



# Cumulus NetQ 1.4.0 Deployment Guide

---



# Table of Contents

<b>Deployment Preface</b>	<b>4</b>
Contents	5
What's New in Cumulus NetQ 1.4.0	5
Available Documentation	5
Document Formatting	6
Typographical Conventions	6
Note Conventions	6
<b>Install NetQ</b>	<b>7</b>
Contents	8
Prerequisites	8
Hardware Support	9
Operating System Support	9
NetQ Application Support	9
Install Workflow	9
Install the NetQ Telemetry Server	11
Install Options	20
Install the NetQ Agent	21
Install NetQ Agent on a Cumulus Linux Switch	21
Install NetQ Agent on an Ubuntu Server (Optional)	22
Install NetQ Agent on a Red Hat or CentOS Server (Optional)	23
Set Up the NetQ Agents	25
Basic Configuration	25
Configure the Agent to Use a VRF (Optional)	26
Configure the Agent to Communicate over a Specific Port (Optional)	26
Enabling Docker for Container Environments	27
Set Up Security	28
<b>Configure Optional NetQ Capabilities</b>	<b>28</b>
Contents	29
Activate Early Access Features	30
Integrate NetQ with an Event Notification Application	31
NetQ Notifier	31
NetQ Notifier CLI Overview	33
Configure NetQ Notifier Logging	34
Configure PagerDuty Using NetQ Notifier CLI	36
Configure Slack Using NetQ Notifier CLI	38
Configure PagerDuty Manually	40
Configure Slack Manually	41
View Integrations	42
Export Notifications to ELK	42



Export Notifications to Splunk .....	44
Create NetQ Notifier Filters .....	44
Build Filter Rules .....	46
Specify Output Location .....	48
Example: Create a Filter for Events from Selected Switches .....	48
Example: Verify NetQ Notifier Status .....	48
Example: Add Multiple Rules to a Filter .....	48
Example: Place One Filter Before or After Another Filter .....	49
Example: Create a Filter to Drop Notifications from a Given Interface .....	50
Example: Create a Filter to Send BGP Session State Notifications to Slack .....	51
Example: Create a Filter to Drop All Temperature-Related Event Notifications .....	52
Example: Create a Filter for License Validation Event Notifications .....	52
Configure High Availability Mode .....	53
Configure HA Mode .....	53
Check Database Cluster Status .....	54
Restart HA Mode Services .....	55
Change the Master Telemetry Server .....	55
Replace a Replica with a New Server .....	56
Reset the Database Cluster .....	57
Troubleshoot HA Mode .....	58
Integrate with a Hardware Chassis .....	59
Extending NetQ with Custom Services Using curl .....	59
<b>Deployment Appendices .....</b>	<b>61</b>
Contents .....	62
Example NetQ Switch Configuration File .....	62
Example Ansible Playbook to Install NetQ on All Cumulus Switches .....	68

©2018 Cumulus Networks. All rights reserved

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



This guide is intended for network administrators who are responsible for installation, setup, and maintenance of Cumulus NetQ in their data center environment. NetQ offers the ability to monitor and manage your data center network infrastructure and operational health with simple tools based on open source Linux. This guide provides instructions and information about installing NetQ core capabilities, configuring optional capabilities, and upgrading an existing NetQ installation. It assumes you have already installed Cumulus Linux on your network switches and you are ready to add these NetQ capabilities.

This guide is organized into the following topics:

- [Deployment Preface \(see page 4\)](#)
- [Install NetQ \(see page 7\)](#)
- [Configure Optional NetQ Capabilities \(see page 28\)](#)
- [Maintain NetQ](#)
- [Deployment Appendices \(see page 61\)](#)

Information about monitoring and troubleshooting your network using NetQ is contained in the associated user guide.



Before you get started, you should review the [release notes](#) for this version.

# Deployment Preface

A variety of resources are available for you to become familiar with Cumulus NetQ and aid in its deployment. These are identified here along with information about how the content is presented.

## Contents

This topic describes...

- [What's New in Cumulus NetQ 1.4.0 \(see page 5\)](#)
- [Available Documentation \(see page 5\)](#)
- [Document Formatting \(see page 6\)](#)
  - [Typographical Conventions \(see page 6\)](#)
  - [Note Conventions \(see page 6\)](#)

## What's New in Cumulus NetQ 1.4.0

Cumulus NetQ 1.4.0 includes the following new features and updates:

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

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

## Available Documentation

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

You may start anywhere in the documentation or read it from start to finish depending on your role and familiarity with the NetQ software and Linux networking. If you are new to NetQ, you may want to read the Cumulus NetQ Primer before reading the other available documents.

The following NetQ documents are available:

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

## Document Formatting

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

### Typographical Conventions

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

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

### Note Conventions

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

#### ✓ Tip or Best Practice

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

#### i Information

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



### Caution

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



### Warning

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

# Install NetQ

This topic provides step-by-step instructions for installing Cumulus NetQ 1.4 with Cumulus Linux 3.3.2 or later. It provides setup instructions for the Telemetry Server, for the Cumulus Linux switches, and for the Linux-based hosts.

Cumulus NetQ core telemetry and analytics capabilities require two components to be installed to gain access to the network connectivity and actionable insights NetQ provides:

- NetQ Telemetry application installed on the Telemetry Server
- `cumulus-netq` meta package installed on Cumulus<sup>®</sup> Linux<sup>®</sup> switches

Optionally, you can:

- Add greater fabric visibility by installing Net Q, `cumulus-netq` meta package, on host servers running CentOS, Red Hat Enterprise, or Ubuntu Linux Operating Systems
- Add local storage and distribution services for the Cumulus Linux network operating system (NOS) and provisioning scripts used to deploy and upgrade Cumulus Linux and NetQ. See [Cumulus NetQ Image and Provisioning Management User Guide](#) for details.
- Add Free Range (FR) Routing capabilities on your hosts.

## Contents

This topic describes...

- [Prerequisites \(see page 8\)](#)
  - [Hardware Support \(see page 9\)](#)
  - [Operating System Support \(see page 9\)](#)
  - [NetQ Application Support \(see page 9\)](#)
- [Install Workflow \(see page 9\)](#)
- [Install the NetQ Telemetry Server \(see page 11\)](#)
  - [Install Options \(see page 20\)](#)
- [Install the NetQ Agent \(see page 21\)](#)
  - [Install NetQ Agent on a Cumulus Linux Switch \(see page 21\)](#)
  - [Install NetQ Agent on an Ubuntu Server \(Optional\) \(see page 22\)](#)
  - [Install NetQ Agent on a Red Hat or CentOS Server \(Optional\) \(see page 23\)](#)
- [Set Up the NetQ Agents \(see page 25\)](#)
  - [Basic Configuration \(see page 25\)](#)
  - [Configure the Agent to Use a VRF \(Optional\) \(see page 26\)](#)
  - [Configure the Agent to Communicate over a Specific Port \(Optional\) \(see page 26\)](#)
  - [Enabling Docker for Container Environments \(see page 27\)](#)
- [Set Up Security \(see page 28\)](#)





## Prerequisites

### Hardware Support

NetQ is supported on a variety of hardware. Refer to the [Cumulus Hardware Compatibility List](#) for the hardware supported and descriptions of the available options.

### Operating System Support

NetQ 1.4 is supported on the following operating systems:

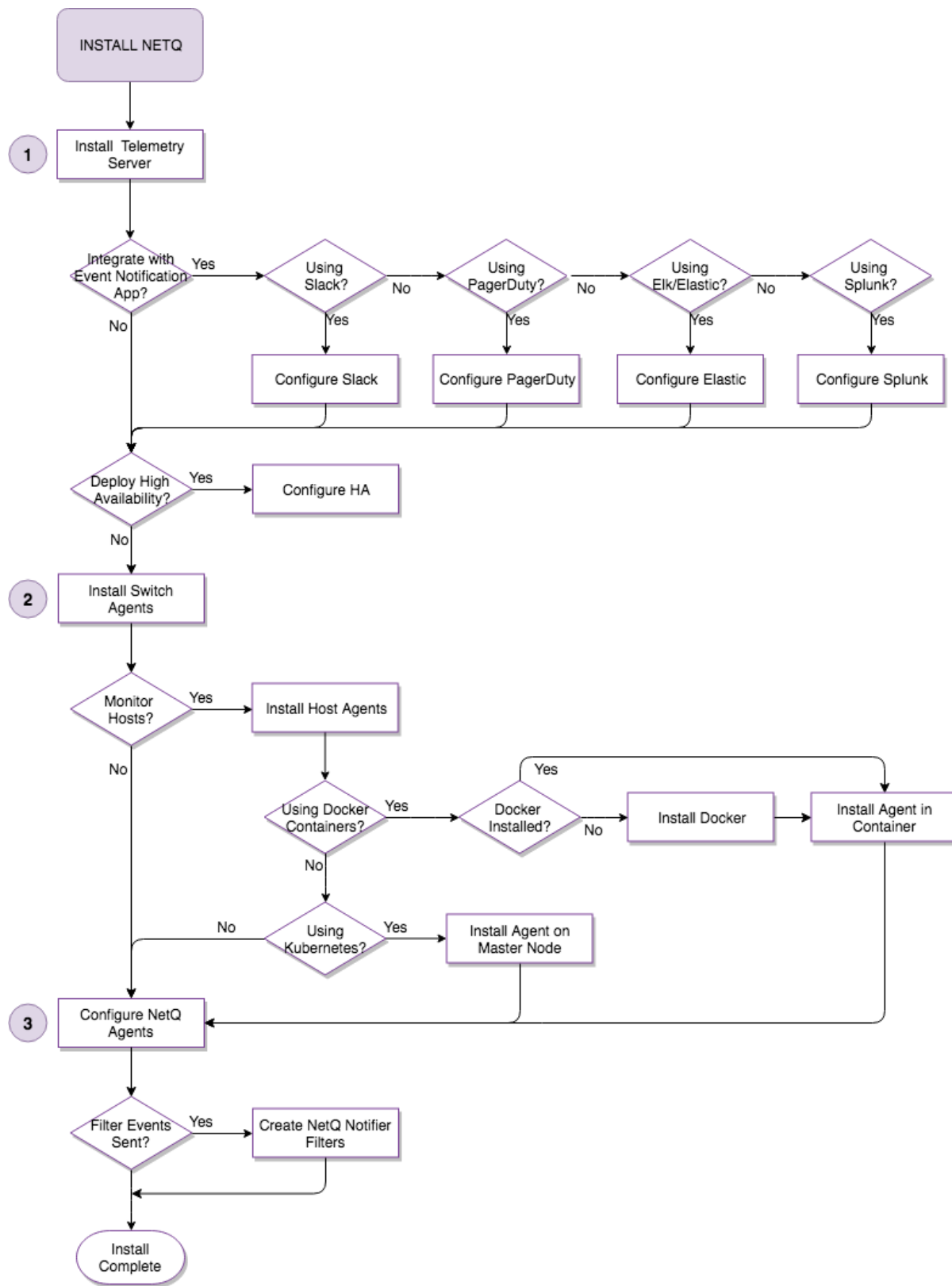
- Cumulus Linux 3.3.2 and later
- Ubuntu 16.04
- Red Hat® Enterprise Linux (RHEL) 7.1
- CentOS 7

### NetQ Application Support

- The NetQ Telemetry application is supported on NetQ 1.0 and later.
- The NetQ Service Console is supported on NetQ 1.0 and later.
- The NetQ Telemetry, Image and Provisioning Management Server (IPM) application (an early access feature) is available with NetQ 1.4 and later.

## Install Workflow

Installation of NetQ involves installing the Telemetry Server, and installing and configuring the NetQ Agents. Additional steps are needed to install NetQ in a [High Availability \(see page 28\)](#) configuration or to [Integrate NetQ with Event Notification Applications \(see page 28\)](#). This flow chart shows the three required steps (the numbered items) and the optional steps to install and setup NetQ to start validating your network.



## Install the NetQ Telemetry Server

The NetQ Telemetry Server is comprised of the following components:

- **NetQ Telemetry application:** network monitoring and analytics functionality
- **NetQ Telemetry CLI:** command line user interface for monitoring network and administering NetQ through a terminal session
- **NetQ Service Console:** web console user interface for monitoring and administering NetQ with the NetQ Shell
- **Redis:** primary data base storage for collected data
- **InfluxDB:** secondary data base storage for network snapshots
- **Authorization:** secure access functionality

The server is available in one of two formats, as a:

- VMware ESXi™ 6.5 virtual machine (VM)
- KVM/QCOW (QEMU Copy on Write) image for use on CentOS, Ubuntu and RHEL hosts

### Best Practice

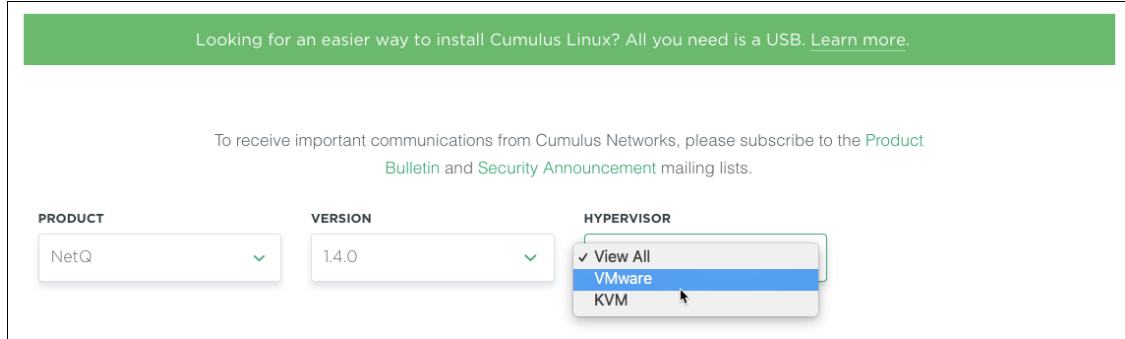
Cumulus Networks recommends you install the Telemetry Server on an out-of-band management network to ensure it can monitor in-band network issues without being affected itself. Ideally, you should run the Telemetry Server on a separate, powerful server for maximum usability and performance. For more information on system requirements, [read this chapter](#).



The NetQ Telemetry Server components reside in containers in the VM. These containers are completely separate from any containers you may have on the hosts you are monitoring with NetQ. The NetQ containers do not overwrite the host containers and vice versa.

To install the Telemetry Server VM:

1. Download the NetQ Telemetry Server (TS) VM.
  - a. On the [Cumulus Downloads](#) page, select *NetQ* from the **Product** list box.
  - b. Optionally, select the latest available version from the **Version** list box.
  - c. Optionally, select the hypervisor you wish to use from the **Hypervisor** list box.

