# m31271n/system-monitor-handbook:

A handbook about system montioring.

https://github.com/m31271n/system-monitor-handbook

system-monitor-handbook

文章介绍了两种现有的监控服务部署的基本思路。

* [Graphite 监控](https://github.com/m31271n/system-monitor-handbook/blob/master/Graphite.md)

标准 Graphite 由于性能低下，备受诟病。

* [TIGK 监控](https://github.com/m31271n/system-monitor-handbook/blob/master/TIGK.md)

TICK 堆栈由于被商业公司 InfluxData 控制，现在并不被社区看好，从 0.12 以后，开源版本的 InfluxDB 剔除了集群功能，拓展性和性能都非常成问题。在监控量很大时，笔者不再推荐使用。

* [Prometheus 监控](https://github.com/m31271n/system-monitor-handbook/blob/master/PROMETHEUS.md)

待评价。

* [Open-Falcon](https://github.com/m31271n/system-monitor-handbook/blob/master/OPEN_FALCON.md)

小米运维部开源的一款互联网企业级监控系统解决方案。经过了一些公司的检验。详情查看其[主页](http://open-falcon.org/)

# system-monitor-handbook/TIGK.md

at master ·

m31271n/system-monitor-handbook <https://github.com/m31271n/system-monitor-handbook/blob/master/TIGK.md>

TIGK 监控

**Table of Contents**

* [TIGK 监控](https://github.com/m31271n/system-monitor-handbook/blob/master/TIGK.md#tigk-%E7%9B%91%E6%8E%A7)
  + [工具选择](https://github.com/m31271n/system-monitor-handbook/blob/master/TIGK.md#%E5%B7%A5%E5%85%B7%E9%80%89%E6%8B%A9)
  + [数据流向](https://github.com/m31271n/system-monitor-handbook/blob/master/TIGK.md#%E6%95%B0%E6%8D%AE%E6%B5%81%E5%90%91)
  + [服务部署](https://github.com/m31271n/system-monitor-handbook/blob/master/TIGK.md#%E6%9C%8D%E5%8A%A1%E9%83%A8%E7%BD%B2)
    - [配置 InfluxDB](https://github.com/m31271n/system-monitor-handbook/blob/master/TIGK.md#%E9%85%8D%E7%BD%AE-influxdb)
      * [开启 HTTP 验证机制](https://github.com/m31271n/system-monitor-handbook/blob/master/TIGK.md#%E5%BC%80%E5%90%AF-http-%E9%AA%8C%E8%AF%81%E6%9C%BA%E5%88%B6)
      * [创建管理员用户](https://github.com/m31271n/system-monitor-handbook/blob/master/TIGK.md#%E5%88%9B%E5%BB%BA%E7%AE%A1%E7%90%86%E5%91%98%E7%94%A8%E6%88%B7)
      * [创建数据库和用户，并配置权限](https://github.com/m31271n/system-monitor-handbook/blob/master/TIGK.md#%E5%88%9B%E5%BB%BA%E6%95%B0%E6%8D%AE%E5%BA%93%E5%92%8C%E7%94%A8%E6%88%B7%E5%B9%B6%E9%85%8D%E7%BD%AE%E6%9D%83%E9%99%90)
    - [配置 Grafana](https://github.com/m31271n/system-monitor-handbook/blob/master/TIGK.md#%E9%85%8D%E7%BD%AE-grafana)
    - [安装及配置 Telegraf](https://github.com/m31271n/system-monitor-handbook/blob/master/TIGK.md#%E5%AE%89%E8%A3%85%E5%8F%8A%E9%85%8D%E7%BD%AE-telegraf)
    - [配置 Kapacitor](https://github.com/m31271n/system-monitor-handbook/blob/master/TIGK.md#%E9%85%8D%E7%BD%AE-kapacitor)
      * [接入 InfluxDB](https://github.com/m31271n/system-monitor-handbook/blob/master/TIGK.md#%E6%8E%A5%E5%85%A5-influxdb)
      * [书写 TickScripts](https://github.com/m31271n/system-monitor-handbook/blob/master/TIGK.md#%E4%B9%A6%E5%86%99-tickscripts)
      * [使用书写完成的 TickScripts 定义 Task](https://github.com/m31271n/system-monitor-handbook/blob/master/TIGK.md#%E4%BD%BF%E7%94%A8%E4%B9%A6%E5%86%99%E5%AE%8C%E6%88%90%E7%9A%84-tickscripts-%E5%AE%9A%E4%B9%89-task)
      * [开启定义完成的 Task](https://github.com/m31271n/system-monitor-handbook/blob/master/TIGK.md#%E5%BC%80%E5%90%AF%E5%AE%9A%E4%B9%89%E5%AE%8C%E6%88%90%E7%9A%84-task)

