



Replication Overview

Costa Tsaousis edited this page on Sep 19, 2017 · 8 revisions

Each netdata is able to replicate/mirror its database to another netdata, by streaming collected metrics, in real-time to it. This is quite different to [data archiving to third party time-series databases](#).

When a netdata streams metrics to another netdata, the receiving one is able to perform everything a netdata performs:

- visualize them with a dashboard
- run health checks that trigger alarms and send alarm notifications
- archive metrics to a backend time-series database

The following configurations are supported:

netdata without a database or web API (headless collector)

Local netdata (`slave`), **without any database or alarms**, collects metrics and sends them to another netdata (`master`).

The user can take the full functionality of the `slave` netdata at <http://master.ip:19999/host/slave.hostname/>. Alarms for the `slave` are served by the `master`.

In this mode the `slave` is just a plain data collector. It runs with... 5MB of RAM (yes, you read correct), spawns all external plugins, but instead of maintaining a local database and accepting dashboard requests, it streams all metrics to the `master`.

The same `master` can collect data for any number of `slaves`.

database replication

Local netdata (`slave`), **with a local database (and possibly alarms)**, collects metrics and sends them to another netdata (`master`).

The user can use all the functions at both <http://slave.ip:19999/> and <http://master.ip:19999/host/slave.hostname/>.

The `slave` and the `master` may have different data retention policies for the same metrics.

Alarms for the `slave` are triggered by **both** the `slave` and the `master` (and actually each can have different alarms configurations or have alarms disabled).

netdata proxies

Local netdata (`slave`), with or without a database, collects metrics and sends them to another netdata (`proxy`), which may or may not maintain a database, which forwards them to another netdata (`master`).

Alarms for the slave can be triggered by any of the involved hosts that maintains a database.

Any number of daisy chaining netdata servers are supported, each with or without a database and with or without alarms for the `slave` metrics.

mix and match with backends

Pages 99

General

- [Home](#)
- [Why netdata?](#)
- [Installation](#)
- [Command Line Options](#)
- [Configuration](#)
- [Log Files](#)
- [Tracing Options](#)

Running Netdata

- [Performance](#)
- [Memory Requirements](#)
- [netdata Security](#)
- [netdata OOMScore](#)
- [netdata process priority](#)

Special Uses

- [netdata for IoT](#)
lower netdata resource utilization
- [high performance netdata](#)
netdata public on the internet

Notes on memory management

- [Memory deduplication](#)
half netdata memory requirements
- [netdata virtual memory size](#)

Database Replication and Mirroring

- [Replication Overview](#)
- [monitoring ephemeral nodes](#)
Use netdata to monitor auto-scaled cloud servers.
- [netdata proxies](#)
Streaming netdata metrics between netdata servers.

Backends

archiving netdata collected metrics to a time-series database

- [netdata-backends](#)
`graphite`, `opentsdb`, `kairosdb`, `influxdb`, `elasticsearch`, `blueflood`
- [netdata with prometheus](#)
- Walk Through: [netdata with prometheus and grafana](#)

All nodes that maintain a database can also send their data to a backend database. This allows quite complex setups. Example:

1. netdata `A`, `B` do not maintain a database and stream metrics to netdata `C` (live streaming functionality, i.e. this PR)
2. netdata `C` maintains a database for `A`, `B`, `C` and archives all metrics to `graphite` with 10 second detail (backends functionality)
3. netdata `C` also streams data for `A`, `B`, `C` to netdata `D`, which also collects data from `E`, `F` and `G` from another DMZ (live streaming functionality, i.e. this PR)
4. netdata `D` is just a proxy, without a database, that streams all data to a remote site at netdata `H`
5. netdata `H` maintains a database for `A`, `B`, `C`, `D`, `E`, `F`, `G`, `H` and sends all data to `opentsdb` with 5 seconds detail (backends functionality)
6. alarms are triggered by `H` for all hosts
7. users can use all the netdata that maintain a database to view metrics (i.e. at `H` all hosts can be viewed).

netdata.conf configuration

These are options that affect the operation of netdata in this area:

```
[global]
memory mode = none | ram | save | map
```

`[global].memory mode = none` disables the database at this host. This also disables health monitoring (there cannot be health monitoring without a database).

```
[web]
mode = none | single-threaded | multi-threaded
```

`[web].mode = none` disables the API (netdata will not listen to any ports). This also disables the registry (there cannot be a registry without an API).

```
[backend]
enabled = yes | no
type = graphite | opentsdb
destination = IP:PORT ...
update every = 10
```

`[backend]` configures data archiving to a backend (it archives all databases maintained on this host).

streaming configuration

A new file is introduced: `/etc/netdata/stream.conf`. This file holds streaming configuration for both the sending and the receiving netdata.

