p3exporter Documentation

codeaffen

Dec 09, 2021

USER DOCUMENTATION

CHAPTER

ONE

PYTHON PROGRAMMABLE PROMETHEUS EXPORTER

p3exporter will help any DevOps to quickstart its Prometheus exporter development. It is completly written in python and provides a facility for pluggable metric collectors. The exporter comes with real life exporters to illustrate how it works but is also intended to use as a framework for completely custom collectors.

The included collectors were only tested on linux systems. Other *nix derivates are not supported by us but you are welcome to contribute to bring this exporter to a broader audience.

1.1 Installation and Running

There are different ways to run the exporter on your system. Our exporter listen on tcp/5876 by default. You can change this by adding `--port` or `-p` option with the port of your choice.

1.1.1 Running exporter as docker container

The simplest way will is to start it as docker container. The container image is hosted on dockerhub and the latest tag represent the `develop` branch of the github repository. If you want to use a given version you can us the verson string (e.g. `v1.0.0`) as tag instead.

```
docker run -d --net="host" --pid="host" -v "/:/host:ro,rslave" codeaffen/
↪p3exporter:latest
```

1.1.2 Installing from pypi.org

We also release all versions on pypi so you can use `pip` to install the exporter and run it locally.

```
pip install p3exporter
```

This will install the exporter and all of its dependencies. Now you can start it as every other program. You need to add `--config` or `-c` option with path to your `p3.yml` file.

```
$ curl --silent https://raw.githubusercontent.com/codeaffen/p3exporter/develop/p3.yml --
↪output ~/tmp/p3.yml
$ p3exporter --config ~/tmp/p3.yml
INFO:root:Collector 'example' was loaded and registred successfully
INFO:root:Collector 'loadavg' was loaded and registred successfully
```

(continues on next page)

(continued from previous page)

```
INFO:root:Collector 'netdev' was loaded and registred successfully
INFO:root:Start exporter, listen on 5876
```

1.1.3 Install from repository

The last option to install and run p3exporter is to install it from a local clone of our github repository.

```
$ git clone https://github.com/codeaffen/p3exporter.git
Cloning into 'p3exporter'...
remote: Enumerating objects: 158, done.
remote: Counting objects: 100% (158/158), done.
remote: Compressing objects: 100% (112/112), done.
remote: Total 158 (delta 63), reused 101 (delta 28), pack-reused 0
Receiving objects: 100% (158/158), 188.37 KiB | 1.08 MiB/s, done.
Resolving deltas: 100% (63/63), done.
$ cd p3exporter
$ pip install -e .
```

From now you can run it with:

```
$ p3exporter
INFO:root:Collector 'example' was loaded and registred successfully
INFO:root:Collector 'loadavg' was loaded and registred successfully
INFO:root:Collector 'netdev' was loaded and registred successfully
INFO:root:Start exporter, listen on 5876
```

1.2 Building your own container image

To build your own container image you can use the dockerfile which is delivered in our github repository. This file is also used to create our images on dockerhub.

```
$ docker build -t p3exporter .
Sending build context to Docker daemon  181.8kB
...
Successfully built a6bdf60489f5
Successfully tagged p3exporter:latest
```

Now you can start the container. Here you can use the command from above. You have just to use your image

```
docker run -d --net="host" --pid="host" -v "/:/host:ro,rslave" p3exporter:latest
```

1.3 Collectors

Name	Description
example	example collector that actually does nothing but show how long a function has been executed
loadavg	collects average load in 1, 5 and 15 minutes interval
netdev	collects netword device information and statistics

1.3.1 Activation and Deactivation of collectors

To activate or deactive collectors you have to configure it in `p3.yml` within the `collectors` list. All collectors listed in this list will be activated a start time of p3exporter. If there are any issues e.g. collector can't be found or has failures in code a warning will be shown and it will not be activated.

```
exporter_name: "Python prammable Prometheus exporter"
collectors:
  - example
  - loadavg
  - netdev
collector_opts:
  netdev:
    whitelist:
    blacklist:
      - docker0
      - lo
```

CHAPTER

TWO

CHANGELOG

All notable changes to this project will be documented in this file. This project adheres to Semantic Versioning and Keep a Changelog.

2.1 Unreleased

2.1.1 New

2.1.2 Changes

- Switch sphinx from recommonmark to myst_parser

2.1.3 Fixes

2.1.4 Breaks

- We remove support for python prior than 3.7

2.2 1.1.2 - (2021-04-15)

2.2.1 Fixes

- #34 - custom collectors with underscore in name are not supported

2.3 1.1.1 - (2021-04-12)

2.3.1 Changes

- linting code
- cleanup code and documentation
- fix typos

2.4 1.1.0 - (2021-03-30)

2.4.1 New

- introduce `CollectorBase` class to derive new collectors from
- added cache module with timed lru cache
- add netdev collector for network information and statistics

2.4.2 Changes

- reduce docker image size
- we switched base image from python:3-slim to alpine

2.5 1.0.0 - (2021-03-22)

2.5.1 New

- now it is simplier to add new collectors. You have to simply follow the naming convention
- add loadavg collector as a real life example

2.5.2 Breaks

- change load and registration behavior for collectors

2.6 0.1.0 - (2021-03-04)

2.6.1 Changes

- move collector to sub module

2.6.2 Fixes

- signal handling print now clean log messages instead of exceptions

CHAPTER

THREE

P3EXPORTER

3.1 p3exporter package

Init methods for p3exporter package.

p3exporter.**main**()
Start the application.

p3exporter.**shutdown**()
Shutdown the app in a clean way.

p3exporter.**signal_handler**(*signum*, *frame*)
Will be called if a signal was catched.

