number of changes in the network over time.

In general, you can expect to be able to query to a point back in time follows:

Using NetQ to Monitor ...	Data Point	Small Network	Medium Network	Large Network
Switches only	Telemetry server memory minimum	8G	16G	24G
	Years of data retrievable	25.5	17.4	15.6
Switches and Linux hosts		16G	32G	48G

Using NetQ to Monitor ...	Data Point	Small Network	Medium Network	Large Network
	Telemetry server memory minimum			
	Years of data retrievable	4.3	2.7	2.4
Switches, Linux hosts and containers	Telemetry server memory minimum	32G	64G	96G
	Years of data retrievable	2.9	1.5	1.2

The sizing numbers in this table rely on the following assumptions and definitions:

- The types of configuration and operational data being recorded:
  - Switches and hosts: Interfaces; MLAG; LLDP-enabled links; IPv4/v6 addresses, neighbors and routes; BGP sessions; link flaps per day; IPv4/v6 route flaps per day; BGP and MLAG session flaps.
  - Containers: Exposed ports, networks, container flaps per day.
- A small network has 20 racks with 40 leaf nodes, 10 spine nodes and 40 hosts per rack.
- A medium network has 60 racks with 120 leaf nodes, 30 spine nodes and 40 hosts per rack.
- A large network has 100 racks with 200 leaf nodes, 50 spine nodes and 40 hosts per rack.
- The hosts are dual-attached.
- The network is oversubscribed 4:1.
- Adding more memory to the telemetry server allows you to go back even further in time, in a near linear fashion. So doubling the memory should double the range.