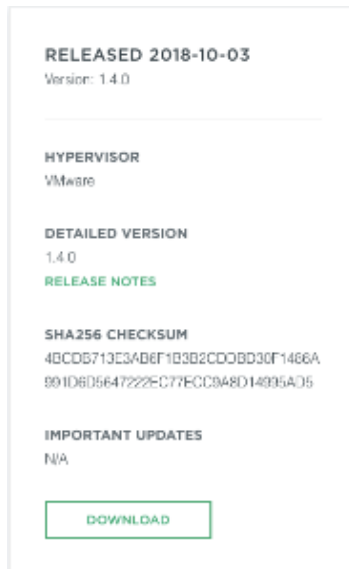


Looking for an easier way to install Cumulus Linux? All you need is a USB. [Learn more.](#)

To receive important communications from Cumulus Networks, please subscribe to the [Product Bulletin](#) and [Security Announcement](#) mailing lists.

PRODUCT	VERSION	HYPERVISOR
NetQ	1.4.0	<div> <input checked="" type="checkbox"/> View All            VMware            KVM         </div>

- d. Scroll down to review the images that match your selection criteria, and click **Download** for the VM you want.



**RELEASED 2018-10-03**  
Version: 1.4.0

---

**HYPERVISOR**  
VMware

**DETAILED VERSION**  
1.4.0

[RELEASE NOTES](#)

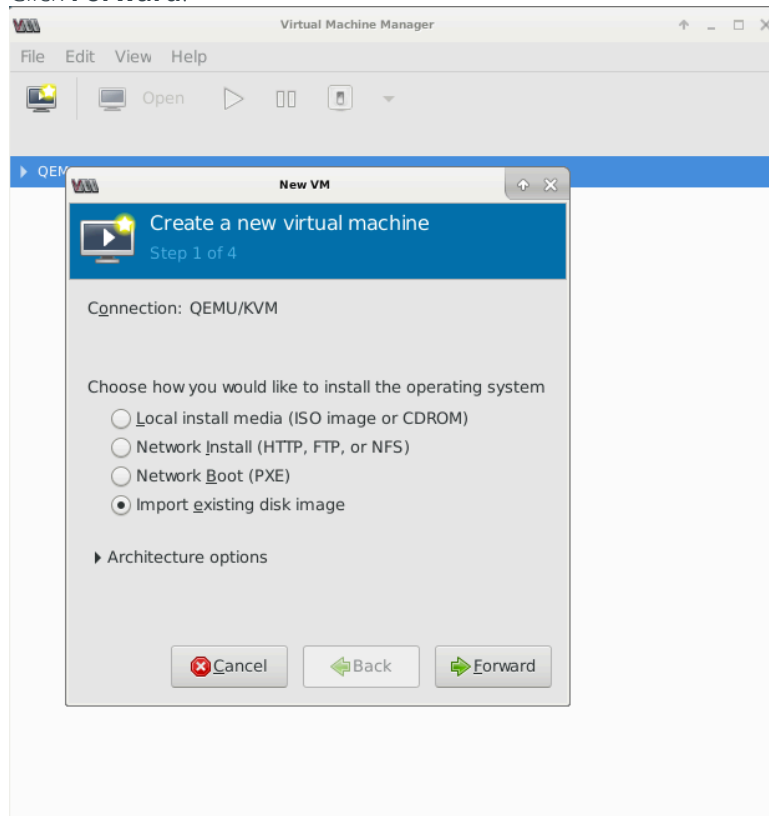
**SHA256 CHECKSUM**  
4BCDB713E34D6F1B3B2CDBD30F1466A  
591D6D5647222EC77ECC948D14935AD5

**IMPORTANT UPDATES**  
N/A

[DOWNLOAD](#)

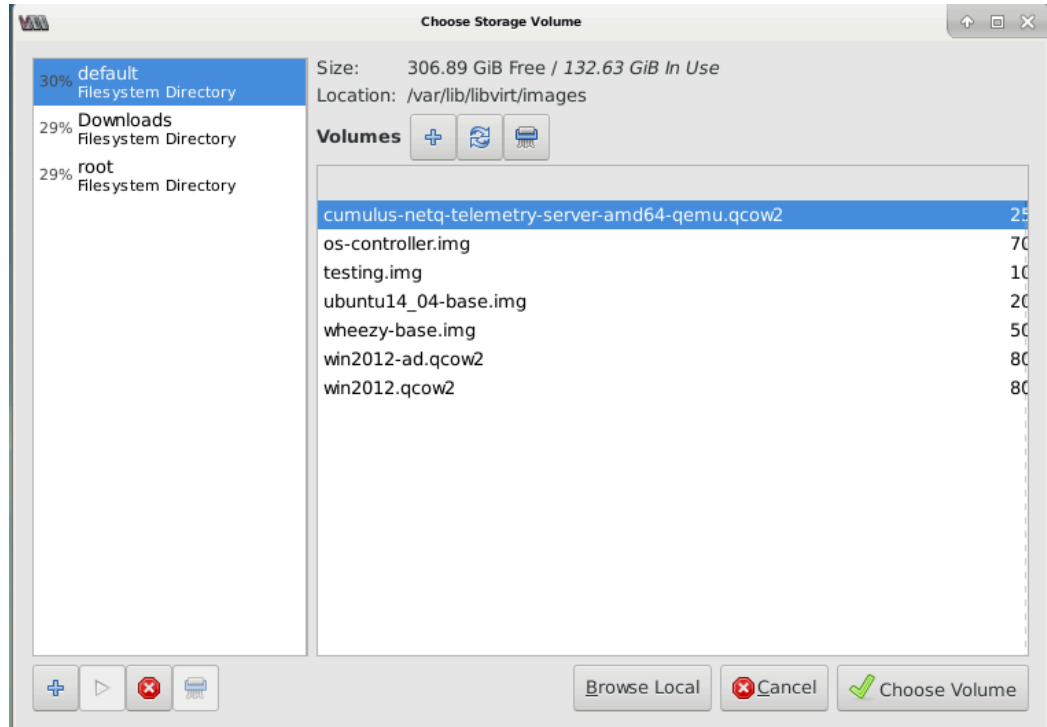
2. Import the VM into your [KVM](#) or [VMware](#) hypervisor.  
This step is shown using KVM with Virtual Machine Manager.
  - a. Open Virtual Machine Manager.

- b. Import the image.
  - i. Select **File > New Virtual Machine**, or click the New VM icon.
  - ii. Select **Import existing disk image**.
  - iii. Click **Forward**.

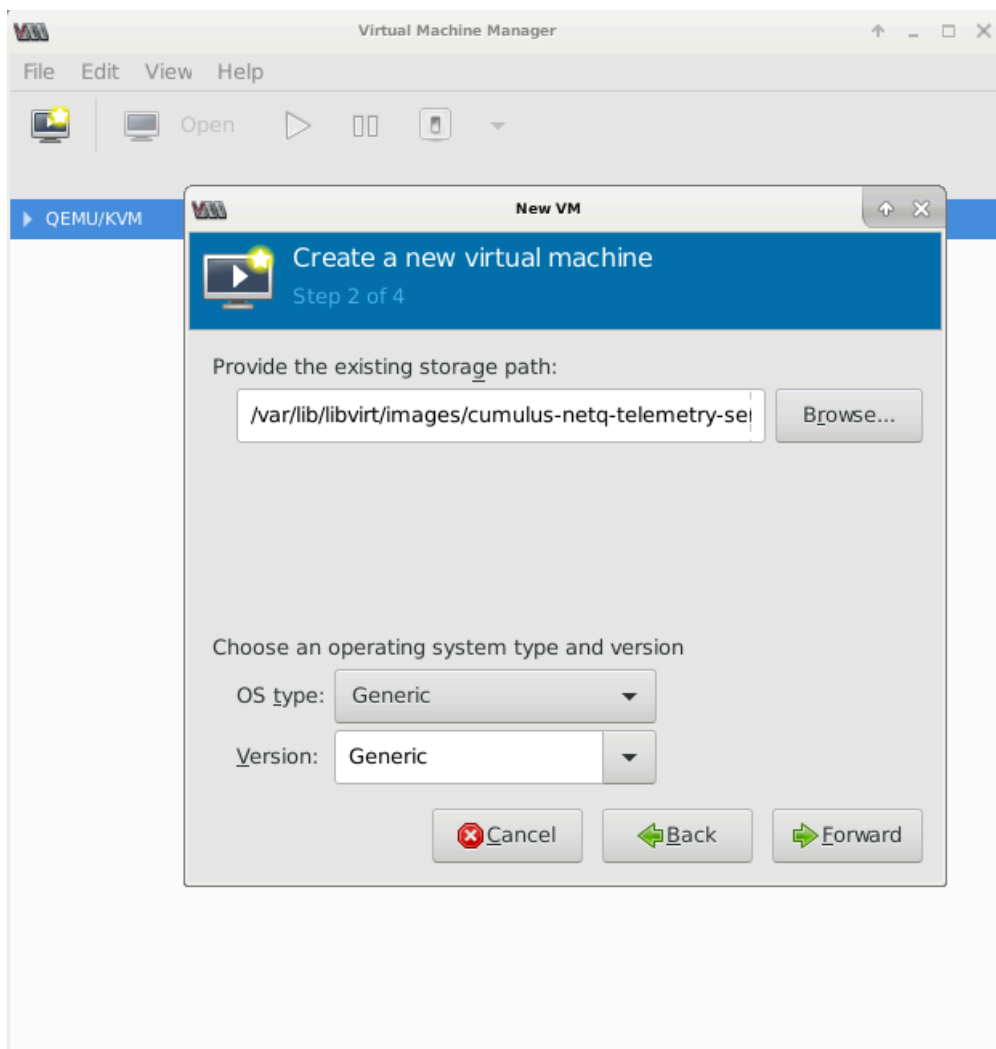


- c. Place the image in the `/var/lib` directory.
  - i. Select the Cumulus image you just downloaded.

ii. Click **Choose Volume**.



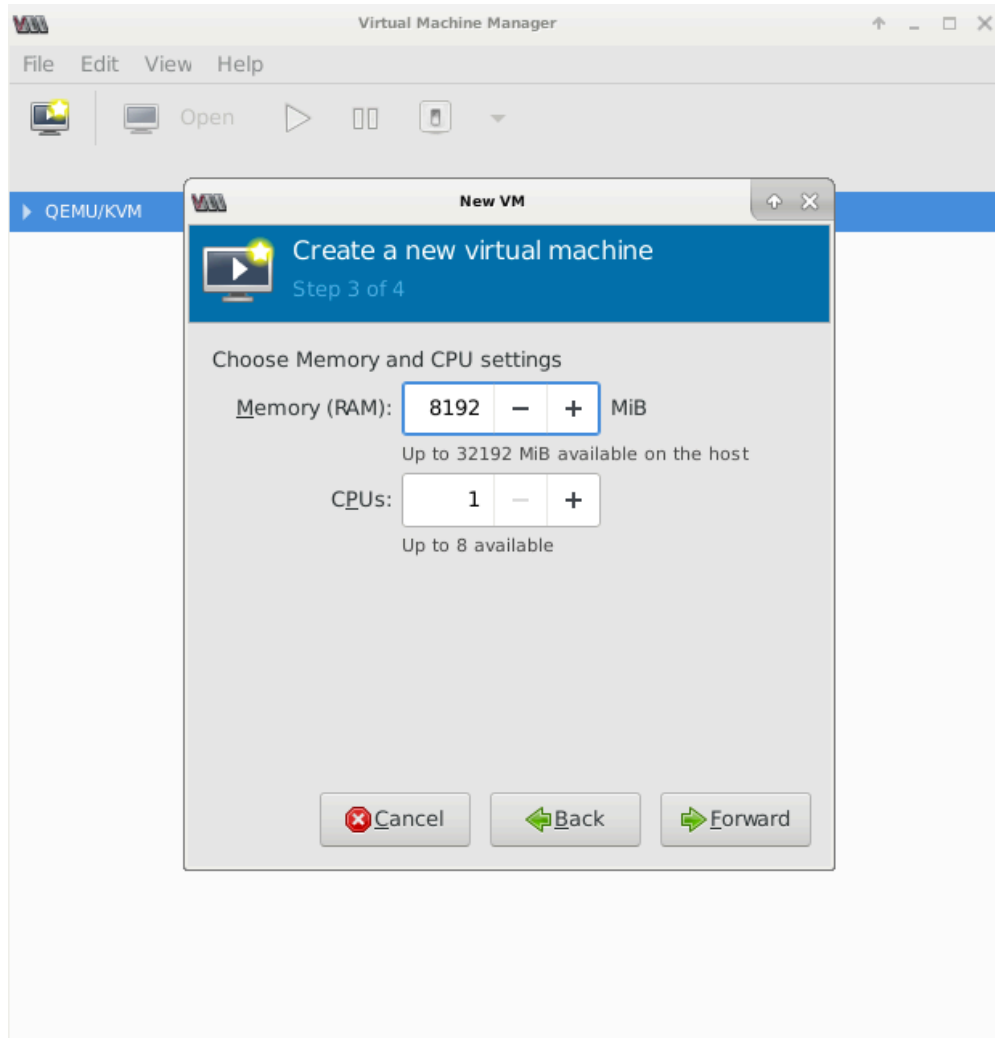
- iii. Type, or browse for, the location where you want to store the volume. The directory must already exist.



- iv. Select **Generic** for the **OS type** and **Version**.
- v. Click **Forward**.

- d. Allocate the amount of memory and number of CPUs you want available to this VM.

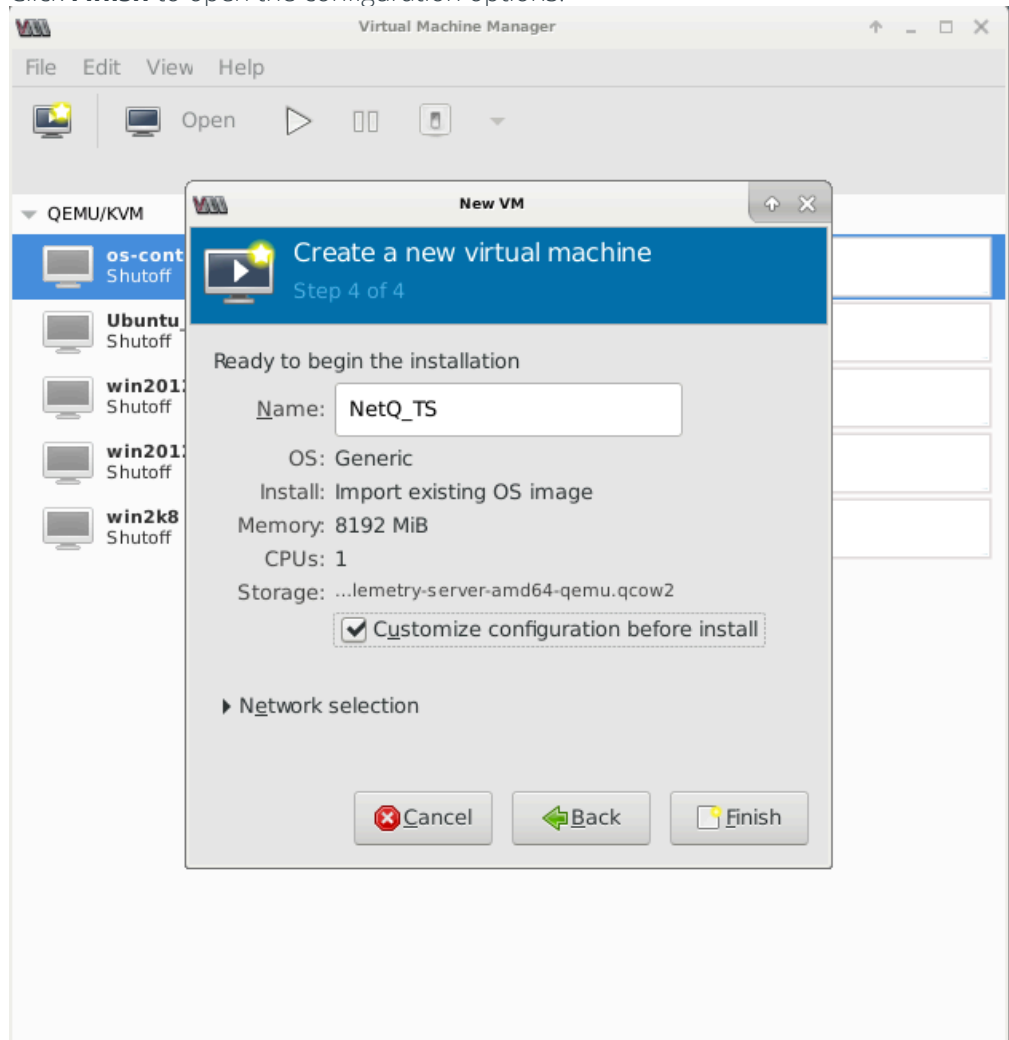
**i** The amount of RAM recommended for the NetQ TS is dependent on your configuration and a number of other criteria; refer to the [Methods for Diagnosing Network Issues](#) topic for more information.



