grafana-pcp Documentation

Release 2.0.2

Performance Co-Pilot

Oct 12, 2020

Getting started

Performance Co-Pilot (PCP) provides a framework and services to support system-level performance monitoring and management. It presents a unifying abstraction for all of the performance data in a system, and many tools for interrogating, retrieving and processing that data.

CHAPTER 1

Features

- analysis of historical PCP metrics using pmseries query language
- analysis of real-time PCP metrics using pmwebapi live services
- enhanced Berkeley Packet Filter (eBPF) tracing using bpftrace scripts
- automatic rate conversation for counter metrics
- heatmap, table and flame graph [3] support
- auto completion of metric names [1,2], qualifier keys and values [1], and bpftrace probes, builtin variables and functions [3]
- display of semantics, units and help texts of metrics [2] and bpftrace builtins [3]
- legend templating support with `$metric`, `$metric0`, `$instance`, `$some_label`, `$some_dashboard_variable`
- container support [1,2]
- support for custom endpoint URL [1,2,3] and container [2] setting per query
- support for repeated panels
- sample dashboards for all data sources

[1] PCP Redis [2] PCP Vector [3] PCP bpftrace

CHAPTER 2

Getting started

2.1 Quickstart

2.1.1 Installation

```
$ sudo dnf install grafana-pcp
$ sudo systemctl restart grafana-server
$ sudo systemctl start pmproxy
```

After Grafana and grafana-pcp is installed, you can enable the plugin: Open the Grafana configuration, go to Plugins, select *Performance Co-Pilot* and click the *Enable* button.

2.1.2 Data Sources

Before using grafana-pcp, you need to configure the data sources. Open the Grafana configuration, go to Data Sources and add the *PCP Redis*, *PCP Vector* and/or *PCP bpftrace* datasources.

The only required configuration field for each data source is the URL to pmproxy. In most cases the default setting of `http://localhost:44322` can be used. All other fields can be left to their default values.

Note: Make sure the URL text box actually contains a value (font color should be white) and not the placeholder value (light grey text).

Note: The Redis and bpftrace data sources need additional configuration on the collector host. See *PCP Redis* and *PCP bpftrace*.

2.1.3 Dashboards

After installing grafana-pcp and configuring the data sources, you're ready to open the pre-installed dashboards or create new ones. Each data source comes with a few pre-installed dashboards, showing most of the respective functionality. Further information on each data source and the functionality can be found in the *Data Sources* section.

2.2 Installation

2.2.1 Distribution Package

This is the recommended method of installing grafana-pcp.

Fedora

```
$ sudo dnf install grafana-pcp
$ sudo systemctl restart grafana-server
```

2.2.2 From GitHub

If there is no package available for your distribution, you can install a release from GitHub.

```
$ wget https://github.com/performancecopilot/grafana-pcp/archive/v2.0.2.tar.gz
$ sudo tar xfz v2.0.2.tar.gz -C /var/lib/grafana/plugins
$ sudo systemctl restart grafana-server
```

2.2.3 From Source

The yarn package manager is required for building grafana-pcp.

```
$ git clone https://github.com/performancecopilot/grafana-pcp.git
$ yarn install
$ yarn run build
$ sudo ln -s $(pwd) /var/lib/grafana/plugins
$ sudo systemctl restart grafana-server
```

For interactive development, run `yarn run watch`.

2.3 Change Log

2.3.1 2.0.2 (2020-02-25)

- **vector, redis**: remove autocompletion cache (PCP metrics can be added and removed dynamically)

2.3.2 2.0.1 (2020-02-17)

- **build**: fix production build (implement workaround for systemjs/systemjs#2117, grafana/grafana#21785)

2.3.3 2.0.0 (2020-02-17)

- **vector, bpftrace**: fix version checks on dashboard load (prevent multiple pmcd.version checks on dashboard load)
- **vector, bpftrace**: change datasource check box to red if URL is inaccessible
- **redis:** add tests
- **flame graphs:** support multidimensional eBPF maps (required to display e.g. the process name)
- **dashboards**: remove BCC metrics from Vector host overview (because the BCC PMDA isn't installed by default)
- **misc**: update dependencies

2.3.4 1.0.7 (2020-01-29)

