Flask-Graphite

Jun 03, 2019

Contents

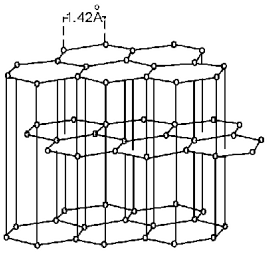
1 Example **3**

2 Documentation **5**
- 2.1 Installation . . . 5
- 2.2 Get Started . . . 6
- 2.3 Carbon aggregator . . . 7
- 2.4 Configuring Flask-Graphite . . . 7
- 2.5 Gathered metrics . . . 8
- 2.6 API Reference . . . 9

3 Indices and tables **13**

Python Module Index **15**

Index **17**



Flask-*graphite*

Flask-Graphite grants you the power to push useful metrics for each request without effort

Documentation: https://flask-graphite.readthedocs.io.

- Send metrics to graphite for each request
- Metric name based on the route of the request
- Average processing time, number of requests, and stats about status code for each route

CHAPTER 1

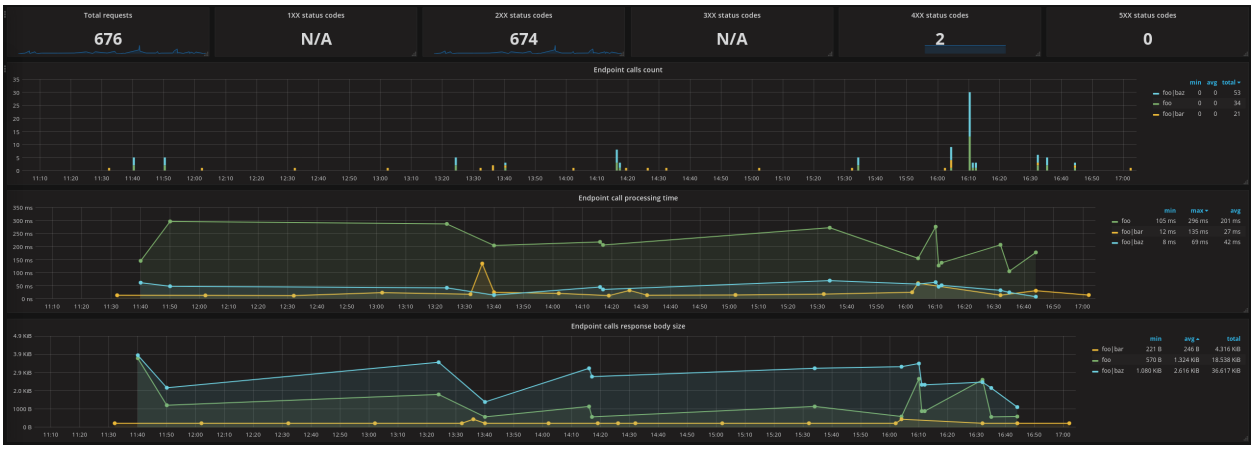
Example

Here is a minimal template to use Flask-Graphite in a project.

```
from flask import Flask
from flask_graphite import FlaskGraphite

app = Flask(__name__)
FlaskGraphite(app)
```

Such a simple snippet, combined with a Grafana dashboard, would give you something like this:



CHAPTER 2

Documentation

2.1 Installation

2.1.1 Stable release

To install Flask-Graphite, run this command in your terminal:

```
$ pip install flask_graphite
```

This is the preferred method to install Flask-Graphite, as it will always install the most recent stable release.

If you don't have pip installed, this Python installation guide can guide you through the process.

2.1.2 From sources

The sources for Flask-Graphite can be downloaded from the Github repository.

You can either clone the public repository:

```
$ git clone git://github.com/numberly/flask_graphite
```

Or download the tarball:

```
$ curl  -OL https://github.com/numberly/flask_graphite/tarball/master
```

Once you have a copy of the source, you can install it with:

```
$ python setup.py install
```

2.2 Get Started

2.2.1 Complete example

Here is a complete example of an application using FlaskGraphite

Write this code in a *test.py* file:

```
from flask import Flask, jsonify
from flask_graphite import FlaskGraphite


metric_sender = FlaskGraphite()

app = Flask(__name__)
app.config["FLASK_GRAPHITE_HOST"] = "localhost"
app.config["FLASK_GRAPHITE_PORT"] = 2023

metric_sender.init_app(app)


@app.route("/foo")
def foo():
    return jsonify({"test": "foo"})


@app.route("/bar")
def bar():
    return jsonify({"test": "bar"})


if __name__ == "__main__":
    app.run(host="localhost", port=5000)
```

2.2.2 Run this example

You can launch the server:

```
$ python test.py
```

You can then make requests to the server:

```
$ curl http://localhost:5000/foo
$ curl http://localhost:5000/bar
```

By doing this, the *foo* view will be executed, and thanks to the Flask-Graphite plugin, a number of metrics will be available on your graphite server.