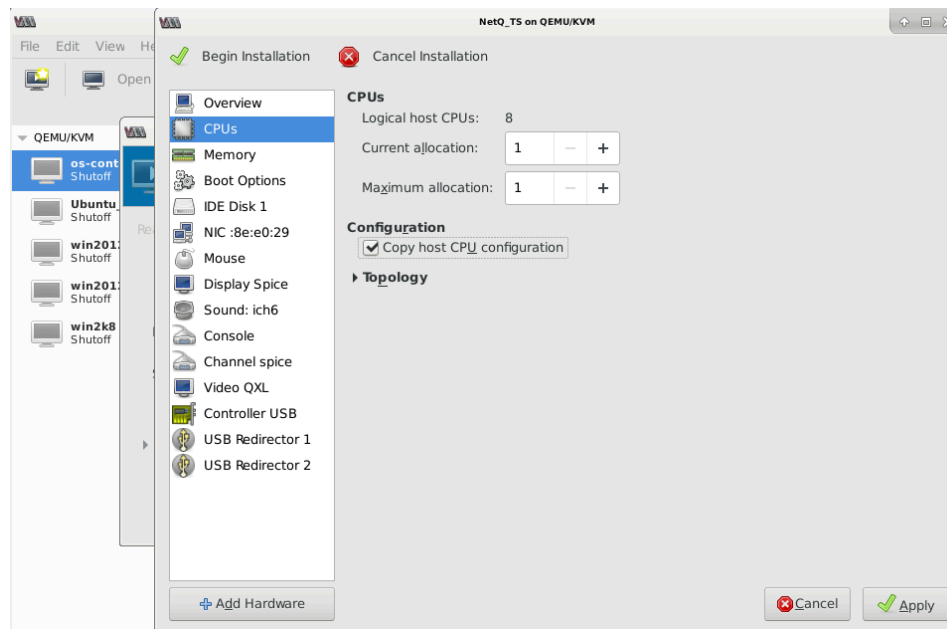
- i. Increase or decrease the amount of **Memory** and **number of CPUs** using the + and - symbols to best meet your environment needs.
- ii. Click **Forward**.



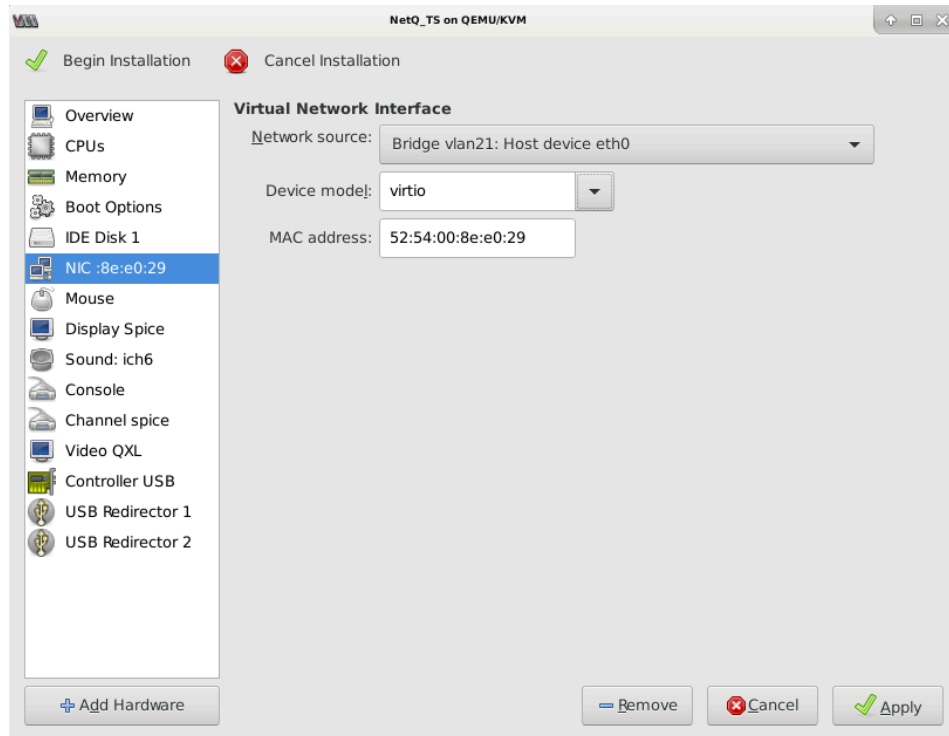
- e. Prepare for installation.
  - i. Provide a unique and useful name for the VM.
  - ii. Select **Customize configuration before install**.
  - iii. Click **Finish** to open the configuration options.



- f. Configure custom CPU parameters.
  - i. Click **CPUs**.
  - ii. Increase or decrease the **Current** and **Maximum allocation** of CPUs using the + and - symbols to best meet your environment needs.
  - iii. Select **Copy host CPU configuration**.
  - iv. Click **Apply**.



- g. Configure custom network interface card (NIC) parameters.
  - i. Click **NIC**.
  - ii. Select the **Network source**.
  - iii. Select or type the **Device model**.
  - iv. Verify the **MAC address** for the NIC.
  - v. Click **Apply**.



3. Verify NetQ TS VM has started.  
 If the VM did not start automatically, click **Begin Installation**.  
 There are two default user accounts you can use to log in:
  - The primary username is *admin*, and its associated password is *CumulusNetQ!*.
  - An alternate username is *cumulus*, and its associated password is *CumulusLinux!*.

4. Note the external IP address of the switch where the TS is running. It is needed to configure the NetQ Agents on each node you want to monitor.

- ✓ The TS obtains its IP address from DHCP. To determine the assigned IP address, log in to the TS and run `ifconfig eth0`. Use the `inet addr` or `inet6 addr` for the TS IP address based on whether you are running IPv4 or IPv6.

```
cumulus@cumulus:~$ ifconfig eth0
eth0      Link encap:Ethernet HWaddr 52:54:00:b8:1e:05
          inet addr:192.168.0.254 Bcast:192.168.0.255 Mask:
          255.255.255.0
          inet6 addr: fe80::5054:ff:feb8:1e05/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
          RX packets:8752 errors:0 dropped:1 overruns:0
frame:0
          TX packets:340 errors:0 dropped:0 overruns:0
carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:567055 (553.7 KiB) TX bytes:34284 (33.4
KiB)
```

For HA mode, you need to note the IP addresses of all three instances of the TS.

If you need the TS to have a static IP address, manually assign one:

- a. Edit the `/etc/network/interfaces` file.

```
root@ts1:~# vi /etc/network/interfaces
```

- b. Add the `address` and `gateway` lines to the `eth0` configuration, specifying the TS's IP address and the IP address of the gateway.

```
auto eth0
iface eth0
    address 198.51.100.10
    gateway 198.51.100.1
```

- c. Save the file and exit.

## Install Options

Two options are available when installing NetQ that require additional configuration:

- High Availability mode
- Integration with third-party notification applications

Once the NetQ Telemetry Server is installed, if you are interested in using the Telemetry Server in high availability (HA) mode, follow the instructions in [Configure High Availability Mode \(see page 28\)](#).

In either standard or HA mode, if you want to proactively monitor events in your network, you can integrate NetQ with PagerDuty, Slack, Elastic, or Splunk. To do so you need to configure both the notification application itself to receive the messages, and the NetQ Notifier with what messages to send and where to send them, *after* installing the NetQ Agents. See [Integrate NetQ with Event Notification Applications \(see page 28\)](#).

## Install the NetQ Agent

The NetQ Agent must be installed on each node you want to monitor. The node can be a:

- Switch running Cumulus Linux version 3.3.2 or later
- Server running Red Hat RHEL 7.1, Ubuntu 16.04 or CentOS 7
- Linux virtual machine running any of the above Linux operating systems

To install the NetQ Agent you need to install an OS-specific meta package, `cumulus-netq`, on each switch. Optionally, you can install it on hosts. The meta package contains the NetQ Agent, the NetQ command line interface (CLI), and the NetQ library. The library contains modules used by both the NetQ Agent and the CLI.

Instructions for installing the meta package on each node type are included here:

- [Install NetQ Agent on a Cumulus Linux Switch \(see page 21\)](#)
- [Install NetQ Agent on an Ubuntu Server \(see page 22\)](#)
- [Install NetQ Agent on a Red Hat or CentOS Server \(see page 23\)](#)



If your network uses a proxy server for external connections, you should first [configure a global proxy](#) so `apt-get` can access the meta package on the Cumulus Networks repository.

## Install NetQ Agent on a Cumulus Linux Switch

A simple two-step process installs the NetQ Agent on a Cumulus switch.

1. On a switch, edit `/etc/apt/sources.list` to add the repository for Cumulus NetQ. Note that NetQ has a separate repository from Cumulus Linux.

```
cumulus@leaf01:~$ sudo nano /etc/apt/sources.list.d
...
deb http://apps3.cumulusnetworks.com/repos/deb CumulusLinux-3
netq-1.4
...
```



The repository `deb http://apps3.cumulusnetworks.com/repos/deb CumulusLinux-3 netq-latest` can be used if you want to always retrieve the latest posted version of NetQ.

2. Update the local `apt` repository, then install the NetQ meta package on the switch.

```
cumulus@leaf01:~$ sudo apt-get update && sudo apt-get install cumulus-netq
```

Repeat these steps for each node, or use an automation tool to install NetQ Agent on multiple nodes. Refer to [Deployment Appendices \(see page 61\)](#) for an example Ansible playbook.

## Install NetQ Agent on an Ubuntu Server (Optional)

Before you install the NetQ Agent on an Ubuntu server, make sure the following packages are installed and running these minimum versions:

- `iproute` 1:4.3.0-1ubuntu3.16.04.1 all
- `iproute2` 4.3.0-1ubuntu3 amd64
- `lldpd` 0.7.19-1 amd64
- `ntp` 1:4.2.8p4+dfsg-3ubuntu5.6 amd64
- `docker-ce` 17.06.1~ce-0~ubuntu amd64



This package is required only if you plan to monitor Docker instances on the host; otherwise do not install it.



Make sure you are running `lldpd`, not `lldpad`. Ubuntu does not include `lldpd` by default, which is required for the installation. To install this package, run the following commands:

```
root@ubuntu:~# apt-get update
root@ubuntu:~# apt-get install lldpd
root@ubuntu:~# systemctl enable lldpd.service
root@ubuntu:~# systemctl start lldpd.service
```

To install the NetQ Agent on an Ubuntu server:

1. Reference and update the local `apt` repository.

```
root@ubuntu:~# wget -O- https://apps3.cumulusnetworks.com/setup/cumulus-apps-deb.pubkey | apt-key add -
```

2. Create the file `/etc/apt/sources.list.d/cumulus-host-ubuntu-xenial.list` and add the following lines:

```
root@ubuntu:~# vi /etc/apt/sources.list.d/cumulus-apps-deb-
xenial.list
...
deb [arch=amd64] https://apps3.cumulusnetworks.com/repos/deb
xenial netq-latest
deb [arch=amd64] https://apps3.cumulusnetworks.com/repos/deb
xenial roh-3
...
```



The use of `netq-latest` in this example means that a get to the repository always retrieves the last version of NetQ, even in the case where a major version update has been made. If you want to keep the repository on a specific version — such as `netq-1.4` — use that instead.

3. Install NTP on the server.

```
root@ubuntu:~# apt install ntp
root@ubuntu:~# systemctl enable ntp
root@ubuntu:~# systemctl start ntp
```

4. Install the meta package on the server.

```
root@ubuntu:~# apt-get update ; apt-get install cumulus-netq
```

5. Restart the NetQ daemon.

```
root@ubuntu:~# systemctl enable netqd ; systemctl restart netqd
```

## Install NetQ Agent on a Red Hat or CentOS Server (Optional)

Before you install the NetQ Agent on a Red Hat or CentOS server, make sure the following packages are installed and running these minimum versions:

- `iproute-3.10.0-54.el7_2.1.x86_64`
- `lldpd-0.9.7-5.el7.x86_64`



Make sure you are running `lldpd`, not `lldpad`.

CentOS does not include `lldpd` by default, nor does it include `wget`, which is required for the installation. To install this package, run the following commands:

```
root@centos:~# yum -y install epel-release
root@centos:~# yum -y install lldpd
```

```
root@centos:~# systemctl enable lldpd.service
root@centos:~# systemctl start lldpd.service
root@centos:~# yum install wget
```

- ntp-4.2.6p5-25.el7.centos.2.x86\_64
- ntpdate-4.2.6p5-25.el7.centos.2.x86\_64

To install the NetQ Agent on a Red Hat or CentOS server:

1. Reference and update the local yum repository.

```
root@rhel7:~# rpm --import https://apps3.cumulusnetworks.com
/setup/cumulus-apps-rpm.pubkey
root@rhel7:~# wget -O- https://apps3.cumulusnetworks.com/setup
/cumulus-apps-rpm-el7.repo > /etc/yum.repos.d/cumulus-host-el.
repo
```

2. Edit /etc/yum.repos.d/cumulus-host-el.repo to set the enabled=1 flag for the two NetQ repositories.

```
root@rhel7:~# vi /etc/yum.repos.d/cumulus-host-el.repo
...
[cumulus-arch-netq-1.1]
name=Cumulus netq packages
baseurl=https://apps3.cumulusnetworks.com/repos/rpm/el/7/netq-1.1
/$basearch
gpgcheck=1
enabled=1
[cumulus-noarch-netq-1.1]
name=Cumulus netq architecture-independent packages
baseurl=https://apps3.cumulusnetworks.com/repos/rpm/el/7/netq-1.1
/noarch
gpgcheck=1
enabled=1
...
```

3. Install NTP on the server.

```
root@rhel7:~# yum install ntp
root@rhel7:~# systemctl enable ntpd
root@rhel7:~# systemctl start ntpd
```

4. Install the Bash completion and NetQ meta packages on the server.

```
root@rhel7:~# yum -y install bash-completion
```



```
root@rhel7:~# yum install cumulus-netq
```

5. Restart the NetQ daemon.

```
root@rhel7:~# systemctl enable netqd ; systemctl restart netqd
```

## Set Up the NetQ Agents

Once the NetQ Agents have been installed on the network nodes you want to monitor, the NetQ Agents must be configured to obtain useful and relevant data. The code examples shown in this section illustrate how to configure the NetQ Agent on a Cumulus switch acting as a host, but it is *exactly* the same for the other type of nodes. Depending on your deployment, follow the relevant additional instructions after the basic configuration steps:

- [Basic Configuration](#)
- [Configuring the Agent to Use a VRF \(see page 26\)](#)
- [Configuring the Agent to Communicate over a Specific Port \(see page 26\)](#)
- [Enabling Docker for Container Environments \(see page \)](#)

## Basic Configuration

This is the minimum configuration required to properly monitor your nodes.