工具选择

标准 TICK 堆栈，包括

* 监控数据的采集：[Telegraf](https://github.com/influxdata/telegraf)
* 监控数据的存储：[InfluxDB](https://github.com/influxdata/influxdb)
* 监控数据的可视化：[Chronograf](https://github.com/influxdata/chronograf_)
* 基于已有监控数据的报警：[Kapacitor](https://github.com/influxdata/kapacitor)

但，推荐使用 [Grafana](https://github.com/grafana/grafana) 替换 Chronograf，原因有：

* Chronograf （成文时，其版本为 0.13.0）缺少必要的用户验证机制，用在内网尚可，暴露在公网欠妥。
* Chronograf 虽有商业公司支持，但目前看来，远没有 Grafana 功能强大。

替换之后，形成了 TIGK 堆栈。

数据流向

Telegraf 作为监控指标的采集器，部署在所有需要监控的机器上；采集到的数据被存储到 InfluxDB 中； InfluxDB 中的数据作为源数据，被 Grafana 和 Kapacitor 所使用，Grafana 做数据可视化，Kapacitor 做报警。

┌──────────┐ ┌──────────┐ ┌──────────┐

│ Telegraf │ │ Telegraf │ │ Telegraf │

└──────────┘ └──────────┘ └──────────┘

│ │ │

│ │ │

└──────────────────┼──────────────────┘

│

▼

┌───────────────┐

┌────┬────│ Influxdb │

│ │ └───────────────┘

│ │

│ │ ┌──────┐

│ │ ┌────────────────────┐ │ │

│ └─▶│ Graphana │──┘ ▼

│ └────────────────────┘ ┌─────┐

│ │Users│

│ ┌───────────────┐ └─────┘

└────────▶│ Kapacitor │────┐ ▲

└───────────────┘ │ │

└───────┘

服务部署

请阅读 docker 及 docker-compose 文档，获取必要知识储备后，再继续阅读。

克隆 [docker-tigk](https://github.com/m31271n/docker-tigk)，其中提供可使用 [docker-compose](https://github.com/docker/compose) 进行部署的相关文件。

创建并运行所需容器：

shell> docker-compose up -d

配置 InfluxDB

开启 HTTP 验证机制

默认情况下，InfluxDB 没有开启任何验证机制，这对于暴露在公网上的服务来说，非常危险。如何开启呢？ 将配置文件influxdb/conf/influxdb.conf 中 [http] 部分的 auth-enabled 设置为 true 即可。

项目中默认已将此设置开启，无需用户再去手动设置，可直接跳至下一步。

创建管理员用户

你可能会好奇：开启 HTTP 验证了，但还没有管理员用户，也就没法登录，这怎么办？

InfluxDB 的文档中写到：“如果开启了验证，并且还没有任何可以使用的用户，InfluxDB 并不会强制要求验证，而是会先要求你创建一个管理员用户。一旦拥有了一个管理员用户，InfluxDB 就会强制要求验证。”

使用 influx 连接到 InfluxDB 后，再使用 SQL 语句来创建管理员用户：

shell> docker exec -it influxdb influx

influxql> CREATE USER billy WITH PASSWORD 'MyNameIsBillyBrown' WITH ALL PRIVILEGES

或者使用 HTTP API 来创建：

shell> curl -POST http://localhost:8086/query --data-urlencode "q=CREATE USER billy WITH PASSWORD 'MyNameIsBillyBrown' WITH ALL PRIVILEGES"