API keys are used to authorize the communication of a pair of sending-receiving netdata. Once the communication is authorized, the sending netdata can push metrics for any number of hosts.

You can generate an API key with the command `uuidgen`. API keys are just random GUIDs. You can use the same API key on all your netdata, or use a different API key for any pair of sending-receiving netdata.

options for the sending node

This is the section for the sending netdata. On the receiving node, `[stream].enabled` can be `no`. If it is `yes`, the receiving node will also stream the metrics to another node (i.e. it will be a `proxy`).

Health monitoring - Alarms

alarms and alarm notifications in netdata

- [Overview](#)
- [Reference](#)
reference for writing alarms
- [Examples](#)
simple how-to for writing alarms
- [Notifications Configuration](#)
 - [web notifications](#)
 - [emails](#)
 - [discordapp.com](#)
 - [irc](#)
 - [messagebird.com](#)
 - [pagerduty.com](#)
 - [pushbullet.com](#)
 - [pushover.net](#)
 - [slack.com](#)
 - [alerta.io](#)
 - [flock.com](#)
 - [telegram.org](#)
 - [twilio.com](#)
 - [kavenegar.com](#)
- [health API calls](#)
- [troubleshooting alarms](#)

Netdata Registry

- [mynetdata Menu Item](#)

Monitoring Info

- [Monitoring web servers](#)
The spectacles of a web server log file
- [monitoring ephemeral containers](#)
Use netdata to monitor **auto-scaled containers**.
- [monitoring systemd services](#)
- [monitoring cgroups](#)
Use netdata to monitor containers and virtual machines.
- [monitoring IPMI](#)
Use netdata to monitor **enterprise server hardware**
- [Monitoring disks](#)
- [Monitoring Go Applications](#)

Netdata Badges

- [Generating Badges](#)

Data Collection

- [Add more charts to netdata](#)
- [Internal Plugins](#)
- [External Plugins](#)
- [statsd](#)
netdata is a fully featured statsd server
- [Third Party Plugins](#)
netdata plugins distributed by third parties

```
[stream]
enabled = yes | no
destination = IP:PORT ...
api key = XXXXXXXXXXXX
```

This is an overview of how these options can be combined:

target	memory mode	web mode	stream enabled	backend	alarms	dashboard
headless collector	none	none	yes	only for data source = as collected	not possible	no
headless proxy	none	not none	yes	only for data source = as collected	not possible	no
proxy with db	not none	not none	yes	possible	possible	yes
central netdata	not none	not none	no	possible	possible	yes

options for the receiving node

stream.conf looks like this:

```
# replace API_KEY with your uuidgen generated GUID
[API_KEY]
enabled = yes
default history = 3600
default memory mode = save
health enabled by default = auto
allow from = *
```

You can add many such sections, one for each API key. The above are used as default values for all hosts pushed with this API key.

You can also add sections like this:

```
# replace MACHINE_GUID with the slave /var/lib/netdata/registry/netdata.public.unique.id
[MACHINE_GUID]
enabled = yes
history = 3600
memory mode = save
health enabled = yes
allow from = *
```

The above is the receiver configuration of a single host, at the receiver end. MACHINE_GUID is the unique id the netdata generating the metrics (i.e. the netdata that originally collects them /var/lib/netdata/registry/netdata.unique.id). So, metrics for netdata A that pass through any number of other netdata, will have the same MACHINE_GUID .

allow from

allow from settings are netdata simple patterns: string matches that use * as wildcard (any number of times) and a ! prefix for a negative match. So: allow from = 10.1.2.3 10.* will allow all IPs in 10.* except 10.1.2.3 . The order is important: left to right, the first positive or negative match is used.