Using trace 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
```

NetQ Service Console

The NetQ Telemetry Server provides access to the NetQ Service Console, a graphical user interface (GUI) for NetQ. The service console in turn provides terminal access to any node in the fabric.

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

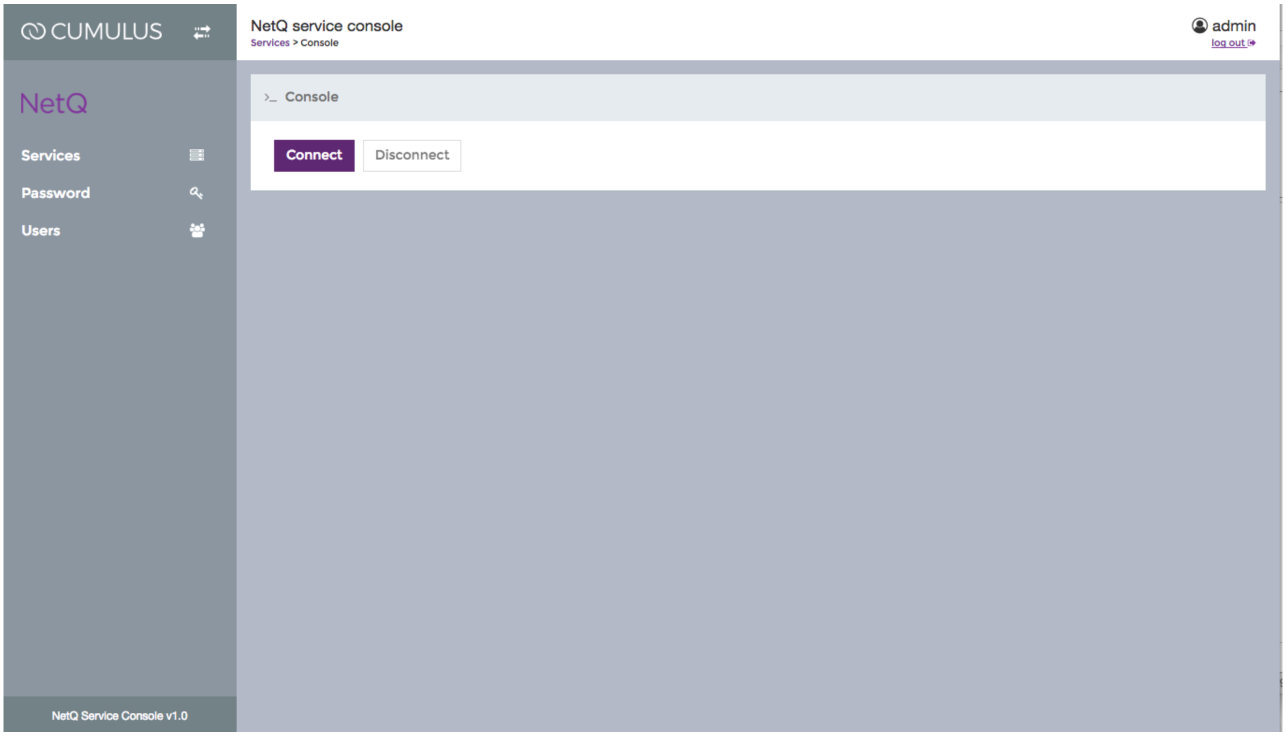
Contents

This chapter covers ...

- Connecting to the Service Console (see page 53)
- Configuring Users (see page 53)
  - Other User Account Actions (see page 54)
- Accessing the NetQ Command Line (see page 54)

Connecting to the Service Console

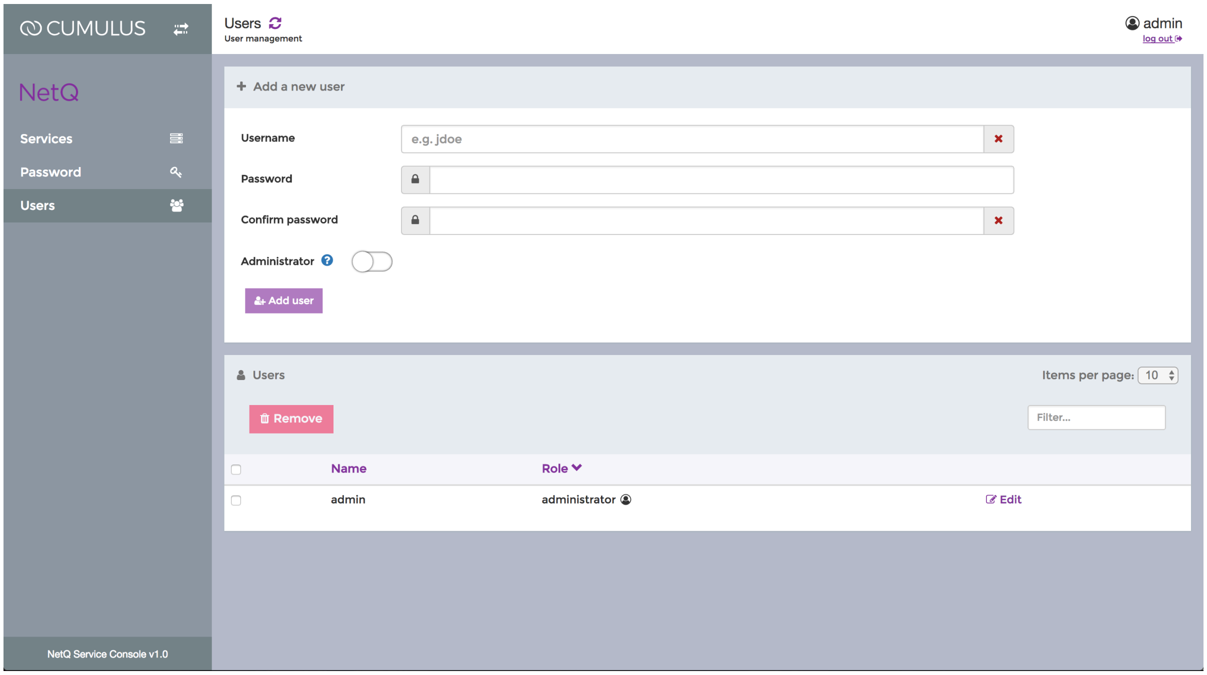
To connect to the service console, open a browser, and go to the IP address of the telemetry server (see page 7). You are prompted to log in with the username and password for the service console. The default account is the *admin* user, and its default password is *CumulusNetQ!*.



Configuring Users

By default, the service console is configured with an administrator account named *admin*, but you can add more users as needed. To add a new user:

1. In the Service Console, click **Users**.



	Name	Role	
□	admin	administrator	Edit

2. Enter the username in the **Username** field.
3. Enter that user's password in the **Password** and **Confirm Password** fields.
4. If this user account is to be an administrator to the service console, enable the **Administrator** toggle.
5. Click **Add user** to create the account.

Other User Account Actions

You can edit a user's role. On the **Users** tab, select the account under **Users**, then click **Edit**.

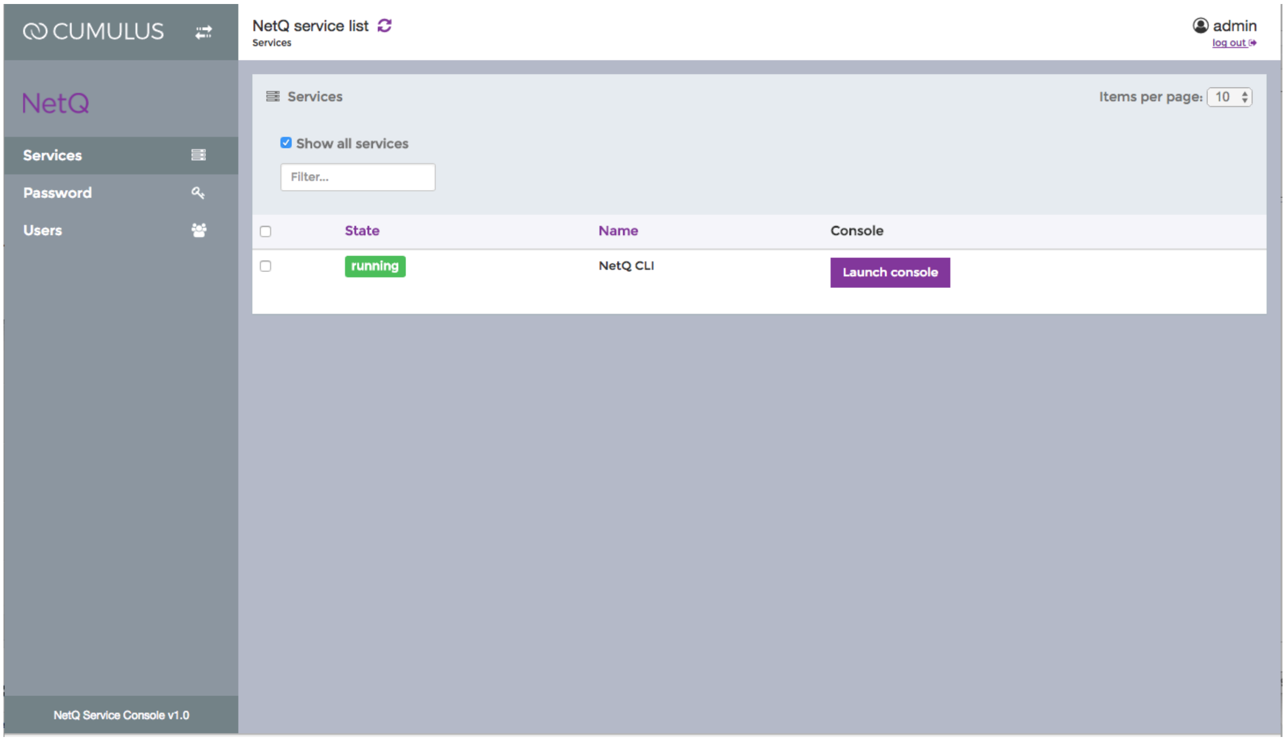
You can delete a user account. On the **Users** tab, select the account under **Users**, then click **Remove**. You cannot delete the last remaining administrator account.

To change an account password, click the **Password** tab.

Accessing the NetQ Command Line

The service console provides access to a standard Bash shell, so you can run NetQ commands — or any Linux command — directly on a given node.

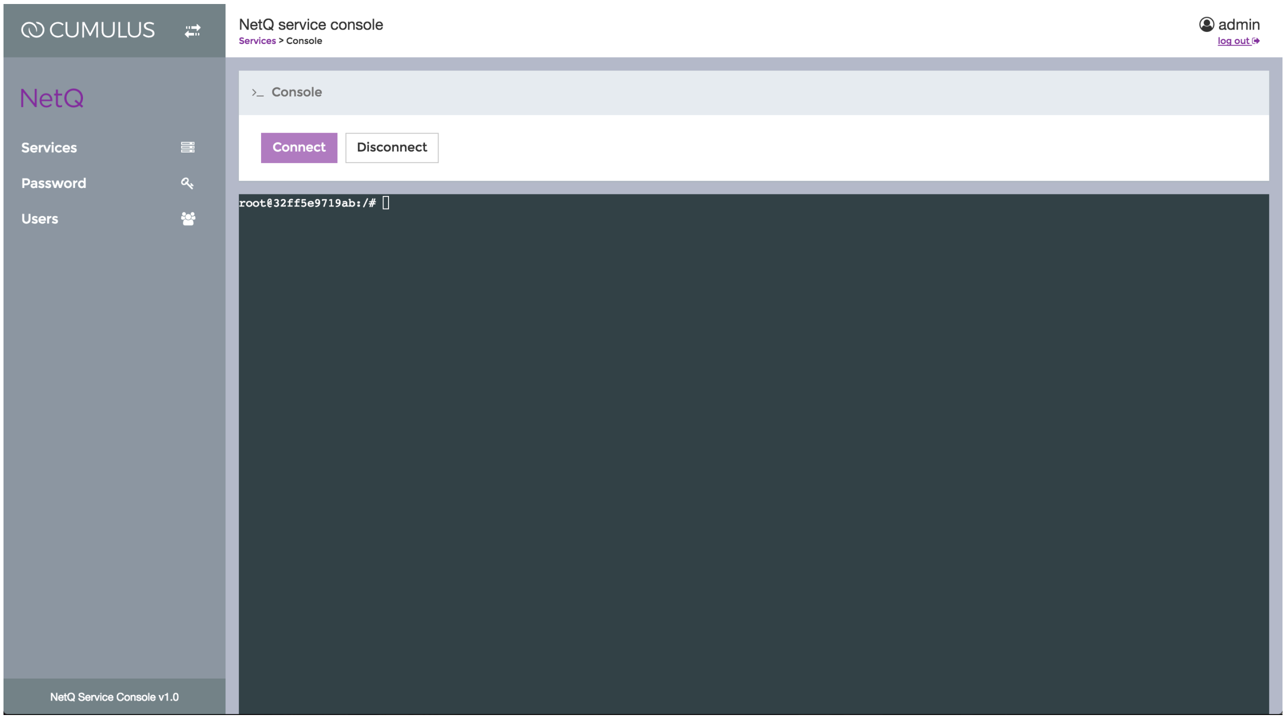
> The console is connected to the NetQ CLI container within the telemetry server; it is not connected to the shell of the telemetry server itself. As such, the `netq-shell` command does not work in the console; it is intended to run regular NetQ commands.



☑ Show all services

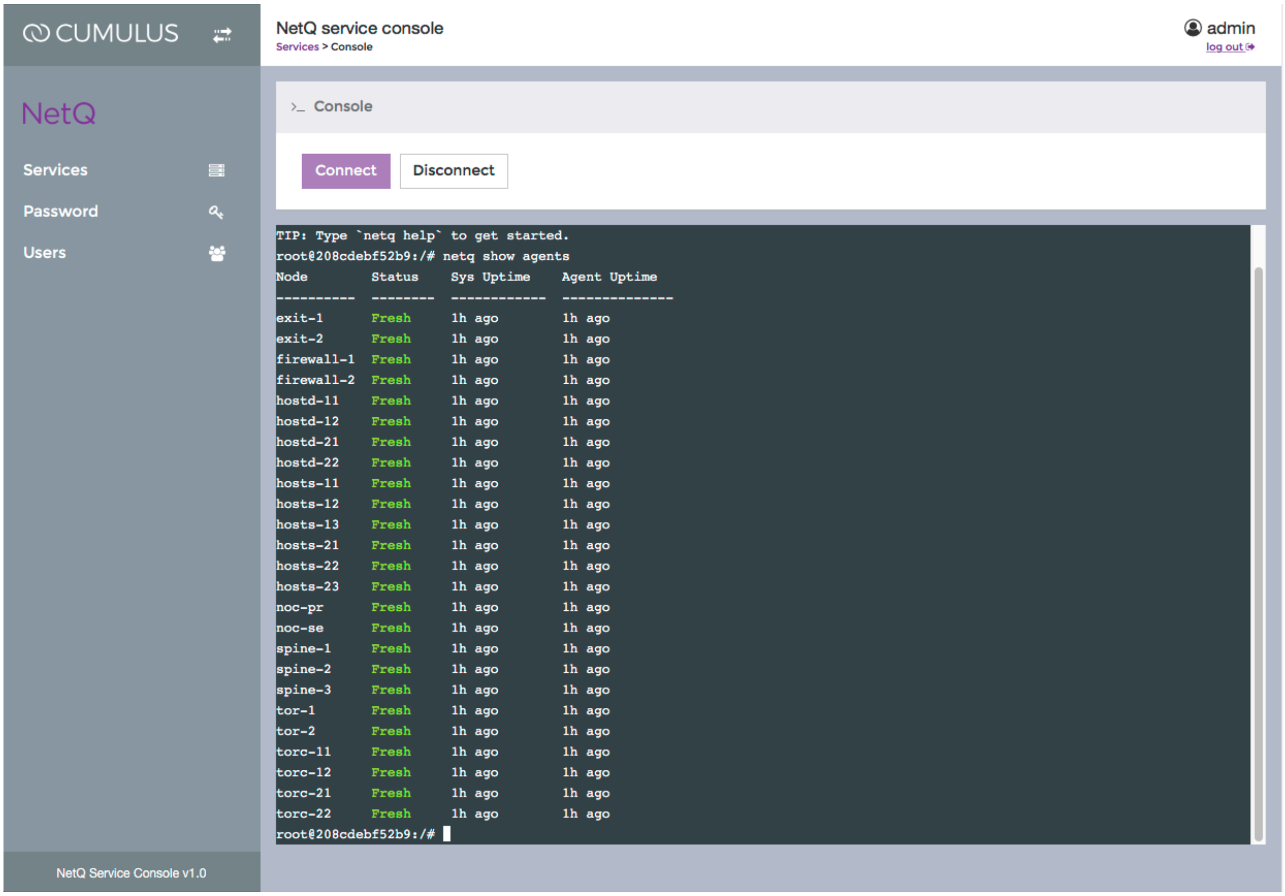
□	State	Name	Console
□	running	NetQ CLI	Launch console

In the Services window of the console, click **Launch console**, then click **Connect**.