Warning: A running instance of carbon-aggregator is needed. See *Carbon aggregator*

2.2.3 Generated metrics

When you run this example, the application will send several metrics to your local instance of carbon.

You can see these metrics in the directory used by carbon to store the metrics (by default `/var/lib/carbon/whisper/`).

You will see that a different set of metric has been generated for the `foo` and the `bar` view. For each of them, you have access to the *standard metrics*

2.3 Carbon aggregator

2.3.1 Why is it needed ?

A running carbon-aggregator instance is needed to use Flask-Graphite.

Since a data point is sent for each requests, they must be aggregated to be useful. Even if they were buffered, any application with multiple workers would still need to aggregate the time series.

If they weren't, each data point would override the previous one, and only the last point received each minute would be considered.

2.3.2 Configuring carbon-aggregator

You can setup the aggregation rules based on the last part of metric names, if you're sure it will not collide with other applications.

As a general rule, counter metrics should be summed, while others should be averaged. Below is a list of aggregation suitable for a simple environment.

```
<route>.count (60) = sum <<route>>.count
<route>.status_code.<type> (60) = sum <<route>>.status_code.<type>
<route>.pt (60) = avg <<route>>.pt
<route>.size (60) = avg <<route>>.size
```

You can consult the carbon documentation for more informations on the syntax.

2.4 Configuring Flask-Graphite

2.4.1 How to configure Flask-Graphite?

You can configure Flask-Graphite through the Flask application's configuration.

The `FLASK_GRAPHITE_` namespace will be fetched from the application's configuration and used as Flask-Graphite own configuration.

2.4.2 Configuration keys

Here is a succinct list of different options available, and their respective meanings.

NAME	Default	Description
FLASK_GRAPHITE_HOST	local-host	The host of the carbon-aggregator installation.
FLASK_GRAPHITE_PORT	2023	The port of the carbon-aggregator installation.
FLASK_GRAPHITE_PREFIX		A prefix that will be prepended to all metrics.
FLASK_GRAPHITE_GROUP	flask-graphite	A string that will be added between the host and the actual metric name to generate the complete metric name.
FLASK_GRAPHITE_AUTORECONNECT	True	Automatically try to reconnect to graphite server.
FLASK_GRAPHITE_METRIC_TEMPLATE	url_rule	The Flask request's attribute that should be used to generate the metric name.

Hint: Any parameter accepted by the graphitesender client is also valid when prefixed with FLASK_GRAPHITE_.

2.5 Gathered metrics

After each request, some metrics are sent to carbon. Each metrics uses the after_request hooks to be compute and send the good value.

A different metric is used for each route of the application.

2.5.1 Metric format

The URL Rule matched by the requested is parsed to create the *route* part of the metric.

The parsing happens on the URI of the URL Rule. Each converter within this URI is replaced with its variable name and the slash characters are replaced with dot characters. Below are some examples of such transformations:

matched URI	corresponding route part
/foo	foo
/foo/bar/baz	foo.bar.baz
/foo/<int:bar>	foo.bar
/foo/<string:bar>/baz/<int:boo>	foo.bar.baz.boo

In addition to this route part, a suffix is used depending on the metric sent. These parts are covered in the next section.

2.5.2 List of the gathered metrics

In this section, <route> refer to the route part as described in the previous section. the <status_code> placeholed represents the status code sent in response, to the request, and the <status_type> is simply the first digit of the status code.

name	metrics	value
processing time	<route>.pt	The processing time of the route
number of request	<route>.count	The number of request received on this route
size of request	<route>.size	The size of the response for this route
status code	<route>.status_code.<status_code>	A counter for each status_code received
status type	<route>.status_type.<status_type>XX	A counter for each type of status code (eg. 2XX)

2.6 API Reference

class flask_graphite.**FlaskGraphite**(*app=None*)
Register a list of hooks meant to monitor a flask application

The configuration is read from the namespace *FLASK_GRAPHITE_* of the application configuration.

The main options are:

- FLASK_GRAPHITE_HOST: The Carbon host to send metrics
- FLASK_GRAPHITE_PORT: The Carbon port to send metrics

You can read Configuring Flask-Graphite to learn about the other option configurations.

Parameters **app** – The application to monitor

init_app(*app*)
Read config and set the hooks in place to monitor the application

Parameters **app** – The application to monitor

class flask_graphite.**MetricHook**(*function*, *type='after_request'*)
Represent a hook for flask requests

This hook decorates a function to make it a suitable Flask hook.

The function *must* return a 2-tuple which represents a metric name and it's value.

Parameters

- **function** – The function used as hook
- **type** – The type of hook (before_request, after_request or teardown_request)

is_setup_hook
Property to test if a hook is a setup hook

register_into(*obj*)
Register the hook as a request hook in *obj*

Can only be used on setup hooks. Bind the hook to a client for other types of hooks.