创建数据库和用户，并配置权限

为每个项目创建独立的数据库，将不相关的数据隔离开，这将提高数据库的可维护性和安全性。

现在，假设有个项目叫 crap，为它创建数据库和读 / 写用户：

influxql> CREATE DATABASE crap

influxql> CREATE USER crap\_writer WITH PASSWORD 'writerpass'

influxql> CREATE USER crap\_reader WITH PASSWORD 'readerpass'

influxql> GRANT WRITE ON crap TO crap\_writer

influxql> GRANT READ ON crap TO crap\_reader

配置 Grafana

Grafana 默认监听在 3000 端口，根据服务所处网络的实际情况来设置防火墙，端口转发或是反向代理等。

然后使用 Grafana 默认的管理员用户及其对应密码 —— admin/admin 登录，将 InfluxDB 设置为数据源即可。

Dashboard 的配置方法参考 Grafana 文档。

通常的监控项有：

* CPU 使用率
* 内存使用率
* 硬盘使用率
* 硬盘 IOPS
* 硬盘吞吐量
* 网络 IO

其他监控项根据自身情况，自行添加。

安装及配置 Telegraf

在需要监控的机器上安装 Telegraf，根据 Telegraf 文档设置需要的监控项，将 Telegraf 获取到 metrics 输出到 InfluxDB 。

配置 Kapacitor

接入 InfluxDB

在 Kapacitor 配置文件中配置 InfluxDB 所使用的用户名和密码，之后，Kapacitor 将会自动订阅 InfluxDB 的数据。

书写 TickScripts

照着文档的示例写即可。

[docker-tigk](https://github.com/m31271n/docker-tigk) 预置了一些 TickScripts。

使用书写完成的 TickScripts 定义 Task

shell> docker exec -it kapacitor kapacitor define alert-cpu\_usage \

-type stream \

-tick /tickscripts/alert-cpu\_usage.tick \

-dbrp <db>.<rp>

开启定义完成的 Task

shell> docker exec -it kapacitor kapacitor enable alert-cpu\_usage

信任固然好，监控更重要。 —— 前苏联国家领导人列宁

我知道我部署技能炉火纯青，但我要弄个监控，让监控的报警在我狂妄时来抽醒我。 —— m31271n

# system-monitor-handbook/Graphite.md

at master · m31271n/system-monitor-handbook https://github.com/m31271n/system-monitor-handbook/blob/master/Graphite.md

Graphite 监控

**Table of Contents**

* [Graphite 监控](https://github.com/m31271n/system-monitor-handbook/blob/master/Graphite.md#graphite-%E7%9B%91%E6%8E%A7)
  + [Metric Gatherer](https://github.com/m31271n/system-monitor-handbook/blob/master/Graphite.md#metric-gatherer)
  + [Listener](https://github.com/m31271n/system-monitor-handbook/blob/master/Graphite.md#listener)
  + [Storage Database](https://github.com/m31271n/system-monitor-handbook/blob/master/Graphite.md#storage-database)
  + [Visualizer](https://github.com/m31271n/system-monitor-handbook/blob/master/Graphite.md#visualizer)
  + [Alert](https://github.com/m31271n/system-monitor-handbook/blob/master/Graphite.md#alert)

在标准的 Graphite 部署中，Carbon 将收集到 metrics 存储到 Whisper，然后 Graphite-web 对这些数据进行可视化处理。但是由于 Graphite 的标准组件的性能问题，产生了大量的替代组件。你可以根据自身情况进行组合。

这不是一篇教授如何配置 Graphite 监控的教程，而是帮助理解 Graphite 组件的说明。

Metric Gatherer

收集 metrics。

Graphite 自身不负责收集任何监控数据，它需要额外的 Metric Gatherer 将数据发送给它。

Graphite 并不包括 Metric Gatherer。然而，因为 Graphite 的协议很简单，同时受益于开源社区，有大把的 Metric Gatherer 可供使用。

官方有一个[列表](http://graphite.readthedocs.io/en/latest/tools.html#collection)列出了可用的 Metric Gatherer。

Listener

Listener 监听 metrics，并把它们写入数据库。

Graphite 中包含的标准 Listener 是 Carbon。在小规模部署时，Carbon 监听 metrics，然后将它们存储到 Whisper 数据库。 随着监控节点的增多，将会遇到 Carbon 的性能问题：

* Carbon 是用 Python 写的，由于 GIL 的原因，即使是多线程也不能增大单位时间内的 metric 处理量。 所以，单位时间内，Carbon 只能处理那么多的 metrics。当数据量超过了 Carbon 能处理的阈值，它就开始扔 metric 了。
* 上一条提到 Carbon 单位时间内处理的数据量有个阈值，但是这个阈值，没有可靠的测量的方法。
* Carbon 和 Whisper 之间没有持续打开的文件句柄，所以存储每个 metric 都需要完整的多步读写操作。