allow from is available in netdata v1.9+

tracing

Binary Modules

- Apps Plugin
- fping Plugin

Python Modules

- How to write new module
- apache
- apache_cache
- beanstalkd
- bind_rndc
- couchdb
- cpuidle
- cpufreq
- dns_query_time
- dovecot
- elasticsearch
- exim
- fail2ban
- freeradius
- go_expvar
- haproxy
- hddtemp
- ipfs
- isc_dhcpd
- mdstat
- memcached
- mongodb
- mysql
- nginx
- nginx_plus
- nsd
- ntpd
- ovpn_status_log
- phpfpn
- postfix
- postgres
- powerdns
- rabbitmq
- redis
- retroshare
- sensors
- springboot
- squid
- smartd_log
- tomcat
- varnish
- web_log

Node.js Modules

- General Info - node.d
- named.node.js
- sma_webbox.node.js
- fronijs.node.js
- stiebeltron.node.js
- snmp.node.js

BASH Modules

- General Info - charts.d

Active BASH Modules

- ap.chart.sh
- hostapd access points
- apcupsd.chart.sh
- APC UPSes

When a `slave` is trying to push metrics to a `master` or `proxy`, it logs entries like these:

```
2017-02-25 01:57:44: netdata: ERROR: Failed to connect to '10.11.12.1', port '19999'
(errno 111, Connection refused)
2017-02-25 01:57:44: netdata: ERROR: STREAM costa-pc [send to 10.11.12.1:19999]: failed
to connect

2017-02-25 01:58:04: netdata: INFO : STREAM costa-pc [send to 10.11.12.1:19999]:
initializing communication...
2017-02-25 01:58:04: netdata: INFO : STREAM costa-pc [send to 10.11.12.1:19999]: waiting
response from remote netdata...
2017-02-25 01:58:14: netdata: INFO : STREAM costa-pc [send to 10.11.12.1:19999]:
established communication - sending metrics...
2017-02-25 01:58:14: netdata: ERROR: STREAM costa-pc [send]: discarding 1900 bytes of
metrics already in the buffer.
2017-02-25 01:58:14: netdata: INFO : STREAM costa-pc [send]: ready - sending metrics...
```

The receiving end (`proxy` or `master`) logs entries like these:

```
2017-02-25 01:58:04: netdata: INFO : STREAM [receive from [10.11.12.11]:33554]: new
client connection.
2017-02-25 01:58:04: netdata: INFO : STREAM costa-pc [10.11.12.11]:33554: receive thread
created (task id 7698)
2017-02-25 01:58:14: netdata: INFO : Host 'costa-pc' with guid '12345678-b5a6-11e6-8a50-
00508db7e9c9' initialized, os: linux, update every: 1, memory mode: ram, history entries:
3600, streaming: disabled, health: enabled, cache_dir: '/var/cache/netdata/12345678-b5a6-
11e6-8a50-00508db7e9c9', varlib_dir: '/var/lib/netdata/12345678-b5a6-11e6-8a50-
00508db7e9c9', health_log: '/var/lib/netdata/12345678-b5a6-11e6-8a50-
00508db7e9c9/health/health-log.db', alarms default handler:
'/usr/libexec/netdata/plugins.d/alarm-notify.sh', alarms default recipient: 'root'
2017-02-25 01:58:14: netdata: INFO : STREAM costa-pc [receive from [10.11.12.11]:33554]:
initializing communication...
2017-02-25 01:58:14: netdata: INFO : STREAM costa-pc [receive from [10.11.12.11]:33554]:
receiving metrics...
```

For netdata v1.9+, streaming can also be monitored via `access.log`.

Viewing remote host dashboards, using mirrored databases

On any receiving netdata, that maintains remote databases and has its web server enabled, `my-netdata` menu will include a list of the mirrored databases.