- **redis:** fix timespec (fixes empty graphs for large time ranges)

2.3.5 1.0.6 (2020-01-07)

- **redis:** support wildcards in metric names (e.g. `disk.dev.*`)
- **redis:** fix label support
- **redis:** fix legends

2.3.6 1.0.5 (2019-12-16)

- **redis:** set default sample interval to `60s` (fixes empty graph borders)
- **build:** upgrade `copy-webpack-plugin` to mitigate XSS vulnerability in the `serialize-javascript` transitive dependency
- **build:** remove deprecated `uglify-webpack-plugin`

2.3.7 2.0.0-beta1 (2019-12-12)

- support Grafana 6.5+, drop support for Grafana < 6.5

2.3.8 1.0.4 (2019-12-11)

Enhancements

- **flame graphs:** clean flame graph stacks every 5s (reduces CPU load)
- **general:** implement PCP version checks

Bug Fixes

- **build:** remove `weak` dependency (doesn't work with Node.js 12)
- **build:** upgrade `terser-webpack-plugin` to mitigate XSS vulnerability in the `serialize-javascript` transitive dependency

2.3.9 1.0.3 (2019-11-22)

- fix flame graph dependency (`flamegraph.destroy` error in javascript console)

2.3.10 1.0.2 (2019-11-12)

- handle counter wraps (overflows)
- convert time based counters to time utilization

2.3.11 1.0.1 (2019-10-24)

Flame Graphs

- aggregate stack counts by selected time range in the Grafana UI
- add option to hide idle stacks

Vector

- fix container dropdown in query editor
- remove container setting from datasource settings page

Redis

- fix value transformations (e.g. rate conversation of counters)

All

- request more datapoints from the datasource to fill the borders of the graph panel

2.3.12 1.0.0 (2019-10-11)

bpftrace

- support for Flame Graphs
- context-sensitive auto completion for bpftrace probes, builtin variables and functions incl. help texts
- parse output of bpftrace scripts (e.g. using `printf()`) as CSV and display it in the Grafana table panel
- sample dashboards (BPFtrace System Analysis, BPFtrace Flame Graphs)

Vector

- table output: show instance name in left column
- table output: support non-matching instance names (cells of metrics which don't have the specific instance will be blank)

Vector & bpftrace

- if the metric/script gets changed in the query editor, immeditately stop polling the old metric/deregister the old script
- improve pmwebd compatibility

miscellaneous

- help texts for all datasources (visible with the **[?]** button in the query editor)
- renamed PCP Live to PCP Vector
- logos for all datasources
- improved error handling

2.3.13 0.0.7 (2019-08-16)

- Initial release of grafana-pcp

Features

- retrieval of Performance Co-Pilot metrics from pmseries (PCP Redis), pmproxy and pmwebd (PCP Live)
- automatic rate conversation of counter metrics
- auto completion of metric names 1,2, qualifier keys and values 2
- display of semantics, units and help texts of metrics 1
- legend templating support with `$metric`, `$metric0`, `$instance`, `$some_label`
- container support
- support for repeating panels
- support for custom endpoint URL and container setting per query, with templating support 1
- heatmap and table support 1
- sample dashboards for PCP Redis and PCP Live

1 PCP Live 2 PCP Redis

Known Bugs

- the bpftrace datasource is work-in-progress and will be ready with the next release (approx. 1-2 weeks)

Thanks to Jason Koch for the initial pcp-live datasource implementation and the host overview dashboard.

2.4 Overview

2.4.1 PCP Redis

This data source queries the fast, scalable time series capabilities provided by the pmseries functionality. It is intended to query **historical** data across **multiple hosts** and supports filtering based on labels.

2.4.2 PCP Vector

The PCP Vector data source shows **live, on-host metrics** from the real-time pmwebapi interfaces. It is intended for individual host, on-demand performance monitoring and includes container support.

2.4.3 PCP bpftrace

The PCP bpftrace data source supports system introspection using bpftrace scripts. It connects to the bpftrace PMDA and runs bpftrace scripts on the host.