标准的 Graphite 中，解决的方案有：

* 让 Metric Gatherer 缓存一段时间内的 metric，之后再将它们发送到 Carbon
* 使用 Carbon Relay 和 Carbon Cache

但是，你还是得留神 Carbon 的阈值，因为一旦超过了阈值，Carbon 还是会丢数据。说实话，Carbon 用起来挺蛋疼的，还是找个替代品吧：

* [graphite-ng](https://github.com/graphite-ng) - 用 Go 重写的 Carbon，可以避免上述的性能问题
* [Reimann](http://riemann.io/)

Storage Database

负责数据的存储。

Graphite 中包含的标准 Storage Database 是 Whisper。

Whisper 是一个用来存储时间序列数据的数据库，基于 RPD (Round-Robin Database)。但是它像 Carbon 一样，也有性能问题：

* Python 写，性能差
* 由于设计，每个 metric 都需要一个文件，同时，因为磁盘扇区的限制（一个小于磁盘扇区大小的文件会占用一个扇区，多余的空间不会再被使用到）， 这造成磁盘的使用率低下
* 由于设计，Carbon 和 Whisper 之前存在大量 I/O 调用

标准的 Graphite 中，解决的方案有：

* 使用高性能的 SSD
* （其他的，不晓得了）

按照套路，该找替代品了：

* InfluxDB - InflubDB 是一个时间序列数据库，基于 LevelDB，用 Go 写的。它对于写操作进行了优化，同时没有一个 metric 对应一个文件的要求，很好地解决磁盘 IO 的问题。InfluxDB 支持 Carbon 所使用的协议，这让它可以成为 Whisper 的替代品。用了 InfluxDB 虽然棒，但是数据可视化的时候，就没法用 Graphite-web 了，得换用 Grafana 等支持 InfluxDB 数据格式的可视化工具。
* [Cyanite](https://github.com/pyr/cyanite) - Cyanite 致力于替换 Graphite 中的 Whisper，这意味着它可以和 Carbon、Graphite-web（需要额外依赖）一起使用。假如你只想换掉 Whisper，Cyanite 是个不错的选择。

Visualizer

一旦收集到了 metrics，就可以将它们可视化出来了。

在标准的 Graphite 中，这个工作是 Graphite-web 来做的。

Graphite-web 是一个使用 Django 框架写成的 Web 应用，渲染收集到 metrics，供用户查看。但是界面不怎么好看。

不过由于 Graphite API 的存在，有[大量的 Dashbaoard](http://graphite.readthedocs.io/en/latest/tools.html#visualization) 可供使用。

其中，[Grafana](http://grafana.org/) 应该是最好的了。

Alert

不可能一直盯着可视化工具去监控服务器，所以报警机制也是必要的。

Graphite 提供一个[列表](http://graphite.readthedocs.io/en/latest/tools.html#monitoring)，选一个适合你的就行了。

# Tools That Work With Graphite — Graphite 0.10.0 documentation

http://graphite.readthedocs.io/en/latest/tools.html#collection

Tools That Work With Graphite

Collection

[**Brubeck**](https://github.com/github/brubeck)

A statsd-compatible stats aggregator written in C.

[**Bucky**](http://pypi.python.org/pypi/bucky)

A small service implemented in Python for collecting and translating metrics for Graphite. It can currently collect metric data from CollectD daemons and from StatsD clients.

[**Carbonator Windows Service**](https://github.com/CryptonZylog/carbonator)

Simple lightweight Windows Service that collects Performance Counter metrics and sends them over to the Graphite server. Configured via .NET xml application configuration.

[**collectd**](http://collectd.org/)

A daemon which collects system performance statistics periodically and provides mechanisms to store the values in a variety of ways, including RRD. To send collectd metrics into carbon/graphite, use collectd’s [write-graphite](http://collectd.org/wiki/index.php/Plugin:Write_Graphite) plugin (available as of 5.1). Other options include:

* Jordan Sissel’s node [collectd-to-graphite](https://github.com/loggly/collectd-to-graphite) proxy
* Joe Miller’s perl [collectd-graphite](https://github.com/joemiller/collectd-graphite) plugin
* Gregory Szorc’s python [collectd-carbon](https://github.com/indygreg/collectd-carbon) plugin
* Paul J. Davis’s [Bucky](http://pypi.python.org/pypi/bucky) service

Graphite can also read directly from [collectd](http://collectd.org/)‘s RRD files. RRD files can simply be added toSTORAGE\_DIR/rrd (as long as directory names and files do not contain any . characters). For example, collectd’s host.name/load/load.rrd can be symlinked torrd/collectd/host\_name/load/load.rrd to graphcollectd.host\_name.load.load.{short,mid,long}term.

