aiographite Documentation

Release 0.1

Yun Xu

Dec 21, 2017

Contents

An asyncio library for graphite.

CHAPTER 1

What is aiographite ?

aiographite is Python3 library ultilizing asyncio, designed to help Graphite users to send data into graphite easily.

CHAPTER 2

Quick start

Let's get started.

```
from aiographite import connect
from aiographite.protocol import PlaintextProtocol
import asyncio

"""
  Initialize a aiographite instance
"""
loop = asyncio.get_event_loop()
plaintext_protocol = PlaintextProtocol()
graphite_conn = await connect(*httpd.address, plaintext_protocol, loop=loop)


"""
  Send a tuple (metric, value , timestamp)
"""
graphite_conn.send(metric, value, timestamp)


"""
  Send a list of tuples List[(metric, value , timestamp)]
"""
graphite_conn.send_multiple(list)


"""
  aiographite library also provides GraphiteEncoder module,
  which helps users to send valid metric name to graphite.
  For Example: (metric_parts, value ,timestamp)
"""
metric = graphite_conn.clean_and_join_metric_parts(metric_parts)
graphite_conn.send(metric, value, timestamp)
```

```
"""
  Close connection
"""
graphite_conn.close()
```

Contents:

2.1 Installation

2.1.1 Installing it globally

You can install aiographite globally with any Python package manager:

```
pip install aiographite
```

2.2 AIOGraphite

AIOGraphite is a Graphite client class, ultilizing asyncio, designed to help Graphite users to send data into graphite easily.

```
from aiographite.aiographite import connect

"""
  Initialize a aiographite instance
"""
loop = asyncio.get_event_loop()
plaintext_protocol = PlaintextProtocol()
graphite_conn = await aiographite.connect(*httpd.address, plaintext_protocol,␣
↪loop=loop, timeout=None)


"""
  Send a tuple (metric, value , timestamp)
"""
graphite_conn.send(metric, value, timestamp)


"""
  Send a list of tuples List[(metric, value , timestamp)]
"""
graphite_conn.send_multiple(list)


"""
  aiographite library also provides GraphiteEncoder module,
  which helps users to send valid metric name to graphite.
  For Example: (metric_parts, value ,timestamp)
"""
metric = graphite_conn.clean_and_join_metric_parts(metric_parts)
graphite_conn.send(metric, value, timestamp)


"""
```

```
  Close connection
"""
graphite_conn.close()
```

2.2.1 Full API Reference

class aiographite.aiographite.**AIOGraphite**(*graphite_server*, *graphite_port=2003*, *protocol=<aiographite.protocol.PlaintextProtocol object>*, *loop=None*, *timeout=None*)

AIOGraphite is a Graphite client class, ultilizing asyncio, designed to help Graphite users to send data into graphite easily.

clean_and_join_metric_parts(*metric_parts: typing.List[str]*) → str

This method helps encode any input metric to valid metric for graphite in case that the metric name includes any special character which is not supported by Graphite.

args: a list of metric parts(string).

returns a valid metric name for graphite.

example:

```
metric = aiographite.clean_and_join_metric_parts(metric_parts)
```

close() → None

Close the TCP connection to graphite server.

send(*metric: str*, *value: int*, *timestamp: int = None*) → None

send a single metric.

args: metric, value, timestamp. (str, int, int).

send_multiple(*dataset: typing.List[typing.Tuple], timestamp: int = None*) → None

send a list of tuples.

args: a list of tuples (metric, value, timestamp), and timestamp is optional.

2.3 Protocols

AIOGraphite support two protocols:

- The plaintext protocol
- The pickle protocol

2.3.1 Plaintext Protocol

class aiographite.protocol.**PlaintextProtocol**

generate_message(*listOfTuples: typing.List[typing.Tuple[[str, int], int]]*) → bytes

This method helps generate message with proper format for plaintext protocol.

args: a list of tuples (metric, value, timestamp).

2.3.2 Pickle Protocol

class aiographite.protocol.**PickleProtocol**

generate_message(*listOfTuples: typing.List[typing.Tuple[[str, int], int]]*) → bytes
This method helps generate message with proper format for pickle protocol.

args: a list of tuples (metric, value, timestamp).

2.4 GraphiteEncoder

aiographite library also provides GraphiteEncoder module, which helps users to send valid metric name to graphite.

2.4.1 Full API Reference

class aiographite.graphite_encoder.**GraphiteEncoder**
Graphite expects everything to be just ASCII to split/processing them, and then make directories based on metric name. So any special name not allow to appear in directory/file name is not supported by Graphite.

GraphiteEncoder is designed to help users to send valid metric name to graphite.

Metrics: <section_name>.<section_name>.<section_name>.<section_name>

static decode(*idna_str*)
This method helps to decode a valid metric name in graphite to its original metric name.

args: a valid metric name in graphite.

returns the original metric name.

static encode(*section_name*)
This method helps to encode any input metric name to a valid graphite metric name.

args: section name(could include any character), a string

returns valid metric name for graphite

2.5 Example

A simple example.

```
from aiographite.protocol import PlaintextProtocol
from aiographite import connect
import time
import asyncio


LOOP = asyncio.get_event_loop()
SERVER = '127.0.0.1'
PORT = 2003


async def test_send_data():
  # Initiazlize an aiographite instance
  plaintext_protocol = PlaintextProtocol()
```

```
  graphite_conn = await connect(SERVER, PORT, plaintext_protocol, loop=LOOP)

  # Send data
  timestamp = time.time()
  for i in range(10):
      await graphite_conn.send("yun_test.aiographite", i, timestamp + 60 * i)


def main():
  LOOP.run_until_complete(test_send_data())
  LOOP.close()


if __name__ == '__main__':
  main()
```

2.6 Development

aiographite accepts contributions on GitHub, in the form of issues or pull requests.

2.6.1 Running the tests

Run unit tests.

```
./uranium test
```

CHAPTER 3

Indices and tables

- genindex
- modindex
- search

Index