txes2 Documentation

Release 0.0.1

Multiple authors

April 10, 2016

Contents

1 Setup **3**
- 1.1 Installing via pip . . . 3
- 1.2 Installing from source . . . 3
- 1.3 Running tests . . . 3

2 Usage Example **5**

3 API **7**

4 Contributing **11**
- 4.1 Style Guide . . . 11
- 4.2 Writing Tests . . . 11

5 Indices and tables **13**

`txes2` is a PyES-like ElasticSearch client for Twisted applications. `txes2` is a fork of txes with a more PEP8-friendly interface. Credits are available at the project's Contributors section.

`txes2` has the following goals:

- A clean, safe API for interacting with ElasticSearch using Twisted's deferred model.
- Support for all PyES features and more (in progress).
- 95% unit and integration test coverage (in progress).
- Excellent documentation coverage (in progress).

Contents:

CHAPTER 1

Setup

1.1 Installing via pip

```
$ pip install txes2
```

1.2 Installing from source

Clone repository and run `setup.py`

```
$ git clone git@github.com:lextoumbourou/txes2.git
$ cd txes2
$ python setup.py install
```

1.3 Running tests

1.3.1 Unit

```
./bin/unit
```

1.3.2 Integration

1. Start Vagrant box, which runs an ElasticSearch cluster with ports `9210` and `9211` forwarded to the separate ElasticSearch instances.

```
vagrant up --provision
```

2. Run integration tests.

```
./bin/integration
```

CHAPTER 2

Usage Example

```
from pprint import pprint

from twisted.internet import reactor
from twisted.internet.defer import inlineCallbacks

from txes2 import ElasticSearch


@inlineCallbacks
def example():
    index = 'main'
    doc_type = 'person'

    ########################
    # Setup connection
    ########################
    es = ElasticSearch('127.0.0.1:9200')

    ########################
    # Index a document
    ########################
    document = {'name': 'Travis Bickle', 'tagline': 'You talking to me?'}
    yield es.index(document, doc_type=doc_type, index=index, id=123)

    ########################
    # Retrieve a document
    ########################
    result = yield es.get(index=index, doc_type=doc_type, id=123)
    pprint(result)

    ########################
    # Refresh an index
    ########################
    yield es.refresh(index)

    ########################
    # Perform a query
    ########################
    query = {'query': {'match': {'name': 'Travis'}}}
    results = yield es.search(query, doc_type=doc_type, index=index)
    pprint(results)
```

```
if __name__ == '__main__':
    df = example()
    df.addErrback(lambda err: err.printTraceback())
    df.addCallback(lambda _: reactor.stop())
    reactor.run()
```

CHAPTER 3

API

class txes2.**ElasticSearch**(*servers='127.0.0.1:9200'*, *timeout=30*, *bulk_size=400*, *discover=True*, *retry_time=10*, *discovery_interval=300*, *default_indexes=None*, *autorefresh=False*, **args*, ***kwargs*)

A PyES-like ElasticSearch client.

Parameters

- **servers** – either a single ES server URL or list of servers. If you don't provide a scheme (eg *https://*) then the request will use HTTP by default.
- **timeout** (*int*) – connection timeout in seconds.
- **bulk_size** (*int*) – how much bulk data to accumulate before indexing (when indexing in bulk).
- **retry_time** (*int*) – frequency in seconds for retrying broken ES nodes.
- **discover** (*bool*) – if True, will autodiscover ES nodes at connection time.
- **discovery_interval** (*bool*) – time in seconds between node discovery after initial discovery, set to False to skip.
- **default_indexes** (*list*) – list of indexes to use by default when querying ES.
- **autorefresh** (*bool*) – should we perform index autorefresh.
- **persistent** (*bool*) – use persistent connection.
- **http_auth** (*tuple*) – optional http auth tuple in the form *('username', 'password')*.
- **pool** (*HTTPConnectionPool*) – optionally pass in HTTPConnectionPool instance to use for connection pool.

add_alias(*alias*, *indices*)

Add an alias to point to a set of indices.

analyze(*text*, *index*, *analyzer=None*)

Perform analysis on textual input.

change_aliases(**commands*)

Change the aliases stored.

A command is a tuple of (["add"|"remove"], index, alias).

You may specify multiple commands as additional arguments.

close_index(*index*)

Close an index.

cluster_health(*level='cluster'*, *wait_for_status=None*, *wait_for_relocating_shards=None*, *wait_for_nodes=None*, *timeout=30*)
Check the current cluster health.

cluster_nodes(*nodes=None*)
The cluster nodes info API.

cluster_state(*metrics=None*, *indices=None*, ***kwargs*)
Retrieve the cluster state.

Parameters

- **metric** – a list of metrics for filtering results (see ES docs).
- **indices** – a list of indicies for filtering results (see ES docs).

collect_info()
Collect info about the connection and fill the info dictionary.

count(*query*, *indexes=None*, *doc_types=None*, ***params*)
Execute a query against one or more indices & get the hit count.

create_index(*index*, *settings=None*)
Create an index with the optional settings dict.

create_index_if_missing(*index*, *settings=None*)
Create an index with the optional settings dict.

