Asyncflux Documentation

Release 0.0+

Jorge Puente-Sarrín

August 07, 2015

Contents

Asynchronous client for InfluxDB and Tornado.

CHAPTER 1

Installation

You can use pip to install Asyncflux:

```
$ pip install git+https://github.com/puentesarrin/asyncflux.git
```

CHAPTER 2

Documentation

Sphinx is needed to generate the documentation. Documentation can be generated by issuing the following commands:

```
$ cd docs
$ make html
```

Or simply:

```
$ python setup.py doc
```

Also, the current documentation can be found at ReadTheDocs.

CHAPTER 3

License

Asyncflux is available under the Apache License, Version 2.0.

CHAPTER 4

Indices and tables

- genindex
- modindex
- search

4.1 Package Documentation

4.1.1 `asyncflux` Package

Asynchronous client for InfluxDB and Tornado.

`asyncflux.__init__.`**`version`** = '0.0+'
Current version of Asyncflux.

class `asyncflux.__init__.`**`AsyncfluxClient`**(*host=None*, *port=None*, *username=None*, *password=None*, *is_secure=False*, *io_loop=None*, ***kwargs*)

Bases: `object`

`HOST` = 'localhost'

`PASSWORD` = 'root'

`PORT` = 8086

`USERNAME` = 'root'

`authenticate_cluster_admin`(**args*, ***kwargs*)

`base_url`

`change_cluster_admin_password`(**args*, ***kwargs*)

`create_cluster_admin`(**args*, ***kwargs*)

`create_database`(**args*, ***kwargs*)

`delete_cluster_admin`(**args*, ***kwargs*)

`delete_database`(**args*, ***kwargs*)

`get_cluster_admin_names`(**args*, ***kwargs*)

`get_cluster_admins`(**args*, ***kwargs*)

`get_database_names`(**args*, ***kwargs*)

get_databases(**args*, ***kwargs*)

get_shard_spaces(**args*, ***kwargs*)

host

password

ping(**args*, ***kwargs*)

port

request(**args*, ***kwargs*)

username

4.1.2 asyncflux Modules

asyncflux.client – Connection to InfluxDB

Connection to InfluxDB

class asyncflux.client.**AsyncfluxClient**(*host=None*, *port=None*, *username=None*, *password=None*, *is_secure=False*, *io_loop=None*, ***kwargs*)

Bases: object

HOST = 'localhost'

PASSWORD = 'root'

PORT = 8086

USERNAME = 'root'

authenticate_cluster_admin(**args*, ***kwargs*)

base_url

change_cluster_admin_password(**args*, ***kwargs*)

create_cluster_admin(**args*, ***kwargs*)

create_database(**args*, ***kwargs*)

delete_cluster_admin(**args*, ***kwargs*)

delete_database(**args*, ***kwargs*)

get_cluster_admin_names(**args*, ***kwargs*)

get_cluster_admins(**args*, ***kwargs*)

get_database_names(**args*, ***kwargs*)

get_databases(**args*, ***kwargs*)

get_shard_spaces(**args*, ***kwargs*)

host

password

ping(**args*, ***kwargs*)

port

request(**args*, ***kwargs*)

username

asyncflux.database – Database level operations

Database level operations

class asyncflux.database.**Database**(*client*, *name*)
Bases: object

authenticate_user(**args*, ***kwargs*)

change_user_password(**args*, ***kwargs*)

change_user_permissions(**args*, ***kwargs*)

change_user_privileges(**args*, ***kwargs*)

client

create_user(**args*, ***kwargs*)

delete(**args*, ***kwargs*)

delete_user(**args*, ***kwargs*)

get_user(**args*, ***kwargs*)

get_user_names(**args*, ***kwargs*)

get_users(**args*, ***kwargs*)

name

update_user(**args*, ***kwargs*)

asyncflux.clusteradmins – Tools for cluster administration

asyncflux.testing – Unit testing support for asynchronous code

Unit testing support for asynchronous code

class asyncflux.testing.**AsyncfluxTestCase**(*methodName='runTest'*, ***kwargs*)
Bases: tornado.testing.AsyncTestCase

assert_mock_args(*fetch_mock*, *path*, *method='GET'*, *body=None*, *auth_username='root'*, *auth_password='root'*, **args*, ***kwargs*)

patch_fetch_mock(*client*)

setup_fetch_mock(*fetch_mock*, *status_code*, ***kwargs*)

stop_op(*result*, *error*)

asyncflux.testing.**gen_test**(*func=None*, *timeout=None*)
Testing equivalent of `@gen.coroutine`, to be applied to test methods.

`@gen.coroutine` cannot be used on tests because the *.IOLoop* is not already running. `@gen_test` should be applied to test methods on subclasses of *AsyncTestCase*.

Example:

```
class MyTest(AsyncHTTPTestCase):
    @gen_test
    def test_something(self):
        response = yield gen.Task(self.fetch('/'))
```

By default, `@gen_test` times out after 5 seconds. The timeout may be overridden globally with the `ASYNC_TEST_TIMEOUT` environment variable, or for each test with the `timeout` keyword argument:

```
class MyTest(AsyncHTTPTestCase):
    @gen_test(timeout=10)
    def test_something_slow(self):
        response = yield gen.Task(self.fetch('/'))
```

New in version 3.1: The `timeout` argument and `ASYNC_TEST_TIMEOUT` environment variable.

Changed in version 4.0: The wrapper now passes along `*args, **kwargs` so it can be used on functions with arguments.

`asyncflux.util` – General-purpose utilities

General-purpose utilities

`asyncflux.util.`**`asyncflux_coroutine`**(*f*)
A coroutine that accepts an optional callback.

Given a callback, the function returns None, and the callback is run with (result, error). Without a callback the function returns a Future.

`asyncflux.util.`**`snake_case`**(*string*)

`asyncflux.util.`**`snake_case_dict`**(*_dict*)

4.2 Release Notes

4.2.1 Next Release

Very soon

- Initial release.
- Added Sphinx docs and ReadTheDocs configuration.

a

A

B

C

D

G

H

N

P

R

S

U

V