1. Verify that [NTP](#) is running on the host node. Nodes must be in time synchronization with the Telemetry Server to enable useful statistical analysis.

```
cumulus@switch:~$ sudo systemctl status ntp
[sudo] password for cumulus:
ntp.service - LSB: Start NTP daemon
   Loaded: loaded (/etc/init.d/ntp; bad; vendor preset: enabled)
   Active: active (running) since Fri 2018-06-01 13:49:11 EDT; 2
   weeks 6 days ago
     Docs: man:systemd-sysv-generator(8)
    CGroup: /system.slice/ntp.service
            2873 /usr/sbin/ntpd -p /var/run/ntpd.pid -g -c /var
            /lib/ntp/ntp.conf.dhcp -u 109:114
```

2. Restart `rsyslog` so log files are sent to the correct destination.

```
cumulus@switch:~$ sudo systemctl status ntp
```

3. Link the host node to the TS you configured above.  
In this code example, the IP address for the TS is `198.168.1.254`. Note: Run `ifconfig eth0` on the TS if you forgot to write down the address.

```
cumulus@switch:~$ netq config add server 198.168.1.254
```

This command updates the configuration in the `/etc/netq/netq.yml` file and enables the NetQ CLI.

4. Restart NetQ Agent.

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



If you see the following error, it means you haven't added the telemetry server or the server wasn't configured:

```
Error: Please specify IP address of DB server
```

5. Verify NetQ Agent can reach the TS.

```
cumulus@switch:~$ netq config show server
```

Server	Port	Vrf	Status
-----	-----	-----	-----
198.168.1.254	6379	mgmt	ok

## Configure the Agent to Use a VRF (Optional)

While optional, Cumulus strongly recommends that you configure NetQ Agents to communicate with the telemetry server only via a [VRF](#), including a [management VRF](#). To do so, 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@leaf01:~$ netq config add server 198.168.1.254 vrf mgmt
```

You then restart the agent as described in the previous section:

```
cumulus@leaf01:~$ netq config restart agent
```

## Configure the Agent to Communicate over a Specific Port (Optional)

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 config add server 198.168.1.254 port 7379
```



If you are using NetQ in high availability mode, you can only configure it on port 6379 or 26379.

## Enabling Docker for Container Environments

Before enabling Docker, you must first install Docker. The code examples used here were created on an Ubuntu 16.04 host.

To install and enable Docker:

1. Add the Docker repository key.

```
root@host:~# curl -fsSL https://download.docker.com/linux/ubuntu  
/gpg | sudo apt-key add -
```

2. Install the Docker repository.

```
root@host:~# echo "deb [arch=amd64] https://download.docker.com  
/linux/ubuntu $(lsb_release -cs) stable" | sudo tee /etc/apt  
/sources.list.d/docker.list
```

3. Update the package lists.

```
root@host:~# apt-get update
```

4. Install Docker on the Ubuntu host.

```
root@host:~# apt-get install -y docker-ce
```

5. Check that the Docker service is running on the Ubuntu 16.04 host.

```
root@host:~# systemctl status docker  
docker.service - Docker Application Container Engine  
Loaded: loaded (/lib/systemd/system/docker.service; enabled;  
vendor preset: enabled)  
Active: active (running) since Wed 2017-10-18 02:51:48 UTC;  
1min 42s ago  
Docs: https://docs.docker.com  
Main PID: 18661 (dockerd)  
CGroup: /system.slice/docker.service  
18661 /usr/bin/dockerd -H fd://
```

```
18666 docker-containerd -l unix:///var/run/docker
/libcontainerd/docker-containerd.sock --metrics-interval=0 --
start-timeout 2m --state-dir /var/run/docker/libcontainerd
/containerd --shim docker-containerd-shim --runtime docker-runc
```

6. **Optional:** Add the docker group to your user account to be able to run docker commands without using sudo.

```
user@host:~$ sudo adduser ${USER} docker
```



Adding groups to different users requires a logout and login to take effect.

7. Enable Docker by adding the following three lines to the `netq.yml` file on the container host. This command also sets how often to pull data from the container to every 15 seconds.

```
root@host:~# vi /etc/cts/netq/netq.yml

...
docker:
  enable: true
  poll_period: 15
...
```

## Set Up Security

When you set up and configured your Cumulus Linux switches, you likely configured a number of the security features available. Cumulus recommends the same security measures be followed for the Telemetry Server in the out-of-band-network. Refer to the [Securing Cumulus Linux white paper](#) for details.

Your Cumulus Linux switches have a number of ports open (refer to [Default Open Ports in Cumulus Linux](#) article).

# Configure Optional NetQ Capabilities

After you have installed the Telemetry Server and the NetQ Agents, you may want to configure some of the additional capabilities that NetQ offers. This topic describes how to install, setup, and configure these capabilities.

## Contents

This topic describes how to...

- [Activate Early Access Features \(see page 30\)](#)
- [Integrate NetQ with an Event Notification Application \(see page 31\)](#)
  - [NetQ Notifier \(see page 31\)](#)
    - [Log Message Format \(see page 32\)](#)
    - [Default Output \(see page 33\)](#)
  - [NetQ Notifier CLI Overview \(see page 33\)](#)
  - [Configure NetQ Notifier Logging \(see page 34\)](#)
  - [Configure PagerDuty Using NetQ Notifier CLI \(see page 36\)](#)
  - [Configure Slack Using NetQ Notifier CLI \(see page 38\)](#)
  - [Configure PagerDuty Manually \(see page 40\)](#)
  - [Configure Slack Manually \(see page 41\)](#)
  - [View Integrations \(see page 42\)](#)
  - [Export Notifications to ELK \(see page 42\)](#)
  - [Export Notifications to Splunk \(see page 44\)](#)
- [Create NetQ Notifier Filters \(see page 44\)](#)
  - [Build Filter Rules \(see page 46\)](#)
  - [Specify Output Location \(see page 48\)](#)
  - [Example: Create a Filter for Events from Selected Switches \(see page 48\)](#)
  - [Example: Verify NetQ Notifier Status \(see page 48\)](#)
  - [Example: Add Multiple Rules to a Filter \(see page 48\)](#)
  - [Example: Place One Filter Before or After Another Filter \(see page 49\)](#)
  - [Example: Create a Filter to Drop Notifications from a Given Interface \(see page 50\)](#)
  - [Example: Create a Filter to Send BGP Session State Notifications to Slack \(see page 51\)](#)
  - [Example: Create a Filter to Drop All Temperature-Related Event Notifications \(see page 52\)](#)
  - [Example: Create a Filter for License Validation Event Notifications \(see page 52\)](#)
- [Configure High Availability Mode \(see page 53\)](#)
  - [Configure HA Mode \(see page 53\)](#)
  - [Check Database Cluster Status \(see page 54\)](#)
  - [Restart HA Mode Services \(see page 55\)](#)

- [Change the Master Telemetry Server](#) (see page 55)
- [Replace a Replica with a New Server](#) (see page 56)
- [Reset the Database Cluster](#) (see page 57)
- [Troubleshoot HA Mode](#) (see page 58)
  - [Relevant Services and Configuration Files](#) (see page 58)
  - [One Replica Must Be Available Always](#) (see page 59)
- [Integrate with a Hardware Chassis](#) (see page 59)
- [Extending NetQ with Custom Services Using curl](#) (see page 59)

## Activate Early Access Features

NetQ contains [early access](#) features that provide advanced access to new functionality before it becomes generally available. The telemetry-based early access features are bundled into the `netq-apps` package (installed automatically with NetQ) ; there is no specific EA package like there typically is with Cumulus Linux.

You enable early access features by running the `netq config add` command. You disable the early access features by running the `netq config del` command.

For example, to activate a feature:

1. Log on to any switch or server running NetQ.
2. Activate the early access feature set.

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

3. Try out the features and provide feedback to your sales representative.  
Usage information is contained in the [Cumulus NetQ Telemetry User Guide](#).

The NetQ Image and Provisioning Management (IPM) application are installed automatically, but not enabled.

To enable the IPM application:

1. Log on to the NetQ Telemetry Server.

```
<machine-name>:~<username>$ ssh cumulus@<telemetry-server-name-or-ip-address>  
cumulus@ts:~$
```

2. Open required ports.
3. Enable and start the IPM application (named *tips-appliance*).

```
cumulus@ts:~$ sudo systemctl enable tips-appliance  
Created symlink from /etc/systemd/system/multi-user.target.wants  
/tips-appliance.service to /lib/systemd/system/tips-appliance.  
service.
```

```
cumulus@ts:~$ sudo systemctl start tips-appliance
```

Once the IPM application is running, the IPM Command Line Interface, *TIPCTL*, is available. TIPCTL is the user interface used to activate, configure, and monitor the IPM application and services.

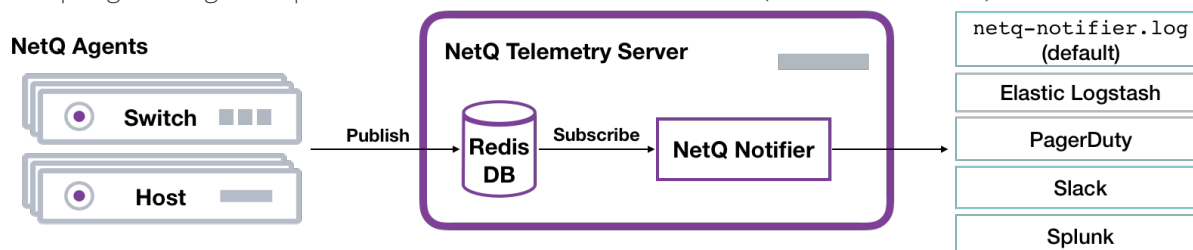
4. Try out the application and provide feedback to your sales representative.  
Usage information is contained in the [Cumulus NetQ Image and Provisioning Management User Guide](#).

## Integrate NetQ with an Event Notification Application

By default, alert messages are sent to `rsyslog`; however, other third-party tools may be configured to receive the alert messages. Once you have observed the events received in `rsyslog` for some period of time, you can more easily determine which events you want to direct to a given notification application and how you might assign the events to various event channels. You can integrate NetQ with a number of third-party notification applications, including PagerDuty, Slack, Elastic Logstash, and Splunk. You may integrate with one or more of these applications with or without filtering the events sent. To set up the integrations, you must configure the NetQ Notifier component on the Telemetry Server.

### NetQ Notifier

NetQ Notifier is responsible for delivering alerts to users through mediums such as Slack and syslog, informing users of network events. NetQ Notifier listens for events on the public subscription channel and writes them to `/var/log/netq-notifier.log`. As events occur, NetQ Notifier sends alert messages to `rsyslog`. NetQ Agents communicate directly with NetQ Notifier. Note that if you are running the TS in [HA mode \(see page 53\)](#), then NetQ Notifier only runs on the master instance of the TS and is responsible for accepting messages for publication from all of the TS instances (master and slaves).



Notifications can be generated for the following types of events:

- BGP session state
- Fan
- License
- Link
- LNV session state
- MLAG session state
- NTP
- OS
- Port
- PSU (power supply unit)

- Services
- Temperature

The NetQ Notifier is preconfigured to monitor for *all* events and publish *all* alert messages. No additional configuration is required; however, you can modify this default behavior to:

- Limit the events that are logged: NetQ Notifier sends out alerts based on the configured logging level. If you want to change the default logging level of *info* to increase or decrease the set of events that are logged, refer to [Configure NetQ Notifier Logging \(see page 34\)](#).
- Limit the events that are published: If you do not want to receive alerts for all events, you can limit the messages to publish by creating a NetQ Notifier filter. Refer to [Create NetQ Notifier Filters \(see page 44\)](#) for details.
- Use an alternate tool to view alerts: You can integrate NetQ Notifier with third-party event notification applications. Integration consists of setting a few parameters in the NetQ configuration file to identify the output, and then creating a filter to publish messages on that channel ([Create NetQ Notifier Filters \(see page 44\)](#)).

NetQ Notifier was installed as part of the TS virtual machine installation process and resides in the TS VM. You control the operation of NetQ Notifier using `systemd` commands (such as `systemctl stop|start netq-notifier`). You configure NetQ Notifier using the NetQ Notifier CLI, or by manually editing the `/etc/netq/netq.yml` configuration file.

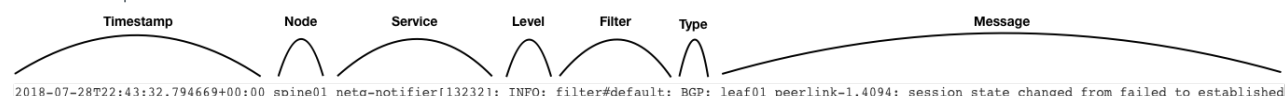
## Log Message Format

Messages have the following structure:

```
<timestamp> <node> <service>[PID]: <level>: filter#<name>: <type>: <message>
```

Element	Description
timestamp	Date and time event occurred in UTC format
node	Hostname of network node where event occurred
service [PID]	Service and Process IDentifier that generated the event
level	Logging level in which the given event is classified; <i>debug</i> , <i>error</i> , <i>info</i> , or <i>warning</i>
filter#<name>	Name of filter that captured the event; a name of <i>none</i> indicates no notification is sent (even if logged); a name of <i>default</i> indicates no filter is applied (all messages are sent)
type	Category of event; <i>BgpSession</i> , <i>Fan</i> , <i>License</i> , <i>Link</i> , <i>LnvSession</i> , <i>ClagSession</i> , <i>NTP</i> , <i>OS</i> , <i>Port</i> , <i>PSU</i> , <i>Services</i> , or <i>Temp</i>
message	Text description of event, including the node where the event occurred

For example:



```
2018-07-28T22:43:32.794669+00:00 spine01 netq-notifier[13232]: INFO: filter#default: BGP: leaf01 peerlink-1.4094: session state changed from failed to established
```



## Default Output

Using `rsyslog`, a standard mechanism to capture log files in Linux, NetQ Notifier sends alerts and events to the `/var/log/netq-notifier.log`.