[**Collectl**](http://collectl.sourceforge.net/)

A collection tool for system metrics that can be run both interactively and as a daemon and has support for collecting from a broad set of subsystems. Collectl includes a Graphite interface which allows data to easily be fed to Graphite for storage.

[**Diamond**](https://diamond.readthedocs.io/en/latest/)

a Python daemon that collects system metrics and publishes them to Graphite. It is capable of collecting cpu, memory, network, I/O, load and disk metrics. Additionally, it features an API for implementing custom collectors for gathering metrics from almost any source.

[**Ganglia**](http://ganglia.info/)

A scalable distributed monitoring system for high-performance computing systems such as clusters and Grids. It collects system performance metrics and stores them in RRD, but now there is an [add-on](https://github.com/ganglia/ganglia_contrib/tree/master/graphite_integration/) that allows Ganglia to send metrics directly to Graphite. Further integration work is underway.

[**graphite-pollers**](https://github.com/phreakocious/graphite-pollers)

A collection of scripts that shovel data into Graphite including a multi-threaded SNMP poller for network interface IF-MIB statistics and another which pulls linux network stack data from files in /proc/net. Add to cron and go.

[**Graphite PowerShell Functions**](https://github.com/MattHodge/Graphite-PowerShell-Functions)

A group of functions that can be used to collect Windows Performance Counters and send them over to the Graphite server. The main function can be run as a Windows service, and everything is configurable via an XML file.

[**HoardD**](https://github.com/coredump/hoardd)

A Node.js app written in CoffeeScript to send data from servers to Graphite, much like collectd does, but aimed at being easier to expand and with less footprint. It comes by default with basic collectors plus Redis and MySQL metrics, and can be expanded with Javascript or CoffeeScript.

[**Host sFlow**](http://host-sflow.sourceforge.net/)

An open source implementation of the sFlow protocol ([http://www.sflow.org](http://www.sflow.org/)), exporting a standard set of host cpu, memory, disk and network I/O metrics. The sflow2graphite utility converts sFlow to Graphite’s plaintext protocol, allowing Graphite to receive sFlow metrics.

[**jmx2graphite**](https://github.com/logzio/jmx2graphite)

The easiest way to poll JMX metrics and write them to Graphite. This tool runs as a Docker container, polling your JMX every X seconds and writing the metrics to Graphite. Requires a minimum of configuration to get started.

[**jmxtrans**](https://github.com/jmxtrans/jmxtrans)

A powerful tool that performs JMX queries to collect metrics from Java applications. It is requires very little configuration and is capable of sending metric data to several backend applications, including Graphite.

[**Logster**](https://github.com/etsy/logster)

A utility for reading log files and generating metrics in Graphite or Ganglia. It is ideal for visualizing trends of events that are occurring in your application/system/error logs. For example, you might use logster to graph the number of occurrences of HTTP response code that appears in your web server logs.

[**metrics-sampler**](https://github.com/dimovelev/metrics-sampler)

A java program which regularly queries metrics from a configured set of inputs, selects and renames them using regular expressions and sends them to a configured set of outputs. It supports JMX and JDBC as inputs and Graphite as output out of the box.

[**Sensu**](http://sensuapp.org/)

A monitoring framework that can route metrics to Graphite. Servers subscribe to sets of checks, so getting metrics from a new server to Graphite is as simple as installing the Sensu client and subscribing.

[**snort2graphite**](https://github.com/gregvolk/snort2graphite)

Snort IDS/IPS can be configured to generate a rich set of metrics about network traffic. Presently there are more than 130 metrics available. Snort2graphite will pick up the most recent data from your snort.stats file and send all the metrics into Graphite.

[**SqlToGraphite**](https://github.com/perryofpeek/SqlToGraphite)

An agent for windows written in .net to collect metrics using plugins (WMI, SQL Server, Oracle) by polling an endpoint with a SQL query and pushing the results into graphite. It uses either a local or a centralised configuration over HTTP.

[**SSC Serv**](https://ssc-serv.com/)

A Windows service (agent) which periodically publishes system metrics, for example CPU, memory and disk usage. It can store data in Graphite using a naming schema that’s identical to that used by collectd.