```
root@32ff5e9719ab:/# 
```

You can run any NetQ commands within the console, such as `netq show agents`:



```
TIP: Type `netq help` to get started.
root@208cdebf52b9:/# netq show agents
Node         Status    Sys Uptime    Agent Uptime
----------   --------  ------------  --------------
exit-1       Fresh     1h ago        1h ago
exit-2       Fresh     1h ago        1h ago
firewall-1   Fresh     1h ago        1h ago
firewall-2   Fresh     1h ago        1h ago
hostd-11     Fresh     1h ago        1h ago
hostd-12     Fresh     1h ago        1h ago
hostd-21     Fresh     1h ago        1h ago
hostd-22     Fresh     1h ago        1h ago
hosts-11     Fresh     1h ago        1h ago
hosts-12     Fresh     1h ago        1h ago
hosts-13     Fresh     1h ago        1h ago
hosts-21     Fresh     1h ago        1h ago
hosts-22     Fresh     1h ago        1h ago
hosts-23     Fresh     1h ago        1h ago
noc-pr       Fresh     1h ago        1h ago
noc-se       Fresh     1h ago        1h ago
spine-1      Fresh     1h ago        1h ago
spine-2      Fresh     1h ago        1h ago
spine-3      Fresh     1h ago        1h ago
tor-1        Fresh     1h ago        1h ago
tor-2        Fresh     1h ago        1h ago
torc-11      Fresh     1h ago        1h ago
torc-12      Fresh     1h ago        1h ago
torc-21      Fresh     1h ago        1h ago
torc-22      Fresh     1h ago        1h ago
root@208cdebf52b9:/#
```

When you're finished with the session, click **Disconnect** to close the console.

Monitoring Linux Hosts with NetQ

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 (see page 7) 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 — see Monitoring Container Environments with NetQ (see page 58)

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@server01:~$ netq server01 show lldp
LLDP peer info for server01:*
Node      Interface    LLDP Peer        Peer Int    Last Changed
--------  -----------  ---------------  ----------  --------------
server01  eth0         oob-mgmt-switch  swp2        10m ago
server01  eth1         leaf01           swp1        10m ago
server01  eth2         leaf02           swp1        10m ago
```

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

```
cumulus@server01:~$ netq leaf01 show lldp
LLDP peer info for leaf01:*
Node    Interface    LLDP Peer        Peer Int                 Last
Changed
------  -----------  ---------------  ---------------------
--------------
leaf01  eth0         oob-mgmt-switch  swp6                     18m ago
leaf01  swp1         server01         mac:44:38:39:00:00:03  18m ago
leaf01  swp2         server02         mac:44:38:39:00:00:15  18m ago
leaf01  swp49        leaf02           swp49                    18m ago
leaf01  swp50        leaf02           swp50                    18m ago
leaf01  swp51        spine01          swp1                     18m ago
```

```
leaf01  swp52        spine02          swp1                     18m ago
```

To get the routing table for a server:

```
cumulus@server01:~$ netq server01 show ip route
Matching IP route records are:
Origin Table            IP               Node
Nexthops                   Last Changed
------ ---------------- ---------------- ----------------
-------------------------- ----------------
0      default          0.0.0.0/0        server01
192.168.0.254: eth0        10m ago
1      default          10.1.20.0/24     server01
bond0                      10m ago
1      default          10.1.20.1/32     server01
bond0                      10m ago
1      default          192.168.0.0/24   server01
eth0                       10m ago
1      default          192.168.0.31/32  server01
eth0                       10m ago
```

Monitoring Container Environments with NetQ

The NetQ Agent monitors Docker containers the same way it monitors physical servers (see page 56). There is no special implementation. The NetQ Agent pulls Docker data from the container as it would pull data from a Cumulus Linux switch or Linux host.

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.

Contents

This chapter covers ...

- NetQ Container Support (see page 59)
    - Telemetry Server Memory Requirement (see page 60)
- Configuring the Container Host (see page 60)
- Starting and Stopping Containers (see page 61)
- Showing Container Summary Information (see page 61)
- Identifying Containers on the Network (see page 62)
- Showing Container Adjacency (see page 67)
- Showing Container-specific Information (see page 71)
- Showing Containers with a Specific Image (see page 71)
- Showing Container Connectivity (see page 74)
- Checking Network Traffic over a Given Protocol (see page 77)

NetQ Container Support

The NetQ Agent supports Docker version 1.13 (Jan 2017), 17.04.0-ce (April 2017).

The NetQ Agent parses the following Docker events:

- Image: pull and delete
- Container: run, stop, start, restart, attach and detach
- Network: create, connect, disconnect and destroy

Currently, the NetQ Agent does not support:

- Monitoring Docker volume mount and unmount events

- Plugin install and deletes
- Third party network configuration through plugins like Calico
- Docker Swarm service

Telemetry Server Memory Requirement

Due to the higher memory requirements to run containers, Cumulus Networks recommends you run the NetQ Telemetry Server on a host with at least 32G RAM. For more information, read the Performing Network Diagnostics (see page 51) chapter.