## NetQ Notifier CLI Overview

The NetQ Notifier Command Line Interface (CLI) is used to filter and send notifications to third-party tools based on severity, service, event-type, and device. The CLI enables you to configure these items without editing the configuration file, reducing the risk of errors. The commands must be run from the Telemetry Server. All commands begin with `netq config ts`. You can use TAB completion or the help keyword to assist when needed.

Command	Description
add	Adds or modifies Notifier configuration, including filters, or sets master data node for DB cluster
del	Removes existing Notifier configuration
reset-cluster	Resets DB cluster to force into known state
restart	Restarts the Notifier daemon
show	Displays current DB server configuration or Notifier integrations
start	Starts the Notifier daemon
status	Display Notifier status
stop	Stop Notifier daemon

The command syntax is:

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

##Filters
netq config ts add notifier filter <text-filter-name> [before <text-
filter-name-anchor>] [after <text-filter-name-anchor>] [rule <text-
rule-key> <text-rule-value>] [output <text-integration-name-anchor>]
```

```
##Logging
netq config ts add notifier loglevel [debug|error|info|warning] [json]

##Management
netq config ts del notifier (slack|pagerduty) <text-integration-name-anchor>
netq config ts del notifier filter <text-filter-name-anchor>
netq config ts del notifier loglevel

netq config ts (start|stop|restart|status) notifier
netq config ts show notifier [<ip-server>|<text-server-name>|config]
[json]
netq config ts reset-cluster
```

The options are described in the following sections where they are used.

## Configure NetQ Notifier Logging

The logging level used by the NetQ Notifier determines what types of messages are published and what messages are suppressed. Five levels of logging are available, as shown in this table.

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

You can view the messages in the NetQ Notifier log, as shown here, or use the notification applications for easier viewing.

```
...
2018-10-01T18:46:00.337079+00:00 cumulus netq-notifier[5617]: INFO:
filter#default: Link: mlx-2700-03 swp3s1: state changed from down to
up
2018-10-01T18:46:01.455979+00:00 cumulus netq-notifier[5617]: INFO:
filter#default: Link: mlx-2700-03 hostbond3 (Local Node/s tor-1 and
Ports swp3s0 <==> Remote Node/s hosts-11 and Ports swp1) (master:
VlanA-1): state changed from down to up
```



```
2018-10-01T18:46:02.590245+00:00 cumulus netq-notifier[5617]: INFO:
filter#default: Link: mlx-2700-03 hostbond5 (Local Node/s tor-1 and
Ports swp3s2 <==> Remote Node/s hosts-13 and Ports swp1) (master:
VlanA-1): state changed from down to up
2018-10-01T18:46:29.585642+00:00 cumulus netq-notifier[5617]: INFO:
filter#default: Port: mlx-2700-03 swp3s2: port is now plugged
2018-10-01T18:46:30.782686+00:00 cumulus netq-notifier[5617]: INFO:
filter#default: Port: mlx-2700-03 swp3s3: port is now plugged
2018-10-01T18:25:45.527880+00:00 cumulus netq-notifier[5617]: INFO:
netq-notifier: Notifier processing messages because local REDIS is a
master.
2018-10-01T18:34:15.278120+00:00 cumulus netq-notifier[5617]:
WARNING: filter#default: Port: mlx-2700-03 swp17: port is now empty
2018-10-01T18:35:47.043010+00:00 cumulus netq-notifier[5617]: INFO:
filter#default: Port: mlx-2700-03 swp17: port is now plugged
2018-10-01T18:43:23.802855+00:00 cumulus netq-notifier[5617]:
WARNING: filter#default: Link: mlx-2700-03 swp3s3: state changed from
up to down
2018-10-01T18:43:25.195741+00:00 cumulus netq-notifier[5617]:
WARNING: filter#default: Link: mlx-2700-03 swp3s0: state changed from
up to down
2018-10-01T18:43:26.583027+00:00 cumulus netq-notifier[5617]:
WARNING: filter#default: Link: mlx-2700-03 swp3s2: state changed from
up to down
netq-notifier.log:2018-09-26T23:45:19.622756+00:00 cumulus netq-
notifier[2201]: CRITICAL: filter#default: Temp: act-omp03-lc101
temp95: state is absent
netq-notifier.log:2018-09-26T23:45:19.623445+00:00 cumulus netq-
notifier[2201]: CRITICAL: filter#default: Temp: act-omp03-lc101
temp96: state is absent
netq-notifier.log:2018-09-26T23:45:19.624140+00:00 cumulus netq-
notifier[2201]: CRITICAL: filter#default: Temp: act-omp03-lc101
temp97: state is absent
netq-notifier.log:2018-09-26T23:45:19.624806+00:00 cumulus netq-
notifier[2201]: CRITICAL: filter#default: Temp: act-omp03-lc101
temp98: state is absent
netq-notifier.log:2018-09-26T23:45:19.625486+00:00 cumulus netq-
notifier[2201]: CRITICAL: filter#default: Temp: act-omp03-lc101
temp99: state is absent
netq-notifier.log.1:2018-09-27T22:37:28.380181+00:00 cumulus netq-
notifier[844]: CRITICAL: filter#default: PSU: act-461054p-03 psu1:
state is bad
...
```

### Example: Configure debug-level logging

1. Set the logging level to *debug*.

```
cumulus@ts:~$ netq config ts add notifier loglevel debug
```

2. Restart the NetQ Agent.

```
cumulus@ts:~$ netq config ts restart notifier
Restarting netq-notifier... Success!
```

3. Verify connection to Telemetry Server.

```
cumulus@ts:~$ netq config ts show server
Server          Role      Master                               R
eplicas                                     Status      Last Changed
-----
192.168.1.254   master   192.168.1.254                       -
                                     ok
```

### Example: Configure warning-level logging

```
cumulus@switch:~$ netq config ts add notifier loglevel warning
cumulus@switch:~$ netq config ts restart notifier
cumulus@switch:~$ netq config ts show server
```

### Example: Disable NetQ Notifier Logging

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

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

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

To disable all logging:

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

## Configure PagerDuty Using NetQ Notifier CLI

NetQ Notifier sends notifications to PagerDuty as PagerDuty events.

For example:

<input type="checkbox"/>	Status	Urgency ▼	Title	Created ↕	Service	Assigned To
<input type="checkbox"/>	Resolved	Low	filter#default: NetQ Agent: spine-1: state changed from fresh to rotten <a href="#">SHOW DETAILS (1 resolved alert)</a> #10659	on Aug 31, 2017 at 3:08 PM	Puneet - Netq Notifier integration	--
<input type="checkbox"/>	Resolved	Low	filter#default: Service: noc-se clagd (vrf default): state changed from ok to warning <a href="#">SHOW DETAILS (1 resolved alert)</a> #10658	on Aug 31, 2017 at 3:08 PM	Puneet - Netq Notifier integration	--
<input type="checkbox"/>	Resolved	Low	filter#default: BGP: tor-2 uplink-1: session state changed from established to failed <a href="#">SHOW DETAILS (1 resolved alert)</a> #10657	on Aug 31, 2017 at 3:08 PM	Puneet - Netq Notifier integration	--
<input type="checkbox"/>	Resolved	Low	filter#default: BGP: torc-12 uplink-1: session state changed from established to failed <a href="#">SHOW DETAILS (1 resolved alert)</a> #10656	on Aug 31, 2017 at 3:08 PM	Puneet - Netq Notifier integration	--



If NetQ generates multiple notifications, on the order of 50/second (which can happen when a node reboots or when one peer in an MLAG pair disconnects), PagerDuty does not see all these notifications. You may see warnings in the netq-notifier.log file similar to this:

```
2017-09-20T20:39:48.222458+00:00 rdsq1 netq-notifier[1]: WARNING:
Notifier: notifier-pagerduty: Request failed with exception: Code: 429,
msg: {"status":"throttle exceeded","message":"Requests for this service
are arriving too quickly. Please retry later."}
```

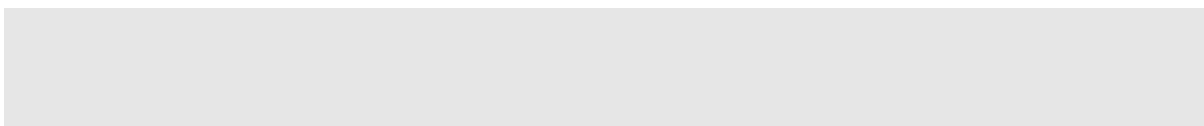
This is a known limitation in PagerDuty at this time.

To configure NetQ Notifier to send notifications to PagerDuty:

1. Configure the following options using the `netq config ts add notifier integration` command:

Option	Description
INTEGRATION_TYPE	The third-party notification system; use <i>pagerduty</i> in this case and provide a name for the integration.
api-access-key	The required API access key is also called the <a href="#">authorization token</a> .
api-integration-key	The <a href="#">integration</a> key is also called the <i>service_key</i> or <i>routing_key</i> . The default in an empty string ("").
severity	The log level to set, which can be one of <i>info</i> , <i>warning</i> , <i>error</i> , <i>critical</i> or <i>debug</i> . The severity defaults to <i>info</i> .

2. Restart the NetQ Notifier service  
For example:



```
cumulus@ts:~$ netq config ts add notifier integration pagerduty
pager-duty-test api-access-key 1234567890 api-integration-key
9876543210
cumulus@ts:~$ netq config ts restart notifier
```

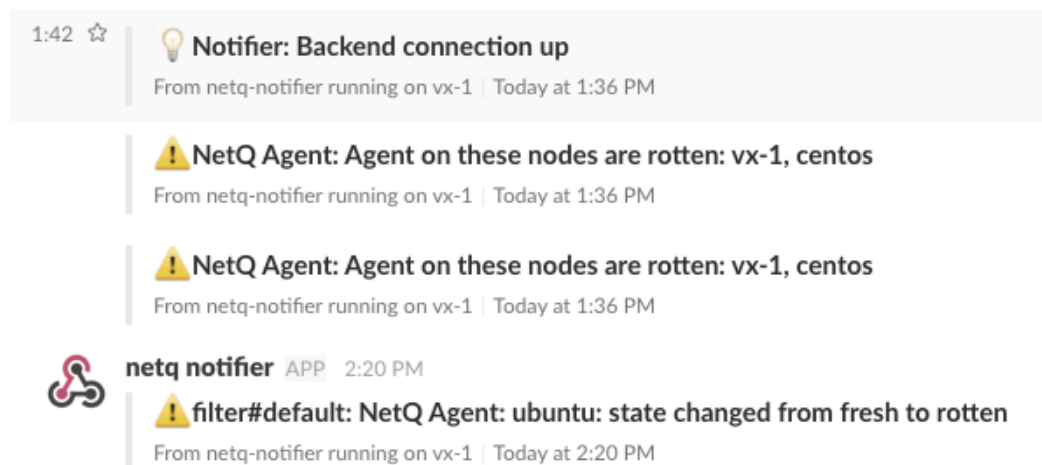
These commands create a new notifier-integration configuration in the `/etc/netq/netq.yml` file.

```
cumulus@ts:~$ cat /etc/netq/netq.yml
backend:
  port: 6379
  role: master
  server: 127.0.0.1
notifier-filters:
- name: default
  output:
  - ALL
  rule:
notifier-integrations:
- api_access_key: 1234567890
  api_integration_key: 9876543210
  name: pager-duty-test
  severity: info
  type: pagerduty
```

## Configure Slack Using NetQ Notifier CLI

NetQ Notifier sends notifications to Slack as incoming webhooks for a Slack channel you configure.

For example:



The screenshot shows a Slack interface with three notifications from the 'netq notifier' app. The first notification is a lightbulb icon with the text 'Notifier: Backend connection up' and 'From netq-notifier running on vx-1 | Today at 1:36 PM'. The second and third notifications are yellow warning triangle icons with the text 'NetQ Agent: Agent on these nodes are rotten: vx-1, centos' and 'From netq-notifier running on vx-1 | Today at 1:36 PM'. The fourth notification is a red warning triangle icon with the text 'filter#default: NetQ Agent: ubuntu: state changed from fresh to rotten' and 'From netq-notifier running on vx-1 | Today at 2:20 PM'.

To configure NetQ Notifier to send notifications to Slack:

1. If needed, create one or more Slack channels on which to receive the notifications.
  - a. Click + next to **Channels**.
  - b. Enter a name for the channel, and click **Create Channel**.

2. Create an incoming WebHook.
  - a. Select **Customize Slack** from the Slack dropdown menu.
  - b. Click **Workspaces** in the top right corner, and select the workspace where the channel is located.
  - c. Click **Add Applications**, and select **Incoming WebHooks**.
  - d. Click **Add Configuration** and enter the name of the channel you created (where you want to post notifications).
  - e. Click **Add Incoming WebHooks integration**.
  - f. Save WebHook URL in a text file for use in next step.

3. Configure the following options in the `netq config ts add notifier integration` command:

Option	Description
INTEGRATION_TYPE INTEGRATION_NAME	The third-party notification system; use <i>slack</i> in this case and provide a name for the integration.
WEBHOOK	Copy the WebHook URL from the text file OR in the desired channel, locate the initial message indicating the addition of the webhook, click <b>incoming-webhook</b> link, click <b>Settings</b> .  Example URL: <code>https://hooks.slack.com/services/text/moretext/evenmoretext</code>
severity	The log level to set, which can be one of <i>error</i> , <i>warning</i> , <i>info</i> , or <i>debug</i> . The severity defaults to <i>info</i> .
tag	Optional tag appended to the Slack notification to highlight particular channels or people. The tag value must be preceded by the @ sign. For example, <i>@ts-info</i> .

4. Restart the NetQ Notifier service.  
For example:

```
cumulus@ts:~$ netq config ts add notifier integration slack
slack-test webhook https://hooks.slack.com/services/text/moretext
/evenmoretext severity warning tag @ts-warning
cumulus@ts:~$ netq config ts restart notifier
```

These commands create a new notifier-integration configuration in the `/etc/netq/netq.yml` file.

```
cumulus@redis-1:~$ cat /etc/netq/netq.yml
backend:
  port: 6379
  role: master
  server: 127.0.0.1
```

```
notifier-filters:
- name: default
  output:
  - ALL
  rule:
notifier-integrations:
- name: slack-test
  severity: warning
  tag: '@ts-warning'
  type: slack
  webhook: https://hooks.slack.com/services/text/moretext
/evenmoretext
```

## Configure PagerDuty Manually

If you prefer to edit the `netq.yml` file directly, rather than use the NetQ Notifier CLI, you can do so.

To configure alerts and PagerDuty integrations on the NetQ Telemetry Server:

1. As a sudo user, open `/etc/netq/netq.yml` in a text editor.
2. Optionally, change the logging level, if you want a more restrictive level than *info* by adding the following two lines to the file:

```
...
netq-notifier:
  log_level: warning
...
```

3. Configure PagerDuty: Input the information for this integration into the `notifier-integrations` stanza, using the syntax shown in the example:
  - a. Add the `notifier-integrations` stanza.
  - b. Add a name, such as *notifier-pagerduty* or *netq-notifier*. Note: no spaces are allowed in the name. Use a dash instead if you want to separate words as shown here.
  - c. Add the type of *pagerduty*.
  - d. Add the message severity level, *debug*, *info*, *warning*, *error*, or *critical*.
  - e. Add the API access key (also called the *authorization token*) and the *integration* key (also called the *service\_key* or *routing\_key*).