Parameters

- **obj** – Either an application or a blueprint
- **client** – The client to use with this hook for this application

setup(*function*)
Mark a function as a setup hook for this hook

A setup hook is a hook required to run before its main hook. It's implemented as a before_request hook.

Parameters **function** – The function used as setup hook

class flask_graphite.hooks.**MetricHook**(*function*, *type='after_request'*)
Represent a hook for flask requests

This hook decorates a function to make it a suitable Flask hook.

The function *must* return a 2-tuple which represents a metric name and it's value.

Parameters

- **function** – The function used as hook
- **type** – The type of hook (before_request, after_request or teardown_request)

is_setup_hook
Property to test if a hook is a setup hook

register_into(*obj*)
Register the hook as a request hook in *obj*

Can only be used on setup hooks. Bind the hook to a client for other types of hooks.

Parameters

- **obj** – Either an application or a blueprint
- **client** – The client to use with this hook for this application

setup(*function*)
Mark a function as a setup hook for this hook

A setup hook is a hook required to run before its main hook. It's implemented as a before_request hook.

Parameters **function** – The function used as setup hook

flask_graphite.request_hooks.**request_count**(*response*)
Represent a hook for flask requests

This hook decorates a function to make it a suitable Flask hook.

The function *must* return a 2-tuple which represents a metric name and it's value.

Parameters

- **function** – The function used as hook
- **type** – The type of hook (before_request, after_request or teardown_request)

flask_graphite.request_hooks.**request_status_code**(*response*)
Represent a hook for flask requests

This hook decorates a function to make it a suitable Flask hook.

The function *must* return a 2-tuple which represents a metric name and it's value.

Parameters

- **function** – The function used as hook
- **type** – The type of hook (before_request, after_request or teardown_request)

flask_graphite.request_hooks.**request_status_type**(*response*)
Represent a hook for flask requests

This hook decorates a function to make it a suitable Flask hook.

The function *must* return a 2-tuple which represents a metric name and it's value.

Parameters

- **function** – The function used as hook
- **type** – The type of hook (before_request, after_request or teardown_request)

flask_graphite.request_hooks.**request_processing_time**(*response*)
Represent a hook for flask requests

This hook decorates a function to make it a suitable Flask hook.

The function *must* return a 2-tuple which represents a metric name and it's value.

Parameters

- **function** – The function used as hook
- **type** – The type of hook (before_request, after_request or teardown_request)

flask_graphite.request_hooks.**set_start_time**()
Represent a hook for flask requests

This hook decorates a function to make it a suitable Flask hook.

The function *must* return a 2-tuple which represents a metric name and it's value.

Parameters

- **function** – The function used as hook
- **type** – The type of hook (before_request, after_request or teardown_request)

flask_graphite.request_hooks.**response_size**(*response*)
Represent a hook for flask requests

This hook decorates a function to make it a suitable Flask hook.

The function *must* return a 2-tuple which represents a metric name and it's value.

Parameters

- **function** – The function used as hook
- **type** – The type of hook (before_request, after_request or teardown_request)

flask_graphite.utils.**get_request_metric_prefix**()
Turn the URI of the current request into a metric

> **Warning:** You must be inside a Flask request context to call this function.

CHAPTER 3

Indices and tables

- genindex
- modindex
- search

Python Module Index

f

flask_graphite, 9
flask_graphite.hooks, 9
flask_graphite.request_hooks, 10
flask_graphite.utils, 11

Index

F

flask_graphite (*module*), 9
flask_graphite.hooks (*module*), 9
flask_graphite.request_hooks (*module*), 10
flask_graphite.utils (*module*), 11
FlaskGraphite (*class in flask_graphite*), 9

G

get_request_metric_prefix() (*in module flask_graphite.utils*), 11

I

init_app() (*flask_graphite.FlaskGraphite method*), 9
is_setup_hook (*flask_graphite.hooks.MetricHook attribute*), 9
is_setup_hook (*flask_graphite.MetricHook attribute*), 9

M

MetricHook (*class in flask_graphite*), 9
MetricHook (*class in flask_graphite.hooks*), 9

R

register_into() (*flask_graphite.hooks.MetricHook method*), 10
register_into() (*flask_graphite.MetricHook method*), 9
request_count() (*in module flask_graphite.request_hooks*), 10
request_processing_time() (*in module flask_graphite.request_hooks*), 10
request_status_code() (*in module flask_graphite.request_hooks*), 10
request_status_type() (*in module flask_graphite.request_hooks*), 10
response_size() (*in module flask_graphite.request_hooks*), 11

S

set_start_time() (*in module flask_graphite.request_hooks*), 11
setup() (*flask_graphite.hooks.MetricHook method*), 10
setup() (*flask_graphite.MetricHook method*), 9