Configuring the Container Host

In order for NetQ to be able to monitor the containers on a host, you need to do three things:

- Configure the host to point to the telemetry server by its IP address
- Enable Docker in the NetQ configuration file
- Restart the agent

See the section on configuring the NetQ agent on a node (see page 10) for details. In the following example `/etc/netq/netq.yml` file on the server, the last three lines enable Docker:

```
cumulus@server01:~$ sudo vi /etc/netq/netq.yml
# See /usr/share/doc/netq/examples for full configuration file
backend:
  port: 6379
  server: 192.168.0.10
  vrf: default
user-commands:
- commands:
  - command: /bin/cat /etc/network/interfaces
    key: config-interfaces
    period: '60'
  - command: /bin/cat /etc/ntp.conf
    key: config-ntp
    period: '60'
  service: misc
- commands:
  - command:
    - /usr/bin/vtysh
    - -c
    - show running-config
    key: config-quagga
    period: '60'
  service: zebra
view-commands:
- commands:
  - command: /bin/cat /etc/network/interfaces
    key: config-interfaces
    period: '60'
  - command: /bin/cat /etc/ntp.conf
    key: config-ntp
    period: '60'
  service: misc
- commands:
  - command:
    - /usr/bin/vtysh
    - -c
    - show running-config
    key: config-quagga
    period: '60'
  service: zebra

docker:
  enable: true
  poll_period: 15
```

Starting and Stopping Containers

If you need to start or stop a single container on a host, use the `docker-compose` command. In the example below, the container is called *netq_cont_a*:

```
cumulus@server01:~$ sudo docker-compose -f /appliance/cfg/docker/netq-
base-compose.yml -p netq_cont_a stop netq-notifier
cumulus@server01:~$ sudo docker-compose -f /appliance/cfg/docker/netq-
base-compose.yml -p netq_cont_a start netq-notifier
```

Showing Container Summary Information

To see a high level view of the network, including the number of containers installed and running on the network, run `netq show docker summary`:

```
cumulus@server01:~$ netq show docker summary
Node        Version       Installed    Running    Images  Swarm
Cluster    Networks
----------  ----------  -----------  ---------  --------
---------------  ----------
exit01      17.03.1-ce           26         26
1                   3
exit02      17.03.1-ce            1          0
3                   3
server01    17.03.1-ce            0          0
0                   3
server02    17.03.1-ce            0          0
0                   3
```

```
server03    17.03.1-ce            0          0
0                   3
server04    17.03.1-ce            0          0
0                   3
server01    17.03.1-ce           13         13
1                   3
server02    17.03.1-ce            0          0
0                   3
```

Identifying Containers on the Network

To view the different container networks and the containers in them, run `netq show docker network`:

```
cumulus@server01:~$ netq show docker network
Network Name     Node       subnet          gateway         ipv6
ip masq. encrypted
---------------- ---------- --------------- --------------- ---------
-------- ---------
bridge           exit01     172.17.0.0/16                   Disabled
True     False
bridge           exit02     172.17.0.0/16                   Disabled
True     False
bridge           server01   172.17.0.0/16                   Disabled
True     False
bridge           server02   172.17.0.0/16                   Disabled
True     False
bridge           server03   172.17.0.0/16                   Disabled
True     False
bridge           server04   172.17.0.0/16                   Disabled
True     False
bridge           server01   172.17.0.0/16                   Disabled
True     False
bridge           server02   172.17.0.0/16                   Disabled
True     False
bridge           server03   172.17.0.0/16                   Disabled
True     False
bridge           server04   172.17.0.0/16                   Disabled
True     False
host             exit01                                     Disabled
False    False
host             exit02                                     Disabled
False    False
host             server01                                   Disabled
False    False
host             server02                                   Disabled
False    False
host             server03                                   Disabled
False    False
```

```
host             server04                                   Disabled
False    False
host             server01                                   Disabled
False    False
host             server02                                   Disabled
False    False
host             server03                                   Disabled
False    False
host             server04                                   Disabled
False    False
none             exit01                                     Disabled
False    False
none             exit02                                     Disabled
False    False
none             server01                                   Disabled
False    False
none             server02                                   Disabled
False    False
none             server03                                   Disabled
False    False
none             server04                                   Disabled
False    False
none             server01                                   Disabled
False    False
none             server02                                   Disabled
False    False
none             server03                                   Disabled
False    False
none             server04                                   Disabled
False    False
```

To view all the hosts using a specific container network driver, use `netq show docker network driver NAME`. Use the `brief` keyword for a shorter summary. Docker supports many network drivers.

```
cumulus@server01:~$ netq show docker network driver bridge brief
Network Name     Node       Driver    subnet          gateway
ipv6      ip masq. encrypted Containers
---------------- ---------- --------- --------------- ---------------
--------- -------- --------- -------------------------
bridge           exit01     bridge    172.17.0.0/16
Disabled  True     False     Name:netcat-8085 IPv4:172

.17.0.7/16,

Name:netcat-8082 IPv4:172

.17.0.4/16,

Name:netcat-8083 IPv4:172
```

```
.17.0.5/16,

Name:netcat-8089 IPv4:172

.17.0.11/16,

Name:netcat-8081 IPv4:172

.17.0.3/16,

Name:netcat-8084 IPv4:172

.17.0.6/16,

Name:netcat-8090 IPv4:172

.17.0.12/16,

Name:netcat-8080 IPv4:172

.17.0.2/16,

Name:netcat-8091 IPv4:172

.17.0.13/16,

Name:netcat-8092 IPv4:172

.17.0.14/16,

Name:netcat-8088 IPv4:172

.17.0.10/16,

Name:netcat-8087 IPv4:172

.17.0.9/16,

Name:netcat-8086 IPv4:172

.17.0.8/16
bridge           exit02     bridge    172.17.0.0/16
Disabled  True     False
bridge           server01   bridge    172.17.0.0/16
Disabled  True     False
bridge           server02   bridge    172.17.0.0/16
Disabled  True     False
bridge           server03   bridge    172.17.0.0/16
Disabled  True     False
bridge           server04   bridge    172.17.0.0/16
Disabled  True     False
```

```
bridge           server01   bridge     172.17.0.0/16
Disabled  True     False     Name:netcat-8082 IPv4:172

.17.0.4/16,

Name:netcat-8085 IPv4:172

.17.0.7/16,

Name:netcat-8083 IPv4:172

.17.0.5/16,

Name:netcat-8086 IPv4:172

.17.0.8/16,

Name:netcat-8089 IPv4:172

.17.0.11/16,

Name:netcat-8084 IPv4:172

.17.0.6/16,

Name:netcat-8092 IPv4:172

.17.0.14/16,

Name:netcat-8087 IPv4:172

.17.0.9/16,

Name:netcat-8080 IPv4:172

.17.0.2/16,

Name:netcat-8081 IPv4:172

.17.0.3/16,

Name:netcat-8090 IPv4:172

.17.0.12/16,

Name:netcat-8091 IPv4:172

.17.0.13/16,

Name:netcat-8088 IPv4:172

.17.0.10/16
```

```
bridge           server02   bridge    172.17.0.0/16
Disabled  True     False
bridge           server03   bridge    172.17.0.0/16
Disabled  True     False
bridge           server04   bridge    172.17.0.0/16
Disabled  True     False
```

To see all the containers on a given container network, run the following command, where the container network is named *host*:

```
cumulus@server01:~$ netq show docker container network host
Name                 Node       IP                IP Masq.
Network        Service Name    Up time
-------------------- ---------- ----------------- --------
-------------- --------------- ---------------
netcat-9080          exit01     45.0.0.17/26,     False
host                           0:29:42
                                27.0.0.3/32,
                                192.168.0.15/24
netcat-9081          exit01     45.0.0.17/26,     False
host                           0:29:41
                                27.0.0.3/32,
                                192.168.0.15/24
netcat-9082          exit01     45.0.0.17/26,     False
host                           0:29:42
                                27.0.0.3/32,
                                192.168.0.15/24
netcat-9083          exit01     45.0.0.17/26,     False
host                           0:29:39
                                27.0.0.3/32,
                                192.168.0.15/24
netcat-9084          exit01     45.0.0.17/26,     False
host                           0:29:40
                                27.0.0.3/32,
                                192.168.0.15/24
netcat-9085          exit01     45.0.0.17/26,     False
host                           0:29:40
                                27.0.0.3/32,
                                192.168.0.15/24
netcat-9086          exit01     45.0.0.17/26,     False
host                           0:29:39
                                27.0.0.3/32,
                                192.168.0.15/24
netcat-9087          exit01     45.0.0.17/26,     False
host                           0:29:38
                                27.0.0.3/32,
                                192.168.0.15/24
netcat-9088          exit01     45.0.0.17/26,     False
host                           0:29:37
                                27.0.0.3/32,
                                192.168.0.15/24
netcat-9089          exit01     45.0.0.17/26,     False
host                           0:29:38
                                27.0.0.3/32,
                                192.168.0.15/24
netcat-9090          exit01     45.0.0.17/26,     False
host                           0:29:36
                                27.0.0.3/32,
                                192.168.0.15/24
netcat-9091          exit01     45.0.0.17/26,     False
host                           0:29:37
                                27.0.0.3/32,
                                192.168.0.15/24
netcat-9092          exit01     45.0.0.17/26,     False
host                           0:29:38
                                27.0.0.3/32,
                                192.168.0.15/24
```