```
notifier-integrations:
- name: notifier-pagerduty
  type: pagerduty
  severity: WARNING (does this need to be in all caps?)
  api_access_key: <API Key>
  api_integration_key: <API Integration Key>
```

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



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

```
cumulus@ts:~$ sudo systemctl restart netq-notifier
```

## Configure Slack Manually

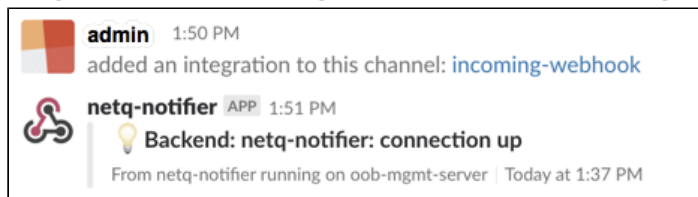
If you prefer to edit the `netq.yml` file directly, rather than using the NetQ Notifier CLI, you can do so.

To configure alerts and Slack integrations on the NetQ Telemetry Server:

1. As a sudo user, open `/etc/netq/netq.yml` in a text editor.
2. Optionally, change the logging level, if you want a more restrictive level than `info` by adding the following two lines to the file.

```
...
netq-notifier:
  log_level: warning
...
```

3. Configure Slack: Input the information for this integration into the notifier-integrations stanza, using the syntax shown in the example.
  - a. Add the the notifier-integrations stanza.
  - b. Add a name, such as `notifier-slack-channel-1` or `netq-notifier`. Note: no spaces are allowed in the name. Use a dash instead if you want to separate words as shown here.
  - c. Add the type of `slack`.
  - d. Add the webhook URL. To obtain this:
    - i. In the desired Slack channel, locate the initial message indicating the addition of the integration, click **incoming-webhook** link, click **Settings**.



- ii. The URL produced by Slack looks similar to the one shown here:

**Webhook URL**

```
https://hooks.slack.com/services/sometext/moretext/evenmoretext
```

- e. Add the message severity level, `debug`, `info`, `warning`, or `error`.
- f. Optionally add a tag, such as `@netqts-sys` or `@ts-info`.

```
notifier-integrations:
- name: notifier-slack-channel-1
```

```
type: slack
webhook: "https://hooks.slack.com/services/sometext
/moretext/evenmoretext"
severity: INFO
tag: "@netqts-sys"
```

4. When you are finished editing the file, save and close it.
5. Stop then start the NetQ Notifier daemon to apply the new configuration.

```
cumulus@netq-appliance:~$ sudo systemctl restart netq-notifier
```



If your webhook does not immediately send a message to your channel, look for errors in syntax. Check the log file located at `/var/log/netq-notifier.log`.

## View Integrations

You can view configured integrations using the `netq config ts show notifier integration` command. Include the `json` option to display JSON-formatted output.

For example:

```
cumulus@ts:~$ netq config ts show notifier integration
```

Integration Name	Attribute	Value
slack-test	type	slack
	tag	@ts-info
	webhook	https://hooks.slack.com/services/text/moretext/evenmoretext
	severity	info
pager-duty-test	api_integration_key	9876543210
	type	pagerduty
	severity	info
	api_access_key	1234567890

Filter Name	Attribute	Value
default	output	ALL
	rule	

## Export Notifications to ELK

NetQ Notifier integrates with ELK/Logstash using plugins to handle `rsyslog` inputs.

For example:

```
{
  "severity": 6, "pid": 8019, "program": "netq-notifier", "message": "INFO: filter#default: Service: ubuntu netq-notifier (vrf default): service restarted\n", "type": "syslog", "priority": 14, "logsource": "vx-1", "@timestamp": "2017-09-02T04:28:45.000Z", "@version": "1", "host": "192.168.50.110", "facility": 1, "severity_label": "Informational", "timestamp": "Sep  2 04:28:45", "facility_label": "user-level"}
{"severity": 6, "pid": 8019, "program": "netq-notifier", "message": "INFO: filter#default: Service: ubuntu rsyslog (vrf default): service restarted\n", "type": "syslog", "priority": 14, "logsource": "vx-1", "@timestamp": "2017-09-02T04:29:45.000Z", "@version": "1", "host": "192.168.50.110", "facility": 1, "severity_label": "Informational", "timestamp": "Sep  2 04:29:45", "facility_label": "user-level"}
{"severity": 6, "pid": 8019, "program": "netq-notifier", "message": "INFO: filter#default: Service: ubuntu netq-notifier (vrf default): service restarted\n", "type": "syslog", "priority": 14, "logsource": "vx-1", "@timestamp": "2017-09-02T04:30:00.000Z", "@version": "1", "host": "192.168.50.110", "facility": 1, "severity_label": "Informational", "timestamp": "Sep  2 04:30:00", "facility_label": "user-level"}
{"severity": 6, "pid": 8019, "program": "netq-notifier", "message": "INFO: filter#default: Service: ubuntu netq-notifier (vrf default): service restarted\n", "type": "syslog", "priority": 14, "logsource": "vx-1", "@timestamp": "2017-09-02T04:35:01.000Z", "@version": "1", "host": "192.168.50.110", "facility": 1, "severity_label": "Informational", "timestamp": "Sep  2 04:35:01", "facility_label": "user-level"}
{"severity": 6, "pid": 8019, "program": "netq-notifier", "message": "INFO: filter#default: Service: ubuntu netq (vrf default): service restarted\n", "type": "syslog", "priority": 14, "logsource": "vx-1", "@timestamp": "2017-09-02T04:38:18.000Z", "@version": "1", "host": "192.168.50.110", "facility": 1, "severity_label": "Informational", "timestamp": "Sep  2 04:38:18", "facility_label": "user-level"}
{"severity": 6, "pid": 8019, "program": "netq-notifier", "message": "INFO: filter#default: Service: ubuntu netq-agent (vrf default): service restarted\n", "type": "syslog", "priority": 14, "logsource": "vx-1", "@timestamp": "2017-09-02T04:38:18.000Z", "@version": "1", "host": "192.168.50.110", "facility": 1, "severity_label": "Informational", "timestamp": "Sep  2 04:38:18", "facility_label": "user-level"}
```

To configure NetQ Notifier to send data to ELK through Logstash:

1. On the host running the Telemetry Server and NetQ Notifier, configure the notifier to send the logs to a Logstash instance by adding the following to the logstash configuration file. In this example, the Logstash instance 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
```

2. Restart rsyslog.

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

3. On the server running the Logstash instance at the address specified earlier, create a file in /etc/logstash/conf.d/ called notifier\_logstash.conf. Then add the following definition, using the port you specified earlier.

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

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

4. Restart Logstash.

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

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

## Export Notifications to Splunk

NetQ integrates with Splunk using plugins to handle `rsyslog` inputs.

For example:

i	Time	Event
>	9/2/17 4:35:01.000 AM	<14>Sep 2 04:35:01 vx-1 netq-notifier[8019]: INFO: filter#default: Service: ubuntu netq-notifier (vrf default): service restarted host = vx-1 source = tcp:51415 sourcetype = syslog
>	9/2/17 4:30:00.000 AM	<14>Sep 2 04:30:00 vx-1 netq-notifier[8019]: INFO: filter#default: Service: ubuntu netq-notifier (vrf default): service restarted host = vx-1 source = tcp:51415 sourcetype = syslog
>	9/2/17 4:29:45.000 AM	<14>Sep 2 04:29:45 vx-1 netq-notifier[8019]: INFO: filter#default: Service: ubuntu rsyslog (vrf default): service restarted host = vx-1 source = tcp:51415 sourcetype = syslog
>	9/2/17 4:28:45.000 AM	<14>Sep 2 04:28:45 vx-1 netq-notifier[8019]: INFO: filter#default: Service: ubuntu netq-notifier (vrf default): service restarted host = vx-1 source = tcp:51415 sourcetype = syslog
>	9/2/17 4:28:30.000 AM	<14>Sep 2 04:28:30 vx-1 netq-notifier[8019]: INFO: filter#default: Service: ubuntu netq-notifier (vrf default): service restarted host = vx-1 source = tcp:51415 sourcetype = syslog

To configure NetQ Notifier to send data to Splunk for display:

1. On the host running the Telemetry Server and NetQ Notifier, configure the notifier to send the logs to a Splunk by adding the following to the Splunk configuration file.  
In this example, Splunk is installed 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
```

2. Restart `rsyslog`.

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

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

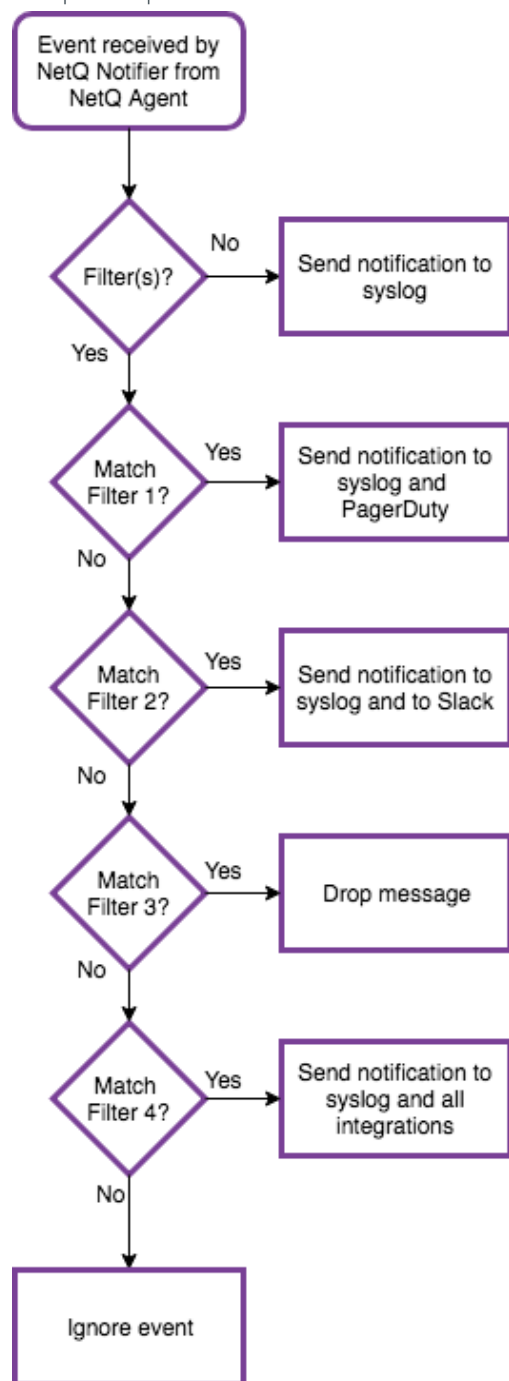
NetQ Notifier messages now appear in Splunk.

## Create NetQ Notifier Filters

You can limit or direct log messages sent by NetQ Notifier using filters. Filters are created based on rules you define. Each filter contains one or more rules. Each rule contains a single key-value pair. When a message matches the rule, it is sent to the indicated destination. The default rule is matches all messages. You can specify rules using entities such as hostnames or interface names in the regular expression, enabling you to filter messages specific to those hosts or interfaces. Additionally, you can send log messages to PagerDuty or a given Slack channel using the `output` keyword; you need to have already defined the PagerDuty or Slack configurations (as described earlier).

By default, filters are processed in the order they appear in the `netq.yml` configuration file (from top to bottom) until a match is found. This means that each event message is first evaluated by the first filter listed, and if it matches then it is processed, ignoring all other filters, and the system moves on to the next

event message received. If the event does not match the first filter, it is tested against the second filter, and if it matches then it is processed and the system moves on to the next event received. And so forth. Events that do not match any filter are ignored. This diagram shows an example with four defined filters with sample output results.



When you create a new filter, it is automatically added to the top of the list, before all others, so you may need to rearrange them to ensure you capture the events you want and drop the events you do not want. Optionally, you can use a *before* or *after* keyword to ensure one rule is processed before or after another.



Filter names may contain spaces, but must be enclosed with single quotes in commands. It is easier to use dashes in place of spaces or mixed case for better readability. For example, use `bgpSessionChanges` or `BGP-session-changes` or `BGPsessions`, instead of 'BGP Session Changes'. Filter names are also case sensitive.

You must restart the NetQ Notifier service after configuring filters so they can take effect; run the `netq config ts restart notifier` command.

## Build Filter Rules

Each rule is comprised of a single key-value pair. Creating multiple rules for a given filter can provide a very defined filter. There is a fixed set of valid rule keys. Values are entered as regular expressions and *vary according to your deployment*.

Rule Key	Rule Values
BgpSession	Regular expression identifying a hostname, peer name, ASN, or VRF Examples: <ul style="list-style-type: none"> <li>• hostname: server02, leaf*, exit01, spine0*</li> <li>• peer_name: server4, leaf*, exit02, spine0*</li> <li>• asn: 65020, 65012</li> <li>• vrf: mgmt, default</li> </ul>
ClagSession	Regular expression identifying a hostname or CLAG system MAC address Examples: <ul style="list-style-type: none"> <li>• hostname: server02, leaf*, exit01, spine0*</li> <li>• Clag_sysmac: 44:38:39:00:00:5C</li> </ul>
Fan	Regular expression identifying a hostname, sensor name, or sensor description Examples: <ul style="list-style-type: none"> <li>• hostname: server02, leaf*, exit01, spine0*</li> <li>• s_name: fan*</li> <li>• s_desc: 'fan 1, tray*'</li> </ul>
hostname	Regular expression identifying a hostname For example, hosts-0* or spine*.
License	Regular expression identifying a hostname or license name Examples: <ul style="list-style-type: none"> <li>• hostname: server02, leaf*, exit01, spine0*</li> <li>• name: Cumulus Linux, Ubuntu, CentOS</li> </ul>
Link	Regular expression identifying a hostname, interface name, or kind of link

Rule Key	Rule Values
	<p>Examples:</p> <ul style="list-style-type: none"> <li>• hostname: server02, leaf*, exit01, spine0*</li> <li>• ifname: eth0, swp*</li> <li>• kind: bond, bridge, eth, loopback, macvlan, swp, vlan, vrf, vxlan</li> </ul>
LnvSession	<p>Regular expression identifying a hostname or role</p> <p>Examples:</p> <ul style="list-style-type: none"> <li>• hostname: server02, leaf*, exit01, spine0*</li> </ul>
OS	<p>Regular expression identifying a hostname</p> <p>For example, hosts-0* or spine*.</p>
Port	<p>Regular expression identifying a hostname or interface name</p> <p>Examples:</p> <ul style="list-style-type: none"> <li>• hostname: server02, leaf*, exit01, spine0*</li> <li>• ifname : eth0, swp*</li> </ul>
PSU	<p>Regular expression identifying a hostname, sensor name</p> <p>Examples:</p> <ul style="list-style-type: none"> <li>• hostname: server02, leaf*, exit01, spine0*</li> <li>• s_name: psu*, psu2</li> </ul>
Services	<p>Regular expression identifying a hostname, service name, or VRF</p> <p>Examples:</p> <ul style="list-style-type: none"> <li>• hostname: server02, leaf*, exit01, spine0*</li> <li>• name: clagd, lldpd, ssh, ntp, netqd, net-agent</li> <li>• vrf: mgmt, default</li> </ul>
Temp	<p>Regular expression identifying a hostname, sensor name, sensor description</p> <p>Examples:</p> <ul style="list-style-type: none"> <li>• hostname: server02, leaf*, exit01, spine0*</li> <li>• s_name: psu1temp*, temp*</li> <li>• s_desc: 'board sensor near fan'</li> </ul>



Rules are case sensitive. Use Tab completion to view the command options syntax.

## Specify Output Location

The filter results may be output to a defined integration (integration name), to syslog (*default*), to all integrations and syslog (*all*), or not at all (*None*).

## Example: Create a Filter for Events from Selected Switches

In this example, we created a NetQ Notifier integration with an existing Slack channel called *slack-channel-leaf-alerts*. We then created a filter called *leaf-filter* for any notifications that mention switches named *leaf-0\**. The result is that any log messages mentioning one of those host names is filtered to the *slack-channel-leaf-alerts* channel.