2.5 PCP Redis

2.5.1 Introduction

This data source provides a native interface between Grafana and Performance Co-Pilot (PCP), allowing PCP metric data to be presented in Grafana panels, such as graphs, tables, heatmaps, etc. Under the hood, the data source makes REST API query requests to the PCP pmproxy service, which can be running either locally or on a remote host. The pmproxy daemon can be local or remote, and uses the Redis time-series database (local or remote) for persistent storage.

2.5.2 Setup Redis and PCP daemons

```
$ sudo dnf install redis
$ sudo systemctl start redis pmlogger pmproxy
```

2.5.3 Query Language

Syntax: `[metric.name] '{metadata qualifiers}'`

Examples:

```
kernel.all.load
kernel.all.load{hostname == "web01"}
network.interface.in.bytes{agent == "linux"}
```

Documentation of the pmseries query language can be found in the man page of pmseries.

2.5.4 Query Formats

Time Series

Returns the data as time series. If there are multiple series for a metric, all series will be shown as separate targets (i.e. a line in a line graph). For metrics with instance domains, each instance is shown as a separate target. If there are multiple queries defined, all values will be combined in the same graph.

Table

Transforms the data for the table panel. Two or more queries are required, and it will transform every metric into a column, and every instance into a row. The latest values of the current selected timeframe will be displayed.

2.5.5 Legend Format Templating

The following variables can be used in the legend format box:

Variable	Description	Example
`$metric`	metric name	`disk.dev.read`
`$metric0`	last part of metric name	`read`
`$instance`	instance name	`sda`
`$some_label`	label value	anything

2.5.6 Query Functions

The following functions are available for dashboard variables of type *Query*:

Function	Description	Example
`metrics([pattern])`	returns all metrics matching a glob pattern (if no pattern is defined, all metrics are returned)	`metrics(disk.*)`
`label_values(metric, label)`	returns all label values for the specified label of the specified metric	`label_values(kernel.all.uptime, hostname)`

2.6 PCP Vector

2.6.1 Query Formats

Time Series

Returns the data as time series. For metrics with instance domains, each instance is shown as a separate target (i.e. line in a line graph). If there are multiple queries defined, all values will be combined in the same graph.

Heatmap

Transforms the data for the heatmap panel. Instance names have to be in the following format: `<lower_bound>-<upper_bound>`, for example `512-1023` (the bcc PMDA produces histograms in this format).

The following settings have to be set in the heatmap panel options:

Setting	Value
Format	**Time Series Buckets**
Bucket bound	**Upper**

Table

Transforms the data for the table panel. Two or more queries are required, and it will transform every metric into a column, and every instance into a row. The latest values of the current selected timeframe will be displayed.

2.6.2 Legend Format Templating

The following variables can be used in the legend format box:

Variable	Description	Example
`$metric`	metric name	`disk.dev.read`
`$metric0`	last part of metric name	`read`
`$instance`	instance name	`sda`
`$some_label`	label value	anything

2.7 PCP bpftrace

2.7.1 bpftrace PMDA installation

```
$ sudo dnf install pcp-pmda-bpftrace
$ cd /var/lib/pcp/pmdas/bpftrace
$ sudo ./Install
```

2.7.2 Query Formats

Time Series

Shows bpftrace variables as time series. For bpftrace maps, each key is shown as a separate target (i.e. line in a line graph), for example `@counts[comm] = count()`. If there are multiple variables (or scripts) defined, all values will be combined in the same graph.

Heatmap

Transforms bpftrace histograms into heatmaps.

The following settings have to be set in the heatmap panel options:

Setting	Value
Format	**Time Series Buckets**
Bucket bound	**Upper**

Table

Transforms CSV output of bpftrace scripts into a table. The first line must be the column names.

2.7.3 Legend Format Templating

The following variables can be used in the legend format box:

Variable	Description
`$metric0`	bpftrace variable name
`$instance`	bpftrace map key

2.7.4 More Information

bpftrace PMDA README

2.8 Troubleshooting

2.8.1 Common Problems

When I try to add a datasource in Grafana I get: "HTTP Error 502: Bad Gateway, please check the datasource and pmproxy settings. To use this data source, please configure the URL in the query editor."

- check if pmproxy is running: `systemctl status pmproxy`
- make sure that pmproxy was built with timeseries (libuv) support enabled. You can find out if so in *$PCP_LOG_DIR/pmproxy/pmproxy.log*