Showing Container Adjacency

NetQ can list all the containers running on hosts adjacent to a top of rack switch. This helps in analyzing what impact the ToR switch can have on an application

To identify all the containers that may have been launched on hosts that are adjacent to a given node, run `netq NODE show docker container adjacent`:

```
cumulus@leaf01:~$ netq leaf01 show docker container adjacent
Interface            Peer Node  Peer Interface         Container
Name       IP                   Network     Service Name
-------------------- ---------- ---------------------
-------------------- -------------------- ---------- ---------------
swp6:VlanA-1         server01   mac:00:02:00:00:00:27 netcat-
9090                               host
swp6:VlanA-1         server01   mac:00:02:00:00:00:27 netcat-
9082                               host
swp6:VlanA-1         server01   mac:00:02:00:00:00:27 netcat-
9091                               host
swp6:VlanA-1         server01   mac:00:02:00:00:00:27 netcat-
9086                               host
swp6:VlanA-1         server01   mac:00:02:00:00:00:27 netcat-
9081                               host
swp6:VlanA-1         server01   mac:00:02:00:00:00:27 netcat-
9083                               host
swp6:VlanA-1         server01   mac:00:02:00:00:00:27 netcat-
9087                               host
swp6:VlanA-1         server01   mac:00:02:00:00:00:27 netcat-
9088                               host
swp6:VlanA-1         server01   mac:00:02:00:00:00:27 netcat-
9085                               host
```

```
swp6:VlanA-1         server01   mac:00:02:00:00:00:27 netcat-
9080                               host
swp6:VlanA-1         server01   mac:00:02:00:00:00:27 netcat-
9084                               host
swp6:VlanA-1         server01   mac:00:02:00:00:00:27 netcat-
9089                               host
swp6:VlanA-1         server01   mac:00:02:00:00:00:27 netcat-
9092                               host
swp7:VlanA-1         server02   mac:00:02:00:00:00:2a netcat-
8089          172.17.0.11          bridge
swp7:VlanA-1         server02   mac:00:02:00:00:00:2a netcat-
8084          172.17.0.6           bridge
swp7:VlanA-1         server02   mac:00:02:00:00:00:2a netcat-
8092          172.17.0.14          bridge
swp7:VlanA-1         server02   mac:00:02:00:00:00:2a netcat-
8083          172.17.0.5           bridge
swp7:VlanA-1         server02   mac:00:02:00:00:00:2a netcat-
8085          172.17.0.7           bridge
swp7:VlanA-1         server02   mac:00:02:00:00:00:2a netcat-
8081          172.17.0.3           bridge
swp7:VlanA-1         server02   mac:00:02:00:00:00:2a netcat-
8080          172.17.0.2           bridge
swp7:VlanA-1         server02   mac:00:02:00:00:00:2a netcat-
8086          172.17.0.8           bridge
swp7:VlanA-1         server02   mac:00:02:00:00:00:2a netcat-
8088          172.17.0.10          bridge
swp7:VlanA-1         server02   mac:00:02:00:00:00:2a netcat-
8082          172.17.0.4           bridge
swp7:VlanA-1         server02   mac:00:02:00:00:00:2a netcat-
8091          172.17.0.13          bridge
swp7:VlanA-1         server02   mac:00:02:00:00:00:2a netcat-
8090          172.17.0.12          bridge
swp7:VlanA-1         server02   mac:00:02:00:00:00:2a netcat-
8087          172.17.0.9           bridge
swp8:VlanA-1         server03   mac:00:02:00:00:00:2d netcat-
8091          172.17.0.13          bridge
swp8:VlanA-1         server03   mac:00:02:00:00:00:2d netcat-
8083          172.17.0.5           bridge
swp8:VlanA-1         server03   mac:00:02:00:00:00:2d netcat-
8087          172.17.0.9           bridge
swp8:VlanA-1         server03   mac:00:02:00:00:00:2d netcat-
8082          172.17.0.4           bridge
swp8:VlanA-1         server03   mac:00:02:00:00:00:2d netcat-
8080          172.17.0.2           bridge
swp8:VlanA-1         server03   mac:00:02:00:00:00:2d netcat-
8092          172.17.0.14          bridge
swp8:VlanA-1         server03   mac:00:02:00:00:00:2d netcat-
8086          172.17.0.8           bridge
swp8:VlanA-1         server03   mac:00:02:00:00:00:2d netcat-
8084          172.17.0.6           bridge
swp8:VlanA-1         server03   mac:00:02:00:00:00:2d netcat-
8088          172.17.0.10          bridge
```

```
swp8:VlanA-1         server03   mac:00:02:00:00:00:2d netcat-
8090          172.17.0.12          bridge
swp8:VlanA-1         server03   mac:00:02:00:00:00:2d netcat-
8085          172.17.0.7           bridge
swp8:VlanA-1         server03   mac:00:02:00:00:00:2d netcat-
8089          172.17.0.11          bridge
swp8:VlanA-1         server03   mac:00:02:00:00:00:2d netcat-
8081          172.17.0.3           bridge
```

You can filter this output for a given interface:

```
cumulus@leaf01:~$ netq leaf01 show docker container adjacent
interfaces swp6
Interface            Peer Node  Peer Interface        Container
Name       IP                   Network    Service Name
-------------------- ---------- ---------------------
-------------------- -------------------- ---------- ---------------
swp6:VlanA-1         server01   mac:00:02:00:00:00:27 netcat-
9090                               host
swp6:VlanA-1         server01   mac:00:02:00:00:00:27 netcat-
9082                               host
swp6:VlanA-1         server01   mac:00:02:00:00:00:27 netcat-
9091                               host
swp6:VlanA-1         server01   mac:00:02:00:00:00:27 netcat-
9086                               host
swp6:VlanA-1         server01   mac:00:02:00:00:00:27 netcat-
9081                               host
swp6:VlanA-1         server01   mac:00:02:00:00:00:27 netcat-
9083                               host
swp6:VlanA-1         server01   mac:00:02:00:00:00:27 netcat-
9087                               host
swp6:VlanA-1         server01   mac:00:02:00:00:00:27 netcat-
9088                               host
swp6:VlanA-1         server01   mac:00:02:00:00:00:27 netcat-
9085                               host
swp6:VlanA-1         server01   mac:00:02:00:00:00:27 netcat-
9080                               host
swp6:VlanA-1         server01   mac:00:02:00:00:00:27 netcat-
9084                               host
swp6:VlanA-1         server01   mac:00:02:00:00:00:27 netcat-
9089
host                                7
swp6:VlanA-1         server01   mac:00:02:00:00:00:27 netcat-
9092                               host
```