Forwarding

[**Backstop**](https://github.com/obfuscurity/backstop)

A simple endpoint for submitting metrics to Graphite. It accepts JSON data via HTTP POST and proxies the data to one or more Carbon/Graphite listeners.

[**carbon-c-relay**](https://github.com/grobian/carbon-c-relay)

Enhanced C implementation of Carbon relay, aggregator and rewriter.

[**carbon-relay-ng**](https://github.com/graphite-ng/carbon-relay-ng)

Fast carbon relay+aggregator with admin interfaces for making changes online - production ready.

[**Evenflow**](https://github.com/github/evenflow)

A simple service for submitting sFlow datagrams to Graphite. It accepts sFlow datagrams from multiple network devices and proxies the data to a Carbon listener. Currently only Generic Interface Counters are supported. All other message types are discarded.

[**Grafsy**](https://github.com/leoleovich/grafsy)

Very light caching proxy for graphite metrics with additional features:

* Caching metrics in case of outage and sending them later
* Validation of metrics
* Aggregating of metrics, including SUM and AVG functions
* Much more

[**Graphite-Newrelic**](https://github.com/gingerlime/graphite-newrelic)

Get your graphite data into [New Relic](https://newrelic.com/platform) via a New Relic Platform plugin.

[**Graphite-relay**](https://github.com/markchadwick/graphite-relay)

A fast Graphite relay written in Scala with the Netty framework.

[**Graphios**](https://github.com/shawn-sterling/graphios)

A small Python daemon to send Nagios performance data (perfdata) to Graphite.

[**Graphout**](http://shamil.github.io/graphout)

A Node.js application that lets you forward Graphite based queries (using the render API) out to different external services. There are built in modules for Zabbix and CloudWatch. Custom modules are very easy to write.

[**Grockets**](https://github.com/disqus/grockets)

A node.js application which provides streaming JSON data over HTTP from Graphite.

[**Gruffalo**](https://github.com/outbrain/gruffalo)

An asynchronous Netty based graphite proxy, for large scale installations. It protects Graphite from the herds of clients by minimizing context switches and interrupts; by batching and aggregating metrics. Gruffalo also allows you to replicate metrics between Graphite installations for DR scenarios, for example.

[**Ledbetter**](https://github.com/github/ledbetter)

A simple script for gathering Nagios problem statistics and submitting them to Graphite. It focuses on summary (overall, servicegroup and hostgroup) statistics and writes them to the nagios.problems metrics namespace within Graphite.

[**pipe-to-graphite**](https://github.com/iFixit/pipe-to-graphite)

A small shell script that makes it easy to report the output of any other cli program to Graphite.

[**Polymur**](https://github.com/jamiealquiza/polymur)

A fast relay and HTTPS forwarder toolset written in Go.

[**statsd**](https://github.com/etsy/statsd)

A simple daemon for easy stats aggregation, developed by the folks at Etsy. A list of forks and alternative implementations can be found at <<http://joemiller.me/2011/09/21/list-of-statsd-server-implementations/>>

Visualization

[**Charcoal**](https://github.com/cebailey59/charcoal)

A simple Sinatra dashboarding frontend for Graphite or any other system status service which can generate images directly from a URL. Charcoal configuration is driven by a YAML config file.

[**Descartes**](https://github.com/obfuscurity/descartes)

A Sinatra-based dashboard that allows users to correlate multiple metrics in a single chart, review long-term trends across one or more charts, and to collaborate with other users through a combination of shared dashboards and rich layouts.

[**Dusk**](https://github.com/obfuscurity/dusk)

A simple dashboard for isolating “hotspots” across a fleet of systems. It incorporates horizon charts using Cubism.js to maximize data visualization in a constrained space.

[**Firefly**](https://github.com/Yelp/firefly)

A web application aimed at powerful, flexible time series graphing for web developers.

[**Gdash**](https://github.com/ripienaar/gdash.git)

A simple Graphite dashboard built using Twitters Bootstrap driven by a small DSL.

[**Giraffe**](http://kenhub.github.com/giraffe)

A Graphite real-time dashboard based on [Rickshaw](http://code.shutterstock.com/rickshaw) and requires no server backend. Inspired by[Gdash](https://github.com/ripienaar/gdash.git), [Tasseo](https://github.com/obfuscurity/tasseo) and [Graphene](http://jondot.github.com/graphene) it mixes features from all three into a slightly different animal.