```
cumulus@ts:~$ netq config ts add notifier integration slack slack-channel-leaf-alerts webhook https://hooks.slack.com/services/text/moretext/evenmoretext severity info
cumulus@ts:~$ netq config ts add notifier filter leaf-filter rule hostname leaf-0* output slack-channel-leaf-alerts
cumulus@ts:~$ netq config ts restart notifier
cumulus@ts:~$ netq config ts show notifier
```

Integration Name	Attribute	Value
slack-channel-leaf-alerts	type	slack
	webhook	https://hooks.slack.com/services/text/moretext/evenmoretext
	severity	info

Filter Name	Attribute	Value
leaf-filter	output	slack-channel-leaf-alerts
	rule	hostname: leaf-0*
default	output	ALL
	rule	

## Example: Verify NetQ Notifier Status

In this example, we are looking to see whether the NetQ Notifier is up and running. If it were not, we would want to restart it.

```
cumulus@ts:/ $ netq config ts status notifier
netq-notifier... Running
```

## Example: Add Multiple Rules to a Filter

In this example, we created a filter with three rules to drop notifications from a particular link.





```
cumulus@ts:~$ netq config ts add notifier filter leaf01-swpl-drop
rule type Link output None
cumulus@ts:~$ netq config ts add notifier filter leaf01-swpl-drop
rule hostname leaf01 output None
cumulus@ts:~$ netq config ts add notifier filter leaf01-swpl-drop
rule ifname swpl output None
cumulus@ts:~$ netq config ts restart notifier
cumulus@ts:~$ netq config ts show notifier
Integration Name      Attribute      Value
-----
None

Filter Name          Attribute      Value
-----
leaf01-swpl-drop     output rule   None
                                     type: Link
                                     hostname: leaf01
                                     ifname: swpl

default              output rule   ALL
```

## Example: Place One Filter Before or After Another Filter

In this example, we created a filter, *interface-filter*, that is processed after *bgp-filter* (an existing filter) and sent the notifications to a defined slack channel, *slack-channel-if-alerts*. Use the **before** keyword to place the new filter before the *bgp-filter*. Note that you must restart the Notifier for this configuration change to take effect. Verify the configuration using the `show` command.

```
cumulus@ts:~$ netq config ts add notifier filter interface-filter
after bgp-filter rule Port swp* output slack-channel-if-alerts
cumulus@ts:~$ netq config ts restart notifier
cumulus@ts:~$ netq config ts show notifier
Integration Name      Attribute      Value
-----
slack-channel-bgp-alerts  type          slack
                           webhook        https://hooks.slack.com
                           /services/text/moretext/evenmoretext
                           severity        info
slack-channel-if-alerts   type          slack
                           webhook        https://hooks.slack.com
                           /services/text/moretext/evenmoretext
                           severity        info

Filter Name          Attribute      Value
-----
```

bgp-filter	output	slack-channel-bgp-alerts
	rule	BgpSession: all
if-filter	output	slack-channel-if-alerts
	rule	Port: swp*
default	output	ALL
	rule	

Alternately, you may edit the `netq.yml` file directly adding the following integrations and filters:

```
cumulus@ts:~$ nano netq.yml
...
notifier-integrations:
  - name: slack-channel-bgp-alerts
    type: slack
    webhook: "https://hooks.slack.com/services/sometext/moretext
/evenmoretext"
    severity: INFO
  - name: slack-channel-if-alerts
    type: slack
    webhook: "https://hooks.slack.com/services/sometext/moretext
/evenmoretext"
    severity: INFO

notifier-filters:
  - name: bgp-filter
    rule:
      BgpSession: all
    output:
      - slack-channel-bgp-alerts
  - name: if-filter
    rule:
      Port: swp*
    output:
      - slack-channel-if-alerts
...
cumulus@ts:~$ netq config ts restart notifier
```

## Example: Create a Filter to Drop Notifications from a Given Interface

In this example, we created a filter, *ifswp21drop*, that drops notifications for events from interface *swp21*. Make sure to capitalize the Port rule option. Note that you must restart the Notifier for this configuration change to take effect.

```
cumulus@ts:~$ netq config ts add notifier filter ifswp21drop rule Port
swp21
cumulus@ts:~$ netq config ts add notifier filter ifswp21drop output
None
cumulus@ts:~$ netq config ts restart notifier
```

or

```
cumulus@ts:~$ nano netq.yml
...
notifier-filters:
  - name: ifswp21drop
    rule:
      Port: swp21
    output:
      - None
...
cumulus@ts:~$ netq config ts restart notifier
```

## Example: Create a Filter to Send BGP Session State Notifications to Slack

In this example, we created a filter, *BGPslackchannel*, to send BGP session state notifications to an existing Slack channel, *slack-channel-BGP*, and all other event notifications to a generic Slack channel, *slack-channel-catchall*. The Slack integration should be created before creating the filter. Note that you must restart the Notifier for this configuration change to take effect.

```
cumulus@ts:~$ netq config ts add notifier filter BGPslackchannel rule
BgpSession all
cumulus@ts:~$ netq config ts add notifier filter BGPslackchannel
output slack-channel-BGP
cumulus@ts:~$ netq config ts add notifier filter default output slack-
channel-catchall
cumulus@ts:~$ netq config ts restart notifier
```

or

```
notifier-filters:
  - name: BGPslackchannel
    rule:
      type: BgpSession
    output:
      - slack-channel-BGP
  - name: default
    rule:
    output:
      - slack-channel-catchall
```

## Example: Create a Filter to Drop All Temperature-Related Event Notifications

In this example, we created a filter, tempDrop, to drop all temperature related event notifications. Note: capitalization is important with these rules. Use Tab completion to view the command options syntax. Restart the Notifier for this configuration change to take effect.

```
cumulus@ts:~$ netq config ts add notifier filter tempDrop rule Temp
all
cumulus@ts:~$ netq config ts add notifier filter tempDrop output None
cumulus@ts:~$ netq config ts restart notifier
```

## Example: Create a Filter for License Validation Event Notifications

In this example, we created a filter, license-valid, to notify persons with access to a PagerDuty messages on the pager-duty-license channel when an invalid license is detected on *spine01*. Note: capitalization is important with these rules. Use Tab completion to view the command options syntax. Restart the Notifier for this configuration change to take effect.

```
cumulus@ts:~$ netq config ts add notifier pagerduty pager-duty-
license api-access-key 1234567890 api-integration-key 9876543210
cumulus@ts:~$ netq config ts add notifier filter license-valid rule
License spine01
cumulus@ts:~$ netq config ts add notifier filter license-valid output
pager-duty-license
cumulus@ts:~$ netq config ts restart notifier
```

```
cumulus@ts:~$ netq config ts show notifier integration
```

Integration Name	Attribute	Value
pager-duty-license	api_integration_key	9876543210
	type	pagerduty
	severity	info
	api_access_key	1234567890

Filter Name	Attribute	Value
license-expired	output	pager-duty-license
	rule	License: spine01
default	output	ALL
	rule	

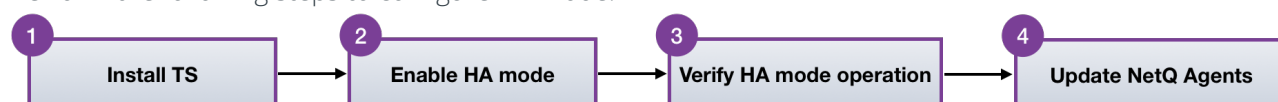
## Configure High Availability Mode

NetQ supports high availability — that is, the ability to continue functioning even in the absence of a single failure of a Telemetry Server node. To make the NetQ Telemetry Server highly available (*HA mode*), you need to run three instances of the Telemetry Server. Currently, exactly three instances are supported in HA mode. Of the three instances, one is considered the *master* and is writeable while the other two are read-only *replicas*. Each server instance is mapped to port 6379 on the host. A Redis *sentinel* on each Telemetry Server host monitors the health of the database cluster and decides which database is the current master. If the master becomes unavailable, the sentinel promotes one of the replicas to become the new master. Each sentinel runs on port 26379.

HA mode is optional.

### Configure HA Mode

Perform the following steps to configure HA Mode:



1. Install the Telemetry Server image on three separate physical hosts to form a database cluster.
  - a. Use the instructions for installing the TS here.



For proper operation of the HA mode you must specify the IP addresses for the Telemetry Servers. You cannot use DNS names.

- b. Note the IP address of each instance.
2. Enable HA Mode
 

These steps assume three Telemetry Servers, ts01 (the original one which was [already configured \(see page 7\)](#) as the Telemetry Server), ts02 and ts03, which are assigned IP addresses 10.0.0.5, 10.0.0.6 and 10.0.0.7, respectively. The servers are all assumed to be up and reachable and can communicate with each other.

On each Telemetry Server, specify the IP address of the master and both replicas, and then restart the netq-notifier service. Wait at least 30 seconds between each instance of the command.

```
cumulus@ts01:~$ netq config ts add server 10.0.0.5 10.0.0.6
10.0.0.7
cumulus@ts01:~$ sudo systemctl restart netq-notifier.service
```

```
cumulus@ts02:~$ netq config ts add server 10.0.0.5 10.0.0.6
10.0.0.7
cumulus@ts02:~$ sudo systemctl restart netq-notifier.service
```

```
cumulus@ts03:~$ netq config ts add server 10.0.0.5 10.0.0.6
10.0.0.7
cumulus@ts03:~$ sudo systemctl restart netq-notifier.service
```

- Verify that HA mode is configured on the three Telemetry Servers.  
Each server should indicate that ts01 (using IP address 10.0.0.5) is the master.

```
cumulus@ts01:~$ netq config ts show server
Server      Role      Master      Replicas      Status      Last
Changed
-----
-----
10.0.0.5    master    10.0.0.5    10.0.0.7, 10.0.0.6    ok          55s
10.0.0.6    replica   10.0.0.5    -               ok          55s
10.0.0.7    replica   10.0.0.5    -               ok          55s
```

```
cumulus@ts02:~$ netq config ts show server
Server      Role      Master      Replicas      Status      Last
Changed
-----
-----
10.0.0.5    master    10.0.0.5    10.0.0.7, 10.0.0.6    ok          55s
10.0.0.6    replica   10.0.0.5    -               ok          55s
10.0.0.7    replica   10.0.0.5    -               ok          55s
```

```
cumulus@ts03:~$ netq config ts show server
Server      Role      Master      Replicas      Status      Last
Changed
-----
-----
10.0.0.5    master    10.0.0.5    10.0.0.7, 10.0.0.6    ok          55s
10.0.0.6    replica   10.0.0.5    -               ok          55s
10.0.0.7    replica   10.0.0.5    -               ok          55s
```

- Update the NetQ Agent on each switch and server node you are monitoring to point to the HA cluster, and restart the NetQ Agent.

```
cumulus@switch:~$ netq config add server 10.0.0.5 10.0.0.6
10.0.0.7
Restarting netqd... Success!
cumulus@switch:~$ netq config restart agent
Restarting netq-agent... Success!
```

You could automate this step to update all of the agents on your monitored nodes.



## Check Database Cluster Status

Run the following `show` command from one of the Telemetry Servers.

```
cumulus@ts01:~$ netq config ts show server
```

Optionally, you can view detailed output for a specific server in the cluster by specifying that server's IP address.

```
cumulus@ts01:~$ netq config ts show server 10.0.0.7
```

## Restart HA Mode Services

You can restart the `netq-appliance` service using:

```
cumulus@ts01:~$ sudo systemctl restart netq-appliance.service
```

## Change the Master Telemetry Server

You can change which Telemetry Server you want to be the master simply by changing the order in which you specify them with the `netq config ts add server` command. You need to run the following command on each Telemetry Server, waiting at least 30 seconds in between updating the configuration on each server.

For example, notice that the Telemetry Server `ts01` is the master in the following configuration:

```
cumulus@ts01:~$ netq config ts show server
```

Server	Role	Master	Replicas	Status	Last
Changed					
-----	-----	-----	-----	-----	
10.0.0.5	master	10.0.0.5	10.0.0.6, 10.0.0.7	ok	50s
10.0.0.6	replica	10.0.0.5	-	ok	50s
10.0.0.7	replica	10.0.0.5	-	ok	50s

To make the first replica the new master, run the following command on each Telemetry Server (you don't need to change anything on the switch and server nodes):

```
cumulus@ts01:~$ netq config ts add server 10.0.0.6 10.0.0.5 10.0.0.7
```

```
cumulus@ts02:~$ netq config ts add server 10.0.0.6 10.0.0.5 10.0.0.7
```

```
cumulus@ts03:~$ netq config ts add server 10.0.0.6 10.0.0.5 10.0.0.7
```

Verify that *ts02* is the new master:

```
cumulus@ts01:~$ netq config ts show server
```

Server Changed	Role	Master	Replicas	Status	Last
10.0.0.5	replica	10.0.0.6	-	ok	28s
10.0.0.6	master	10.0.0.6	10.0.0.7, 10.0.0.5	ok	28s
10.0.0.7	replica	10.0.0.6	-	ok	28s

## Replace a Replica with a New Server

If you need to replace a Telemetry Server with a different physical system, follow the instructions include in this topic.



Do not try and replace the master server. You can only replace a replica. If you need to replace the master, make it a replica first, as described above.

For illustrative purposes, we are using the following cluster of Telemetry Servers, where 10.0.0.5 (*ts01*) is the master, while 10.0.0.6 (*ts02*) and 10.0.0.7 (*ts03*) are the replicas.