And you can go back in time to check adjacency around a given moment:

```
cumulus@leaf01:~$ netq leaf01 show docker container adjacent around 1h
```

```
Interface            Peer Node  Peer Interface        Container
Name       IP                   Network    Service Name
-------------------- ---------- ---------------------
-------------------- -------------------- ---------- ---------------
swp6:VlanA-1         server01   mac:00:02:00:00:00:27 netcat-
9090                               host
swp6:VlanA-1         server01   mac:00:02:00:00:00:27 netcat-
9082                               host
swp6:VlanA-1         server01   mac:00:02:00:00:00:27 netcat-
9091                               host
swp6:VlanA-1         server01   mac:00:02:00:00:00:27 netcat-
9086                               host
swp6:VlanA-1         server01   mac:00:02:00:00:00:27 netcat-
9081                               host
swp6:VlanA-1         server01   mac:00:02:00:00:00:27 netcat-
9083                               host
swp6:VlanA-1         server01   mac:00:02:00:00:00:27 netcat-
9087                               host
swp6:VlanA-1         server01   mac:00:02:00:00:00:27 netcat-
9088                               host
swp6:VlanA-1         server01   mac:00:02:00:00:00:27 netcat-
9085                               host
swp6:VlanA-1         server01   mac:00:02:00:00:00:27 netcat-
9080                               host
swp6:VlanA-1         server01   mac:00:02:00:00:00:27 netcat-
9084                               host
swp6:VlanA-1         server01   mac:00:02:00:00:00:27 netcat-
9089                               host
swp6:VlanA-1         server01   mac:00:02:00:00:00:27 netcat-
9092                               host
swp7:VlanA-1         server02   mac:00:02:00:00:00:2a netcat-
8089          172.17.0.11          bridge
swp7:VlanA-1         server02   mac:00:02:00:00:00:2a netcat-
8084          172.17.0.6           bridge
swp7:VlanA-1         server02   mac:00:02:00:00:00:2a netcat-
8092          172.17.0.14          bridge
swp7:VlanA-1         server02   mac:00:02:00:00:00:2a netcat-
8083          172.17.0.5           bridge
swp7:VlanA-1         server02   mac:00:02:00:00:00:2a netcat-
8085          172.17.0.7           bridge
swp7:VlanA-1         server02   mac:00:02:00:00:00:2a netcat-
8081          172.17.0.3           bridge
swp7:VlanA-1         server02   mac:00:02:00:00:00:2a netcat-
8080          172.17.0.2           bridge
swp7:VlanA-1         server02   mac:00:02:00:00:00:2a netcat-
8086          172.17.0.8           bridge
swp7:VlanA-1         server02   mac:00:02:00:00:00:2a netcat-
8088          172.17.0.10          bridge
swp7:VlanA-1         server02   mac:00:02:00:00:00:2a netcat-
8082          172.17.0.4           bridge
swp7:VlanA-1         server02   mac:00:02:00:00:00:2a netcat-
8091          172.17.0.13          bridge
```

```
swp7:VlanA-1         server02   mac:00:02:00:00:00:2a netcat-
8090          172.17.0.12          bridge
swp7:VlanA-1         server02   mac:00:02:00:00:00:2a netcat-
8087          172.17.0.9           bridge
swp8:VlanA-1         server03   mac:00:02:00:00:00:2d netcat-
8091          172.17.0.13          bridge
swp8:VlanA-1         server03   mac:00:02:00:00:00:2d netcat-
8083          172.17.0.5           bridge
swp8:VlanA-1         server03   mac:00:02:00:00:00:2d netcat-
8087          172.17.0.9           bridge
swp8:VlanA-1         server03   mac:00:02:00:00:00:2d netcat-
8082          172.17.0.4           bridge
swp8:VlanA-1         server03   mac:00:02:00:00:00:2d netcat-
8080          172.17.0.2           bridge
swp8:VlanA-1         server03   mac:00:02:00:00:00:2d netcat-
8092          172.17.0.14          bridge
swp8:VlanA-1         server03   mac:00:02:00:00:00:2d netcat-
8086          172.17.0.8           bridge
swp8:VlanA-1         server03   mac:00:02:00:00:00:2d netcat-
8084          172.17.0.6           bridge
swp8:VlanA-1         server03   mac:00:02:00:00:00:2d netcat-
8088          172.17.0.10          bridge
swp8:VlanA-1         server03   mac:00:02:00:00:00:2d netcat-
8090          172.17.0.12          bridge
swp8:VlanA-1         server03   mac:00:02:00:00:00:2d netcat-
8085          172.17.0.7           bridge
swp8:VlanA-1         server03   mac:00:02:00:00:00:2d netcat-
8089          172.17.0.11          bridge
swp8:VlanA-1         server03   mac:00:02:00:00:00:2d netcat-
8081          172.17.0.3           bridge
```

Showing Container-specific Information

You can see information about a given container by running `netq show docker container name NAME`:

```
cumulus@server01:~$ netq show docker container name netcat-9092
Name                 Node       IP                 IP Masq.
Network        Service Name    Up time
-------------------- ---------- ----------------- --------
-------------- --------------- ---------------
netcat-9092          exit01     45.0.0.17/26,      False
host                           0:34:15
                                27.0.0.3/32,
                                192.168.0.15/24
```

Showing Containers with a Specific Image

To search for all the containers on the network with a specific Docker image, run `netq show docker container image IMAGE_NAME`:

```
cumulus@server01:~$ netq show docker container image chilcano/netcat:
jessie
Name                 Node        IP                IP Masq.
Network        Service Name    Up time
-------------------- ----------  ----------------- --------
-------------- --------------- ---------------
netcat-8080          exit01     172.17.0.2        True
bridge                         0:32:09
netcat-8080          server01   172.17.0.2        True
bridge                         0:23:11
netcat-8081          exit01     172.17.0.3        True
bridge                         0:32:07
netcat-8081          server01   172.17.0.3        True
bridge                         0:23:10
netcat-8082          exit01     172.17.0.4        True
bridge                         0:32:08
netcat-8082          server01   172.17.0.4        True
bridge                         0:23:08
netcat-8083          exit01     172.17.0.5        True
bridge                         0:32:07
netcat-8083          server01   172.17.0.5        True
bridge                         0:23:07
netcat-8084          exit01     172.17.0.6        True
bridge                         0:32:07
netcat-8084          server01   172.17.0.6        True
bridge                         0:23:09
netcat-8085          exit01     172.17.0.7        True
bridge                         0:32:05
netcat-8085          server01   172.17.0.7        True
bridge                         0:23:06
netcat-8086          exit01     172.17.0.8        True
bridge                         0:32:06
netcat-8086          server01   172.17.0.8        True
bridge                         0:23:06
netcat-8087          exit01     172.17.0.9        True
bridge                         0:32:05
netcat-8087          server01   172.17.0.9        True
bridge                         0:23:06
netcat-8088          exit01     172.17.0.10       True
bridge                         0:32:04
netcat-8088          server01   172.17.0.10       True
bridge                         0:23:06
netcat-8089          exit01     172.17.0.11       True
bridge                         0:32:02
```

```
netcat-8089          server01   172.17.0.11       True
bridge                         0:23:03
netcat-8090          exit01     172.17.0.12       True
bridge                         0:32:01
netcat-8090          server01   172.17.0.12       True
bridge                         0:23:05
netcat-8091          exit01     172.17.0.13       True
bridge                         0:32:03
netcat-8091          server01   172.17.0.13       True
bridge                         0:23:04
netcat-8092          exit01     172.17.0.14       True
bridge                         0:31:59
netcat-8092          server01   172.17.0.14       True
bridge                         0:23:03
netcat-9080          exit01     45.0.0.17/26,     False
host                           0:31:51
                                27.0.0.3/32,
                                192.168.0.15/24
netcat-9081          exit01     45.0.0.17/26,     False
host                           0:31:51
                                27.0.0.3/32,
                                192.168.0.15/24
netcat-9082          exit01     45.0.0.17/26,     False
host                           0:31:52
                                27.0.0.3/32,
                                192.168.0.15/24
netcat-9083          exit01     45.0.0.17/26,     False
host                           0:31:49
                                27.0.0.3/32,
                                192.168.0.15/24
netcat-9084          exit01     45.0.0.17/26,     False
host                           0:31:50
                                27.0.0.3/32,
                                192.168.0.15/24
netcat-9085          exit01     45.0.0.17/26,     False
host                           0:31:50
                                27.0.0.3/32,
                                192.168.0.15/24
netcat-9086          exit01     45.0.0.17/26,     False
host                           0:31:48
                                27.0.0.3/32,
                                192.168.0.15/24
netcat-9087          exit01     45.0.0.17/26,     False
host                           0:31:48
                                27.0.0.3/32,
                                192.168.0.15/24
netcat-9088          exit01     45.0.0.17/26,     False
host                           0:31:47
                                27.0.0.3/32,
                                192.168.0.15/24
netcat-9089          exit01     45.0.0.17/26,     False
host                           0:31:48
```

```
                                 27.0.0.3/32,
                                 192.168.0.15/24
netcat-9090          exit01      45.0.0.17/26,     False
host                            0:31:46
                                 27.0.0.3/32,
                                 192.168.0.15/24
netcat-9091          exit01      45.0.0.17/26,     False
host                            0:31:47
                                 27.0.0.3/32,
                                 192.168.0.15/24
netcat-9092          exit01      45.0.0.17/26,     False
host                            0:31:47
                                 27.0.0.3/32,
                                 192.168.0.15/24
```