- [example.chart.sh](#)
Skeleton to copy and adapt
- [libreswan.chart.sh](#)
LibreSWAN IPSEC tunnels
- [nut.chart.sh](#)
NUT UPSes
- [opensips.chart.sh](#)
SIP Server

Obsolete BASH Modules

- [apache.chart.sh](#)
(see [python.d/apache](#))
- [cpufreq.chart.sh](#)
(see [python.d/cpufreq](#))
- [cpu-apps.chart.sh](#)
(see [by apps.plugin](#))
- [exim.chart.sh](#)
(see [python.d/exim](#))
- [hddtemp](#)
(see [python.d/hddtemp](#))
- [load-average.chart.sh](#)
(now an netdata internal plugin)
- [mem-apps.chart.sh](#)
(see [apps.plugin](#))
- [mysql.chart.sh](#)
(see [python.d/mysql](#))
- [nginx.chart.sh](#)
(see [python.d/nginx](#))
- [phpfpm.chart.sh](#)
(see [python.d/phpfpm](#))
- [postfix.chart.sh](#)
(see [python.d/postfix](#))
- [sensors.chart.sh](#)
(see [python.d/sensors](#) - but can be useful for reading RPi temperatures)
- [squid.chart.sh](#)
(see [python.d/squid](#))
- [tomcat.chart.sh](#)
(see [python.d/tomcat](#))

JAVA Modules

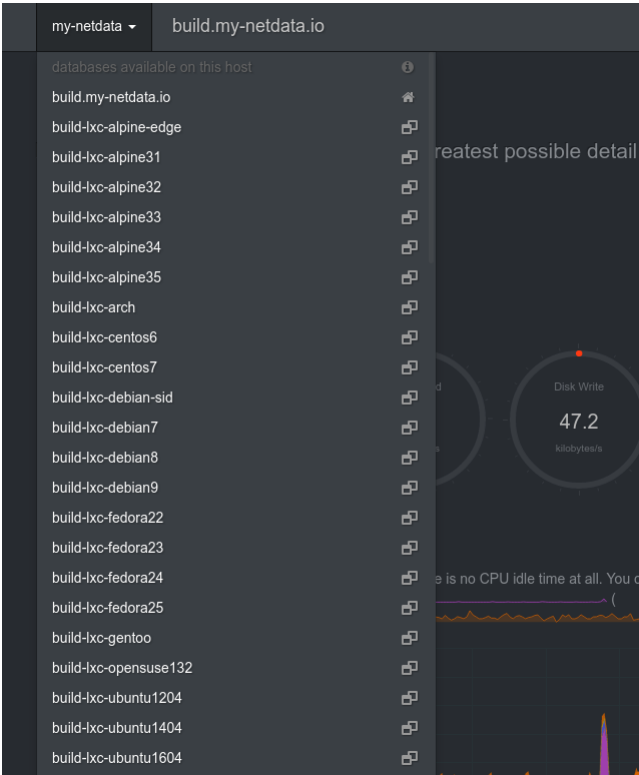
- [jmx](#)

API Documentation

- [REST API v1](#)
- [Obsolete REST API pre-v1](#)
- [Receiving netdata metrics from shell scripts](#)

Web Dashboards

- [Overview](#)
- [Custom Dashboards](#)
- [Custom Dashboard with Confluence](#)
- [Chart Libraries](#)
 - [Dygraph](#)
 - [EasyPieChart](#)
 - [Gauge.js](#)
 - [jQuery Sparkline](#)
 - [Peity](#)



Selecting any of these, the server will offer a dashboard using the mirrored metrics.

Learn how to create dashboards with charts from one or more netdata servers!

Running behind another web server

- [Running behind nginx](#)
- [Running behind apache](#)
- [Running behind lighttpd](#)
- [Running behind caddy](#)

Donations

- [Donations netdata has received](#)

Blog

- December, 2016
 - [Linux console tools, fail to report per process CPU usage properly](#)
- April, 2016
 - [netdata v1.1.0 released](#)
 - [You should install QoS on all your servers \(Linux QoS for humans\)](#)
 - [Monitor application bandwidth with Linux QoS \(Good to do it, anyway\)](#)
 - [Monitoring SYNPROXY \(Linux TCP Anti-DDoS\)](#)
- March, 2016
 - [Article: Introducing netdata \(the design principles of netdata\)](#)

[Other monitoring tools](#)

Clone this wiki locally

<https://github.com/firehol/>

Clone in Desktop

netdata

© Copyright 2016-2018, Costa Tsaousis costa@tsaousis.gr, released under [GPL v3+](#)

[New - stay in touch - follow netdata on twitter](#)