3.1.1 Subpackages

p3exporter.cache package

Module that defines all needed classes and functions for caching facility.

p3exporter.cache.**timed_lru_cache**(*lifetime: int = 3600*, *maxsize: int = 128*)
Provide cache with a given lifetime.

This function has to be used as decorator.

Each time the the cache will be accessed the decorator checks current date is past experation date. If so, the cache will cleared and the new expiration data will be recomputed. If not the cache entry will be delivered.

Parameters

- **lifetime** (*int, optional*) – The lifetime of cache in seconds, defaults to 3600
- **maxsize** (*int, optional*) – The maximum number of cache items, defaults to 128

p3exporter.collector package

Entry point for collector sub module.

class p3exporter.collector.**Collector**(*config:* p3exporter.collector.CollectorConfig)
Bases: object

Base class to load collectors.

All collectors have to be placed inside the directory *collector*. You have to follow the naming convention:

1. Place the collector code in a <name>.py file (e.g. *my.py*)
2. Within the file <name>.py` a class <Name>Collector (e.g. *MyController*) needs to be defined. This is the main collector class which will be imported, instantiate and registered automatically.

class p3exporter.collector.**CollectorBase**(*config:* p3exporter.collector.CollectorConfig)
Bases: object

Base class for all collectors.

This class will provide methods that do generic work.

property collector_name_from_class
Convert class name to controller name.

The class name must follow naming convention:

- camelized string
- starts with camelized module name
- ends with 'Collector'

This will convert <Name>Collector class name to <name> collector name. Examples for valid names:

- MyCollector => my
- FooBarCollector => foo_bar
- FooBarBazCollector => foo_bar_baz

Returns collector name in snake case

Return type string

class p3exporter.collector.**CollectorConfig**(***kwargs*)
Bases: object

Class that provide all the logic needed for configuration handling.

Submodules

p3exporter.collector.example module

Module that defines all needed classes and functions for example collector.

class p3exporter.collector.example.**ExampleCollector**(*config:* p3exporter.collector.CollectorConfig)
Bases: *p3exporter.collector.CollectorBase*

A sample collector.

It does not really do much. It only runs a method and return the time it runs as a gauge metric.

collect()
Collect the metrics.

p3exporter.collector.loadavg module

Module that defines all needed classes and functions for loadavg collector.

class p3exporter.collector.loadavg.**LoadavgCollector**(*config:* p3exporter.collector.CollectorConfig)
Bases: *p3exporter.collector.CollectorBase*

Load avg collector class.

collect()
Collect load avg for 1, 5 and 15 minutes interval.

Returns three gauge metrics. One for each load.

p3exporter.collector.netdev module

Module that defines all needed classes and functions for netdev collector.

class p3exporter.collector.netdev.**NetdevCollector**(*config:* p3exporter.collector.CollectorConfig)
Bases: *p3exporter.collector.CollectorBase*

Netdev collector class.

collect()
Collect netdev metrics.

Returns several info, couter and gauge metrics for interfaces.

3.1.2 Submodules

3.1.3 p3exporter.web module

Web module provide all parts to create the web app.

p3exporter.web.**create_app**(*config:* p3exporter.collector.CollectorConfig)
Create the web app.

This Function creates the flask app and dispatch metrics endpoint to prometheus wsgi app.

Parameters **config** (CollectorConfig) – A configuration object with data for template rendering.

Returns Created web app object.

Return type DispatcherMiddleware

CHAPTER

FOUR

HOW TO CONTRIBUTE TO P3EXPORTER

4.1 Did you found a bug

- Make sure the bug is not already opened by another user.
- If you can't find an open issue which reflects your observed problem go ahead an open a new bug.
- Provide as much information as mentioned in the bug report template.

4.2 Did you wrote a patch for an open bug

- Open new pull request containing the patch.
- Provide a clear description which describes the problem and the solution. Link the existing bug to the PR.

4.3 Do you want to add a new feature

- Make sure there isn't already a feature request.
- If you can't find an open feature request which describe your feature idea or parts of it feel free to open a new feature request.
- Suggest your feature idea within the created feature request.
- Provide as much discribtion as possible to enable others to have a good understanding of what you are doing.
- Point out that you want to start to work on the new feature

4.4 Do you wnat to contribute to documentation

- Write you documentation change.
- Open a PR with your change.
- Discuss with the team about your changes.

4.5 Thank you for any contribution

We will thank you for heed the contribution guidelines and we encourage you to contribute and join the team.

CHAPTER

FIVE

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

p

INDEX