Showing Container Connectivity

To determine how a particular container is attached to a network, run `netq HOST show docker container network NAME connectivity`. The output tells you what host it's launched on, adjacent nodes, adjacent ports.

```
cumulus@leaf01:~$ netq server01 show docker container network host
connectivity
Name            Swarm Service Cont IP         Network    Node
Port                 Peer Node  Peer Port
--------------- ------------- --------------- ---------- ----------
-------------------- ---------- --------------------
netcat-9080                                   host       server01
swp2:NetQBond-1      noc-pr     swp21:NetQBond-19
netcat-9080                                   host       server01
swp3:NetQBond-1      noc-se     swp21:NetQBond-19
netcat-9080                                   host       server01
swp1:swp1            tor-1      Local Node tor-1 and

Ports swp6 <==> Remo

te  Node/s hosts-11

and Ports swp1
netcat-9081                                   host       server01
swp2:NetQBond-1      noc-pr     swp21:NetQBond-19
netcat-9081                                   host       server01
swp3:NetQBond-1      noc-se     swp21:NetQBond-19
netcat-9081                                   host       server01
swp1:swp1            tor-1      Local Node tor-1 and

Ports swp6 <==> Remo

te  Node/s hosts-11
```

```
and Ports swp1
netcat-9082                                   host       server01
swp2:NetQBond-1      noc-pr     swp21:NetQBond-19
netcat-9082                                   host       server01
swp3:NetQBond-1      noc-se     swp21:NetQBond-19
netcat-9082                                   host       server01
swp1:swp1            tor-1      Local Node tor-1 and

Ports swp6 <==> Remo

te  Node/s hosts-11

and Ports swp1
netcat-9083                                   host       server01
swp2:NetQBond-1      noc-pr     swp21:NetQBond-19
netcat-9083                                   host       server01
swp3:NetQBond-1      noc-se     swp21:NetQBond-19
netcat-9083                                   host       server01
swp1:swp1            tor-1      Local Node tor-1 and

Ports swp6 <==> Remo

te  Node/s hosts-11

and Ports swp1
netcat-9084                                   host       server01
swp2:NetQBond-1      noc-pr     swp21:NetQBond-19
netcat-9084                                   host       server01
swp3:NetQBond-1      noc-se     swp21:NetQBond-19
netcat-9084                                   host       server01
swp1:swp1            tor-1      Local Node tor-1 and

Ports swp6 <==> Remo

te  Node/s hosts-11

and Ports swp1
netcat-9085                                   host       server01
swp2:NetQBond-1      noc-pr     swp21:NetQBond-19
netcat-9085                                   host       server01
swp3:NetQBond-1      noc-se     swp21:NetQBond-19
netcat-9085                                   host       server01
swp1:swp1            tor-1      Local Node tor-1 and

Ports swp6 <==> Remo

te  Node/s hosts-11

and Ports swp1
netcat-9086                                   host       server01
swp2:NetQBond-1      noc-pr     swp21:NetQBond-19
```

```
netcat-9086                                   host       server01
swp3:NetQBond-1      noc-se     swp21:NetQBond-19
netcat-9086                                   host       server01
swp1:swp1            tor-1      Local Node tor-1 and

Ports swp6 <==> Remo

te  Node/s hosts-11

and Ports swp1
netcat-9087                                   host       server01
swp2:NetQBond-1      noc-pr     swp21:NetQBond-19
netcat-9087                                   host       server01
swp3:NetQBond-1      noc-se     swp21:NetQBond-19
netcat-9087                                   host       server01
swp1:swp1            tor-1      Local Node tor-1 and

Ports swp6 <==> Remo

te  Node/s hosts-11

and Ports swp1
netcat-9088                                   host       server01
swp2:NetQBond-1      noc-pr     swp21:NetQBond-19
netcat-9088                                   host       server01
swp3:NetQBond-1      noc-se     swp21:NetQBond-19
netcat-9088                                   host       server01
swp1:swp1            tor-1      Local Node tor-1 and

Ports swp6 <==> Remo

te  Node/s hosts-11

and Ports swp1
netcat-9089                                   host       server01
swp2:NetQBond-1      noc-pr     swp21:NetQBond-19
netcat-9089                                   host       server01
swp3:NetQBond-1      noc-se     swp21:NetQBond-19
netcat-9089                                   host       server01
swp1:swp1            tor-1      Local Node tor-1 and

Ports swp6 <==> Remo

te  Node/s hosts-11

and Ports swp1
netcat-9090                                   host       server01
swp2:NetQBond-1      noc-pr     swp21:NetQBond-19
netcat-9090                                   host       server01
swp3:NetQBond-1      noc-se     swp21:NetQBond-19
netcat-9090                                   host       server01
swp1:swp1            tor-1      Local Node tor-1 and
```

```
Ports swp6 <==> Remo

te  Node/s hosts-11

and Ports swp1
netcat-9091                                   host       server01
swp2:NetQBond-1      noc-pr     swp21:NetQBond-19
netcat-9091                                   host       server01
swp3:NetQBond-1      noc-se     swp21:NetQBond-19
netcat-9091                                   host       server01
swp1:swp1            tor-1      Local Node tor-1 and

Ports swp6 <==> Remo

te  Node/s hosts-11

and Ports swp1
netcat-9092                                   host       server01
swp2:NetQBond-1      noc-pr     swp21:NetQBond-19
netcat-9092                                   host       server01
swp3:NetQBond-1      noc-se     swp21:NetQBond-19
netcat-9092                                   host       server01
swp1:swp1            tor-1      Local Node tor-1 and

Ports swp6 <==> Remo

te  Node/s hosts-11

and Ports swp1
```