[**Grafana**](http://grafana.org/)

A general purpose graphite dashboard replacement with feature rich graph editing and dashboard creation interface. It contains a unique Graphite target parser that enables easy metric and function editing. Fast client side rendering (even over large time ranges) using Flot with a multitude of display options (Multiple Y-axis, Bars, Lines, Points, smart Y-axis formats and much more). Click and drag selection rectangle to zoom in on any graph.

[**Graphene**](http://jondot.github.com/graphene)

A Graphite dashboard toolkit based on [D3.js](http://mbostock.github.com/d3) and [Backbone.js](http://documentcloud.github.com/backbone) which was made to offer a very aesthetic realtime dashboard. Graphene provides a solution capable of displaying thousands upon thousands of datapoints all updated in realtime.

[**graphite-dashboardcli**](https://github.com/blacked/graphite-dashboardcli)

A tool for manage graphite dashboards from command line:

* importExport dashboards fromto Graphite servers
* synchronize dashboards between multiple Graphite servers
* keep dashboards in YAML format

[**Graphite-Tattle**](https://github.com/wayfair/Graphite-Tattle)

A self-service dashboard frontend for Graphite and [Ganglia](http://ganglia.info/).

[**Graphiti**](https://github.com/paperlesspost/graphiti)

A powerful dashboard front end with a focus on ease of access, ease of recovery and ease of tweaking and manipulation.

[**Graphitoid**](https://market.android.com/details?id=com.tnc.android.graphite)

An Android app which allows one to browse and display Graphite graphs on an Android device.

[**graphitus**](https://github.com/ezbz/graphitus)

A client side dashboard for graphite built using bootstrap and underscore.js.

[**Graphsky**](https://github.com/hyves-org/graphsky)

A flexible and easy to configure PHP based dashboard. It uses JSON template files to build graphs and specify which graphs need to be displayed when, similar to Ganglia-web. Just like Ganglia, it uses a hierarchial structure: Environment/Cluster/Host/Metric to be able to display overview graphs and host-specific metrics. It communicates directly to the Graphite API to determine which Environments, Clusters, Hosts and Metrics are currently stored in Graphite.

[**Graph-Explorer**](http://vimeo.github.io/graph-explorer)

A graphite dashboard which uses plugins to add tags and metadata to metrics and a query language with lets you filter through them and compose/manipulate graphs on the fly. Also aims for high interactivity using [TimeseriesWidget](https://github.com/Dieterbe/timeserieswidget) and minimal hassle to set up and get running.

[**Graph-Index**](https://github.com/douban/graph-index)

An index of graphs for [Diamond](https://diamond.readthedocs.io/en/latest/).

[**Hubot**](https://github.com/github/hubot)

A Campfire bot written in Node.js and CoffeeScript. The related [hubot-scripts](https://github.com/github/hubot-scripts) project includes a Graphite script which supports searching and displaying saved graphs from the Composer directory in your Campfire rooms.

[**Leonardo**](https://github.com/PrFalken/leonardo)

A Graphite dashboard inspired by Gdash. It’s written in Python using the Flask framework. The interface is built with Bootstrap. The graphs and dashboards are configured through the YAML files.

[**Orion**](https://github.com/gree/Orion)

A powerful tool to create, view and manage dashboards for your Graphite data. It allows easy implementation of custom authentication to manage access to the dashboard.

[**Pencil**](https://github.com/fetep/pencil)

A monitoring frontend for graphite. It runs a webserver that dishes out pretty Graphite URLs in interesting and intuitive layouts.

[**Targets-io**](https://github.com/dmoll1974/targets-io)

A dashboard application for organizing, analyzing, benchmarking and reporting of performance test results. All performance test metrics are stored in Graphite and can be benchmarked between test runs, providing automated feedback on the performance of an application.

[**Tasseo**](https://github.com/obfuscurity/tasseo)

A lightweight, easily configurable, real-time dashboard for Graphite metrics.

[**Terphite**](https://github.com/benwtr/terphite)

Terminal tool for displaying Graphite metrics.

[**Tessera**](https://github.com/urbanairship/tessera)

A flexible front-end for creating dashboards with a wide variety of data presentations.

[**TimeseriesWidget**](https://github.com/Dieterbe/timeserieswidget)

Adds timeseries graphs to your webpages/dashboards using a simple api, focuses on high interactivity and modern features (realtime zooming, datapoint inspection, annotated events, etc). Supports Graphite, flot, rickshaw and anthracite.