```
cumulus@ts01:~$ netq config ts show server
```

Server Changed	Role	Master	Replicas	Status	Last
10.0.0.5	master	10.0.0.5	10.0.0.6, 10.0.0.7	ok	14m:10s
10.0.0.6	replica	10.0.0.5	-	ok	14m:10s
10.0.0.7	replica	10.0.0.5	-	ok	14m:10s

In our example, you want to replace *ts03* with *ts04* (10.0.0.8):

1. Bring up the new Telemetry Server (*ts04*) and make sure the connectivity is okay.
2. Log in to the Telemetry Server you are replacing (*ts03*) and power it down:

```
cumulus@ts03:~$ sudo shutdown
```

3. Execute the following NetQ command on every Telemetry Server to create a cluster with the new Telemetry Server. Wait at least 30 seconds between each instance of the command.





```
cumulus@ts01:~$ netq config ts add server 10.0.0.5 10.0.0.6  
10.0.0.8
```

```
cumulus@ts02:~$ netq config ts add server 10.0.0.5 10.0.0.6  
10.0.0.8
```

```
cumulus@ts04:~$ netq config ts add server 10.0.0.5 10.0.0.6  
10.0.0.8
```

4. Verify the HA status on one of the Telemetry Servers. The status should be the same on all the three Telemetry Servers indicating that ts01 (10.0.0.5) is the master.

```
cumulus@ts01:~$ netq config ts show server
```

Server	Role	Master	Replicas	Status	Last
10.0.0.5	master	10.0.0.5	10.0.0.6, 10.0.0.8	ok	5m:10s
10.0.0.6	replica	10.0.0.5	-	ok	5m:10s
10.0.0.8	replica	10.0.0.5	-	ok	5m:10s

5. Update the agent on each switch and server node to point to the new HA cluster, and restart the NetQ Agent on each node:

```
cumulus@switch:~$ netq config add server 10.0.0.5 10.0.0.6  
10.0.0.8  
Restarting netqd... Success!  
cumulus@switch:~$ netq config restart agent  
Restarting netq-agent... Success!
```

## Reset the Database Cluster

You can force a reset of the Redis HA cluster using:

```
cumulus@netq-ts:~$ netq config ts reset-cluster
```

## Troubleshoot HA Mode

### *Relevant Services and Configuration Files*

The following `systemd` services are involved in HA mode:

- `cts-auth.service`: The Telemetry Server-side service that manages the configuration.
- `cts-auth.socket`: The Telemetry Server-side authorization shim socket for the service console.
- `cts-backup.service`: Runs a cron job to back up the Redis database.
- `cts-backup.timer`: The timer for the backup service, with a minimum interval of 5 minutes.
- `netqd.service`: The service for the Telemetry Server CLI for use locally on the server.
- `netq-appliance.service`: Starts and stops all Telemetry Server services **except** for the `ts-gui` service.
- `netq-gui.service`: Starts and stops Telemetry Server `ts-gui` service.
- `netq-influxdb.service`: The service that manages the HA mode InfluxDB.
- `netq-notifier.service`: Starts and stops the NetQ Notifier service.

The following configuration files are in the `/etc/cts/run/redis` directory:

- `redis_6379.conf`: Contains the runtime Redis database configuration and state.
- `snt1.conf`: Contains the runtime Redis sentinel configuration and state.

The following log file is in the `/var/log` directory:

- `netqd.log`: The logs associated with running the NetQ CLI locally on the machine.

The NetQ Notifier log is:

- `/var/log/netq-notifier.log`

Logging configurations are in:

- `/etc/rsyslog.d`
- `/etc/logrotate.d`

The following log files are in the `/var/log/cts` directory:

- `cts-backup.log`
- `cts-docker-compose.log`
- `cts-dockerd.log`
- `cts-redis.log`
- `cts-sentinel.log`

The `/etc/cts/environment` file sets key environment variables that control the behavior of the NetQ Telemetry Server.

- `REDIS_MEMORY_PCT`: Setting this variable to a value between 10 and 90 allocates that specified percent of the VM's memory to REDIS. The default value is 60. You can check the current allocation with the following command:

```
vagrant@netq-1:~$ /usr/sbin/netq-adjust-mem --show
Current maxmemory 2442384998 is 0.60 of 4070641664 available.
```

For more information about the log files, see the [Investigate NetQ Issues](#) topic.

### *One Replica Must Be Available Always*

While HA mode is enabled, if both the replica servers go down, the master database stops accepting writes. This causes the NetQ agents to go into a rotten state.

This serves to avoid having multiple masters in a split-brain condition. Please refer to the section "Example 2: basic setup with three boxes" on the [Redis Sentinel](#) page for more details.

## Integrate with a Hardware Chassis

NetQ can run within a [Facebook Backpack chassis](#), [Cumulus Express CX-10256-S chassis](#) or [Edgecore OMP-800 chassis](#).

Keep the following issues in mind if you intend to use NetQ with a chassis:

- You must assign a unique hostname to every node that runs the NetQ Agent. By default, all the fabric cards in the chassis have the same hostname.
- The NetQ Agent must be installed on every line card.
- No information is returned about the ASIC when you run `netq show inventory asic`. This is a known issue.
- Since the chassis sensor information is shared, every line card and fabric card can report the same sensor data. By default, sensor data is disabled on a chassis to avoid this duplication. To enable sensor data on a line card, edit `/etc/netq/netq.yml` or `/etc/netq/config.d/user.yml` and set the `send_chassis_sensor_data` keyword to `true`, then restart the NetQ Agent with `netq config agent restart`. Configuring NetQ in this way prevents any duplication of data in the NetQ database.

```
cumulus@chassis-lc101:~$ sudo nano /etc/netq/netq.yml

...
netq-agent:
  send_chassis_sensor_data: true
...
```

## 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](#) 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.
yaml

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 config agent restart
```

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

```
cumulus@leaf01:~$ cat /var/run/netq-agent-running.json
{"commands": [{"service": "smond", "always": false, "period":
30, "callback": {}, "command": "/usr/sbin/smonctl -j", "key":
"smonctl-json"}, {"service": "misc", "always": false, "period":
30, "callback": {}, "command": "/usr/sbin/switchd -lic", "key":
"cl-license"}, {"service": "misc", "always": false, "period":
30, "callback": {}, "command": null, "key": "ports"},
{"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/mstpcctl showall json", "key": "mstpcctl-bridge-
json"}], "backend": {"server": "10.0.0.165"}}
```

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

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

# Deployment Appendices

This topic contains additional reference material that may be useful when deploying the Cumulus NetQ software.

## Contents

This topic describes...

- [Example NetQ Switch Configuration File \(see page 62\)](#)
- [Example Ansible Playbook to Install NetQ on All Cumulus Switches \(see page 68\)](#)

## Example NetQ Switch Configuration File

The following sample `netq.yml` file is located in `/etc/netq/` on the NetQ Telemetry Server. Note that `netq.yml` looks different on a switch or host monitored by NetQ; for example, the backend server IP address and port would be uncommented and listed.



Editing `/etc/netq/config.d` to configure NetQ Notifier or putting other YML files in the `/etc/netq` directory overrides the configuration in `/etc/netq/netq.yml`.

Example `/etc/netq/netq.yml` configuration file

```
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:
#  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 Filter Configuration
##
## NetQ Notifier sends notifications to integrations(syslog, slack or
pagerduty)
## based on the events that are happening across the network.
## Notifications are generated based on the filters that have been
specified in
## "notifier-filters". NetQ Agents generate an event when something
interesting
## happens on the host (switch or server) its running on. The
Notifier is always
## listening for these events and once it receives an event, it makes
it go
## through a set of filters.
##
## A filter has 3 stages:
## a) Rule: Defines a set of conditions against which an incoming
event is
## matched. Input to this stage is an incoming event and the event is
sent to
## the next stage if there is a match. If there is a match, the event
## information is passed to the action stage. The rule is a
dictionary of
## key-value pairs, where the "key" is an item associated with the
event and
## "value" is the value of that item,
## e.g. type: Link
##       hostname: leaf-01
##       ifname: swpl
## The Default rule, if none is specified or if it is empty, is to
always assume
## a match.
```

```

## Notifier-filter rules are matched sequentially and we stop only
when a match
## is found. You can make the notifier continue matching filters even
if a match
## is found, by adding "terminate_on_match: False" to the filter.
Values
## specified in the rule are matched with those received in a event
using python
## regular expressions https://docs.python.org/2/library/re.html
## We can also match for message severity and print messages only if
it is above
## the given severity. Message severity levels are: INFO, WARNING,
ERROR and
## CRITICAL in ascending order.
##
## b) Action: action to perform if the "rule" is matched. The action
stage
## take the event provided by the "rule" stage and generates a message
## dictionary with a message and its severity. Multiple actions can be
## prescribed in the "action" list. "action" is typically a python
function that
## is provided with the tool or a custom one written by the user. If
no action
## is provided, we default to a generic handler which looks at the
event and
## based on the event runs the relevant notification function.
##
## c) output: This stage takes the message dictionary provided by the
action
## stage and sends the message and severity to the right integration
to display
## the message. If the output is None the message is not sent to any
integration
## or syslog. If output is empty, the message is sent only to syslog.
Else the
## message is sent to the list of integrations specified in the
output list and
## syslog. If ALL is specified, the message is sent to all
integrations.
## Integrations are defined in notifier-integrations.
##
## The config file comes with the following default filter:
##
## notifier-filters:
## - name: default
##   rule:
##   output:
##     - ALL
##
## which is an empty rule, empty action and output to all. This
defaults to

```



```
## match all rules and then perform the default action which is to
run the
## generic handler mentioned in the action stage above.
##
## NetQ Integration Configuration
##
## The integrations refer to the external tool where you would like
to receive
## the notification. An integration is added as a list element to
## "notifier-integrations". Each integration must have a "name" and
"type".
## Severity is optional and lets you send messages above that level
to the
## integration. Allowed values are: INFO, WARNING, ERROR, CRITICAL in
increasing
## order. Currently allowed "type" are "slack" and "pagerduty". You
can define
## multiple slack or PD integrations.
##
##For Slack integration, along with a name and "type: slack", you
also need to
## also provide the Incoming Webhook of the channel. The webhook URL
for your
## channel can be found or created in Slack at:
##   Apps -> Custom Integrations -> Incoming Webhooks.
## Tags are optional and are strings that are attached to the end of
the
## notification message.
## E.g.
# notifier-integrations:
# - name: notifier-slack-channel-1
#   type: slack
#   webhook: "https://<slack-webhook1>"
#   severity: INFO
#   tag: "@slack-tag1"
##
## For pagerDuty, along with name and "type: pagerduty", you also
need to
## provide the "api_access_key" and "api_integration_key" from
Pagerduty.
## A unique API Access Key which can be created on your PagerDuty
website at:
## Configuration -> API Access -> Create New API Key
## An 'Integration Key' 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'.
## E.g. pagerduty integration along with slack
# notifier-integrations:
# - name: notifier-slack-channel-1
#   type: slack
#   webhook: "https://<slack-webhook1>"
```

```
# severity: INFO
# tag: "@slack-tag1"
# - name: notifier-pagerduty
# type: pagerduty
# severity: WARNING
# api_access_key: <API Key>
# api_integration_key: <API Integration Key>
##
## Customizing Notifications
## Here are some examples on how to customize your notifications:
##
## a) Filter notifications to integrations (Slack or PD) based on
Severity,
## i.e., WARNING to PD, INFO to Slack
# notifier-integrations:
# - name: notifier-slack-channel-1
# type: slack
# webhook: "https://<slack-webhook1"
# severity: INFO <==== Set the severity type here
# tag: "@slack-tag1"
# - name: notifier-pagerduty
# type: pagerduty
# severity: WARNING <==== Set the severity type here
# api_access_key: "<API Key>"
# api_integration_key: "<API Integration Key>"
#
##
## b) Drop all notifications coming from a switch/host say, leaf-01
# notifier-filters:
# - name: leaf-01 drop
# rule:
#   hostname: leaf-01
# output:
#   - None
# - name: default
# rule:
# output:
#   - ALL
##
## c) Drop all notifications coming from switches whose name starts
with leaf
# notifier-filters:
# - name: leaf drop
# rule:
#   hostname: "leaf-.*"
# output:
#   - None
# - name: default
# rule:
# output:
#   - ALL
##
```

```
## d) Drop all notifications coming from a particular link, e.g. leaf-01 swp1
# notifier-filters:
# - name: leaf-01 swp1 drop
#   rule:
#     type: Link
#     hostname: leaf-01
#     ifname: swp1
#   output:
#     - None
# - name: default
#   rule:
#   output:
#     - ALL
##
## e) Send BGP Session state notifications to particular slack channel
## (slack-channel-BGP), rest to another one (slack-channel-catchall)
# notifier-filters:
# - name: BGP slack channel
#   rule:
#     type: BgpSession
#   output:
#     - slack-channel-BGP
# - name: default
#   rule:
#   output:
#     - slack-channel-catchall
##
## f) Send BgpSession notifications based on severity to different
slack channels
# notifier-filters:
# - name: BGP severity slack channel
#   rule:
#     type: BgpSession
#     severity: WARNING
#   output:
#     - slack-channel-BGP-info
# - name: default
#   rule:
#   output:
#     - slack-channel-catchall
##
## g) Drop all temperature related alerts
# notifier-filters:
# - name: temp drop
#   rule:
#     type: Temp
#   output:
#     - None
# - name: default
#   rule:
#   output:
```

```
# - ALL
notifier-filters:
  - name: default
    rule:
      output:
        - ALL
```

## Example Ansible Playbook to Install NetQ on All Cumulus Switches

The following example installs NetQ on all of your Cumulus switches, including:

- Setting the Management IP address
- Verifying a sufficient Cumulus Linux release is installed
- Applying the latest NetQ software package from the Cumulus repository
- Setting the NetQ Agent to communicate over the management VRF
- Restarting `rsyslog`
- Restarting the NetQ Agent

This performs all of the required installation and configuration to get NetQ installed and running. It assumes a management VRF and NTP has been previously setup. You would need to expand this playbook to install optional capabilities.



To run the playbook in your environment, you must replace the `netq_server` IP address with your Telemetry Server IP address.

```
- hosts: all
  gather_facts: false
  become: true
  remote_user: cumulus
  vars:
    netq_server: "192.168.0.254"
  tasks:
    - name: Collect CL Version
      shell: grep DISTRIB_RELEASE /etc/lsb-release | cut -d "=" -f2
      register: cl_version
      changed_when: false

    - name: Assert Cumulus Version Supports NetQ
      assert:
        that: "{{cl_version.stdout | version_compare('3.3.2', '>=') }}"
        msg: "Cumulus Linux version must be 3.3.2 or later to support
NetQ. Version {{cl_version.stdout}} detected."

    - name: Add Cumulus Repo
      apt_repository:
```



```
    repo: deb http://apps3.cumulusnetworks.com/repos/deb
CumulusLinux-3 netq-latest
    state: present
    update_cache: true
tags:
  - netq_setup

- name: Install NetQ (from Repo)
  apt:
    name: cumulus-netq
    update_cache: false
tags:
  - netq_setup

- name: Add netq server IP (VRF)
  command: netq config add server {{ netq_server }} vrf mgmt
tags:
  - netq_setup

- name: Restart Rsyslog
  service:
    name: rsyslog
    state: restarted
tags:
  - netq_setup

- name: Restart NetQ Agent
  command: netq config restart agent
tags:
  - netq_setup
```