Checking Network Traffic over a Given Protocol

You can include the protocol when you observe a given flow of traffic on the network and want to identify which container sent or received traffic using that protocol from a given port.

```
cumulus@tor-1:mgmt-vrf:~$ netq hosts-11 show docker container 6.0.1.5
tcp
Container Name       Node       Proto  Port     Cont IP
Network        Host IP              Host Port
-------------------- ---------- ------ -------- -----------------
-------------- --------------------- ------------
netcat-9080          server01   tcp    9192
host           6.0.1.5/26:swp1.1004  9192
netcat-9080          server01   tcp    8182
host           6.0.1.5/26:swp1.1004  8182
netcat-9081          server01   tcp    9192
host           6.0.1.5/26:swp1.1004  9192
```

```
netcat-9081          server01   tcp    8182
host           6.0.1.5/26:swp1.1004  8182
netcat-9082          server01   tcp    8182
host           6.0.1.5/26:swp1.1004  8182
netcat-9082          server01   tcp    9192
host           6.0.1.5/26:swp1.1004  9192
netcat-9083          server01   tcp    8182
host           6.0.1.5/26:swp1.1004  8182
netcat-9083          server01   tcp    9192
host           6.0.1.5/26:swp1.1004  9192
netcat-9084          server01   tcp    9192
host           6.0.1.5/26:swp1.1004  9192
netcat-9084          server01   tcp    8182
host           6.0.1.5/26:swp1.1004  8182
netcat-9085          server01   tcp    8182
host           6.0.1.5/26:swp1.1004  8182
netcat-9085          server01   tcp    9192
host           6.0.1.5/26:swp1.1004  9192
netcat-9086          server01   tcp    9192
host           6.0.1.5/26:swp1.1004  9192
netcat-9086          server01   tcp    8182
host           6.0.1.5/26:swp1.1004  8182
netcat-9087          server01   tcp    9192
host           6.0.1.5/26:swp1.1004  9192
netcat-9087          server01   tcp    8182
host           6.0.1.5/26:swp1.1004  8182
netcat-9088          server01   tcp    9192
host           6.0.1.5/26:swp1.1004  9192
netcat-9088          server01   tcp    8182
host           6.0.1.5/26:swp1.1004  8182
netcat-9089          server01   tcp    8182
host           6.0.1.5/26:swp1.1004  8182
netcat-9089          server01   tcp    9192
host           6.0.1.5/26:swp1.1004  9192
netcat-9090          server01   tcp    8182
host           6.0.1.5/26:swp1.1004  8182
netcat-9090          server01   tcp    9192
host           6.0.1.5/26:swp1.1004  9192
netcat-9091          server01   tcp    9192
host           6.0.1.5/26:swp1.1004  9192
netcat-9091          server01   tcp    8182
host           6.0.1.5/26:swp1.1004  8182
netcat-9092          server01   tcp    9192
host           6.0.1.5/26:swp1.1004  9192
netcat-9092          server01   tcp    8182
host           6.0.1.5/26:swp1.1004  8182
```

Configuring the NetQ Virtual Environment

A virtual NetQ demo environment is available on the Cumulus Networks GitHub site, allowing you to try out NetQ on your own, or to test/validate updates to your network before deploying them into production.

The environment uses a series of Cumulus VX virtual machines built using the Cumulus Networks reference topology. This section provides high level instructions for installing and configuring the virtual environment.

Contents

This chapter covers ...

- Setting up the Demo Environment with Vagrant and VirtualBox (see page 79)

Setting up the Demo Environment with Vagrant and VirtualBox

> ⚠ These steps assume that both Vagrant and VirtualBox have been downloaded. For more information on downloading Vagrant and VirtualBox, refer to the Cumulus VX Getting Started documentation.

1. Download the NetQ Telemetry Server, available from the *NetQ Virtual* option in the **Product** menu of the Cumulus Networks website.
2. In a terminal, add the NetQ Telemetry Server to Vagrant:

   ```
   user@machine:~$ vagrant box add cumulus-netq-telemetry-server-
   amd64-1.0.0-vagrant.box --name=cumulus/ts
   ```

3. Clone the demo:

   ```
   user@machine:~$ git clone https://github.com/cumulusnetworks
   /netqdemo-1.0 netqdemo
   ```

4. From the `netqdemo` directory, run the `vagrant up` command to start the demo.

   ```
   user@machine:~$ cd netqdemo
   user@machine:~/netqdemo$ vagrant up
   ```

5. Once the vagrant instance has started, ssh into the NetQ Telemetry Server, which serves as the `oob-mgmt-server` in the topology:

```
user@machine:~$ vagrant ssh oob-mgmt-server
```

Restoring from Backups with NetQ

NetQ automatically takes snapshots of the NetQ Telemetry Server at five minute intervals. These snapshots can be used to restore to a previous configuration, or to diagnose existing issues with the configuration. For information regarding how long snapshot data is stored, refer to the How Far Back in Time Can You Travel section.

> ⚠ There are no configuration steps required for setting up backups. NetQ snapshots occur automatically.

Backup Locations

Backup snapshots can be found in two file locations on the NetQ Telemetry Server:

- `/var/log/backup`: The latest, or master, snapshot.
- `/var/backup`: Directory of previous snapshots.

Use Cases

There are several use-cases in which restoring from a snapshot may be warranted. These include:

- Upgrading the physical server to increase available resources.
- Migrating from one physical server to another.
- A NetQ Telemetry Server crash.

Restoring from a Snapshot

The following steps outline the process for restoring the NetQ Telemetry Server from a snapshot:

1. Extract the GZip snapshot you wish to restore into a file called `appendonly.aof`. The example command below uses the master snapshot:

   ```
   root@cumulus:~# gzip -d < /var/backup/appendonly.aof_master_2017-06-06_054601.gz > appendonly.aof
   ```

   The snapshot filename has several parts:

   - `appendonly.aof`: The base file name.
   - `_master_`: Defines this file as the current master snapshot.
   - `2017-06-06_054601`: The date and time the snapshot was taken.
2. Shutdown the NetQ stack:

```
root@cumulus:~# sudo systemctl stop netq-appliance
```

3. Copy the extracted **appendonly.aof** file into the data directory:

```
root@cumulus:~# cp appendonly.aof /var/data/redis/master
/appendonly.aof
```

4. Remove the **dump.rmb** file from the master directory, if the file is present:

```
root@cumulus:~# rm -f /var/data/redis/master/dump.rdb
```

5. Use the **grep** command to confirm the Redis configuration is still set correctly:

```
root@cumulus:~# grep appendonly /etc/cts/redis/*conf
/etc/cts/redis/redis.conf:appendonly yes
/etc/cts/redis/redis.conf:appendfilename "appendonly.aof"
root@cumulus:~# grep 'save ""' /etc/cts/redis/*conf
/etc/cts/redis/redis.conf:save ""
```

6. Restart the NetQ Stack:

```
root@cumulus:~# sudo systemctl start netq-appliance
```

Troubleshooting NetQ

To aid in troubleshooting issues with NetQ, there are several configuration and log files on the telemetry server that can provide insight into the root cause of the issue:

File	Description
/appliance/cfg/netq/netq.yml	The NetQ Telemetry Server configuration file.
/var/log/docker/netq/cli_1/netqd.log	The NetQ daemon log file for the NetQ CLI.
/var/log/docker/netq/notifier_1/netq-notifier.log	The NetQ Notifier log file.

A node running the NetQ Agent has the following configuration and log files:

File	Description
/etc/netq/netq.yml	The NetQ configuration file.
/var/log/netq-agent.log	The NetQ Agent log file.
/var/log/netqd.log	The NetQ daemon log file.
/etc/netq/config.d/netq-agent-commands.yml	Contains key-value command pairs and relevant custom configuration settings.
/run/netq-agent-running.json	Contains the full command list that will be pushed when the agent starts.

Checking Agent Health

Checking the health of the NetQ agents is a good way to start troubleshooting NetQ on your network. If any agents are rotten, meaning three heartbeats in a row were not sent, then you can investigate the rotten node. In the example below, the NetQ Agent on server01 is rotten, so you know where to start looking for problems:

```
netq@446c0319c06a:/$ netq check agents
Checked nodes: 12,
Rotten nodes: 1
netq@446c0319c06a:/$ netq show agents
Node      Status    Sys Uptime    Agent Uptime
--------  --------  ------------  --------------
exit01
Fresh
     8h ago        4h ago
exit02
Fresh
     8h ago        4h ago
leaf01
Fresh
     8h ago        4h ago
leaf02
Fresh
     8h ago        4h ago
leaf03
Fresh
     8h ago        4h ago
leaf04
Fresh
     8h ago        4h ago
server01
Rotten
    4h ago        4h ago
server02
Fresh
     4h ago        4h ago
server03
Fresh
     4h ago        4h ago
server04
Fresh
     4h ago        4h ago
spine01
Fresh
     8h ago        4h ago
spine02
Fresh
     8h ago        4h ago
```

Error Configuring the Telemetry Server on a Node

If you get an error when your run the `netq add server` command on a node, it's usually due to one of two reasons:

- The hostname or IP address for the telemetry server was input incorrectly when you ran `netq add server`. Check what you input and try again.
- The telemetry server isn't responding. Try pinging the IP address you entered and see if the ping works.

netq-support

The `netq-support` command generates an archive of useful information for troubleshooting issues with NetQ. 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:

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

⚠ The `netq-support` script generates a file called `cl-support`.

Index