Monitoring

[**Cabot**](https://github.com/arachnys/cabot)

A self-hosted monitoring and alerting server that watches Graphite metrics and can alert on them by phone, SMS, Hipchat or email. It is designed to be deployed to cloud or physical hardware in minutes and configured via web interface.

[**graphite-beacon**](https://github.com/klen/graphite-beacon)

A simple alerting application for Graphite. It asynchronous and sends notification alerts based on Graphite metrics. It hasn’t any dependencies except Tornado package. Very light and really very easy deployed.

[**graphite-to-zabbix**](https://github.com/blacked/graphite-to-zabbix)

A tool to make zabbix alerts based on Graphite data.

[**Icinga**](http://docs.icinga.org/icinga2/latest/doc/module/icinga2/chapter/icinga2-features#graphite-carbon-cache-writer)

Icinga 2 will directly write metrics to the defined Graphite Carbon daemon tcp socket if the graphite feature is enabled. This feature is a more simple integration compared to Icinga 1.x and Graphios.

[**Moira**](http://moira.readthedocs.io/)

An alerting system based on Graphite data. Moira is a real-time alerting tool, independent from graphite storage, custom expressions and extendable notification channels.

[**rearview**](http://github.com/livingsocial/rearview)

A real-time monitoring framework that sits on top of Graphite’s time series data. This allows users to create monitors that both visualize and alert on data as it streams from Graphite. The monitors themselves are simple Ruby scripts which run in a sandbox to provide additional security. Monitors are also configured with a crontab compatible time specification used by the scheduler. Alerts can be sent via email, pagerduty, or campfire.

[**Rocksteady**](http://code.google.com/p/rocksteady)

A system that ties together Graphite, [RabbitMQ](http://www.rabbitmq.com/), and [Esper](http://esper.codehaus.org/). Developed by AdMob (who was then bought by Google), this was released by Google as open source (<http://google-opensource.blogspot.com/2010/09/get-ready-to-rocksteady.html>).

[**Seyren**](https://github.com/scobal/seyren)

An alerting dashboard for Graphite.

[**Shinken**](http://www.shinken-monitoring.org/)

A system monitoring solution compatible with Nagios which emphasizes scalability, flexibility, and ease of setup. Shinken provides complete integration with Graphite for processing and display of performance data.

Storage Backend Alternates

If you wish to use a backend to graphite other than Whisper, there are some options available to you.

[**BigGraphite**](https://github.com/criteo/biggraphite)

A time-series database written in Python on top of Cassandra.

[**Ceres**](https://github.com/graphite-project/ceres)

An alternate storage backend provided by the Graphite Project. It it intended to be a distributable time-series database. It is currently in a pre-release status.

[**Cyanite**](http://cyanite.io/)

A highly available, elastic, and low-latency time-series storage wirtten on top of Cassandra

[**InfluxDB**](https://influxdb.com/)

A distributed time series database.

[**KairosDB**](http://kairosdb.github.io/)

A distributed time-series database written on top of Cassandra.

[**OpenTSDB**](http://opentsdb.net/)

A distributed time-series database written on top of HBase.

Other

[**bosun**](http://bosun.org/)

Time Series Alerting Framework. Can use Graphite as time series source.

[**Bryans-Graphite-Tools**](https://github.com/linkslice/graphite-tools)

A collection of miscellaneous scripts for pulling data from various devices, F5, Infoblox, Nutanix, etc.

[**buckytools**](https://github.com/jjneely/buckytools)

Go implementation of useful tools for dealing with Graphite’s Whisper DBs and Carbon hashing.

[**carbonate**](https://github.com/graphite-project/carbonate)

Utilities for managing graphite clusters.

[**go-carbon**](https://github.com/lomik/go-carbon)

Golang implementation of Graphite/Carbon server with classic architecture: Agent -> Cache -> Persister.

[**riemann**](http://riemann.io/)

A network event stream processing system, in Clojure. Can use Graphite as source of event stream.

[**Therry**](https://github.com/obfuscurity/therry)

A simple web service that caches Graphite metrics and exposes an endpoint for dumping or searching against them by substring.

[Next](http://graphite.readthedocs.io/en/latest/development.html)[Previous](http://graphite.readthedocs.io/en/latest/terminology.html)