Doesn't fail when index already exists.

create_river(*river*, *river_name=None*)
Create a river.

delete(*index*, *doc_type*, *id*, *bulk=False*, ***query_params*)
Delete a document based on its id.

delete_alias(*alias*, *indices*)
Delete an alias.

delete_by_query(*indexes*, *doc_types*, *query*, ***query_params*)
Delete documents from one or more indexes/types from query.

delete_index(*index*)
Deletes an index.

delete_index_if_exists(*index*)
Deletes an index if it exists.

delete_mapping(*index*, *doc_type*, ***query_params*)
Delete a document type from a specific index.

delete_river(*river*, *river_name=None*)
Delete a river.

flush_bulk(*forced=False*)
Wait to process all pending operations.

force_bulk()
Force executing of all bulk data.

get(*index*, *doc_type*, *id*, *fields=None*, *routing=None*, ***query_params*)
Get a typed document from an index based on its id.

get_alias(*alias*)
Return a list of indices pointed to by a given alias.

Raises IndexMissionException if the alias does not exist.

get_indices(*include_aliases=False*)

Retrieve a dict where each key is the index name and each value is another dict containing the following properties:

- num_docs: Number of documents in the index or alias.
- **alias_for: Only present for an alias: holds a list of indices** which this is an alias for. Requires the `include_aliases` param to be `True`.

Example:

```
{u'website': {'num_docs': 10837},
```

u'website_alias': { 'alias_for': [u'website'], 'num_docs': 10837}}

get_mapping(*doc_type=None*, *indexes=None*)

Get the mapping definition

index(*doc*, *index*, *doc_type*, *id=None*, *parent=None*, *force_insert=None*, *bulk=False*, *version=None*, ***query_params*)

Index a dict into an index.

mget(*ids*, *index=None*, *doc_type=None*, ***query_params*)

Get multiples documents based on id.

ids can be: list of tuples: (index, type, id) list of ids: index and doc_type are required

more_like_this(*index*, *doc_type*, *id*, ***query_params*)

Execute a "more like this" query against one or more fields.

open_index(*index*)

Open an index.

optimize(*indexes=None*, *wait_for_merge=False*, *max_num_segments=None*, *only_expunge_deletes=False*, *refresh=True*, *flush=True*)

Optimize one or more indices.

partial_update(*index*, *doc_type*, *id*, *doc=None*, *script=None*, *script_file=None*, *params=None*, *upsert=None*, ***query_params*)

Partially update a document with a script.

If `script` is passed in, `script_file` is ignored. For `script_file` to work, ES >= 1.4.5 is required as per: https://github.com/elastic/elasticsearch/issues/10007

put_mapping(*doc_type*, *mapping*, *indexes=None*)

Register mapping definition for a specific type.

scan(*query*, *indexes=None*, *doc_type=None*, *scroll_timeout='10m'*, ***params*)

Start a scan eventually returning a Scroller.

search(*query*, *indexes=None*, *doc_type=None*, ***params*)

Execute a search against one or more indices.

servers

Return a list of servers available for connections.

set_alias(*alias*, *indices*)

Set and alias (possibly removing what it already points to).

status(*indexes=None*)

Retrieve the status of one or more indices.

update_settings(*index*, *settings*)
Update settings of an index.

CHAPTER 4

Contributing

4.1 Style Guide

All code should pass through flake8. This will be enforced at build time by Travis.

4.2 Writing Tests

Please ensure any new functionality has tests written for it. The tests are designed to be run as both integration tests or units tests depending on the existance of the `USE_MOCKS` environment variable. When run as unit tests, the `self._mocks` attribute should be populated with the dict expected to be returned by the `txes2.ElasticSearch.connection.execute` method. Refer to already written test for examples.

CHAPTER 5

Indices and tables

- genindex
- modindex
- search

A

add_alias() (txes2.ElasticSearch method), 7
analyze() (txes2.ElasticSearch method), 7

C

change_aliases() (txes2.ElasticSearch method), 7
close_index() (txes2.ElasticSearch method), 7
cluster_health() (txes2.ElasticSearch method), 7
cluster_nodes() (txes2.ElasticSearch method), 8
cluster_state() (txes2.ElasticSearch method), 8
collect_info() (txes2.ElasticSearch method), 8
count() (txes2.ElasticSearch method), 8
create_index() (txes2.ElasticSearch method), 8
create_index_if_missing() (txes2.ElasticSearch method), 8
create_river() (txes2.ElasticSearch method), 8

D

delete() (txes2.ElasticSearch method), 8
delete_alias() (txes2.ElasticSearch method), 8
delete_by_query() (txes2.ElasticSearch method), 8
delete_index() (txes2.ElasticSearch method), 8
delete_index_if_exists() (txes2.ElasticSearch method), 8
delete_mapping() (txes2.ElasticSearch method), 8
delete_river() (txes2.ElasticSearch method), 8

E

ElasticSearch (class in txes2), 7

F

flush_bulk() (txes2.ElasticSearch method), 8
force_bulk() (txes2.ElasticSearch method), 8

G

get() (txes2.ElasticSearch method), 8
get_alias() (txes2.ElasticSearch method), 8
get_indices() (txes2.ElasticSearch method), 9
get_mapping() (txes2.ElasticSearch method), 9

I

index() (txes2.ElasticSearch method), 9

M

mget() (txes2.ElasticSearch method), 9
more_like_this() (txes2.ElasticSearch method), 9

O

open_index() (txes2.ElasticSearch method), 9
optimize() (txes2.ElasticSearch method), 9

P

partial_update() (txes2.ElasticSearch method), 9
put_mapping() (txes2.ElasticSearch method), 9

S

scan() (txes2.ElasticSearch method), 9
search() (txes2.ElasticSearch method), 9
servers (txes2.ElasticSearch attribute), 9
set_alias() (txes2.ElasticSearch method), 9
status() (txes2.ElasticSearch method), 9

U

update_settings() (txes2.ElasticSearch method), 9