elastalk Documentation

Release 6.5.2.2

patdaburu

Jul 27, 2019

CONTENTS:

Simple Conveniences for Talking to Elasticsearch

CHAPTER

ONE

GETTING STARTED

1.1 Elastic Stack

1.1.1 Elasticsearch

You don't need to install Elasticsearch to start using this library, but it won't be particularly useful until you start talking to a data store.

To get started with Elasticsearch, consult the Elasticsearch docs.

1.1.2 Kibana

You don't absolutely *need* Kibana to get started, but if you want a tool to help you inspect and visualize your data, Kibana is for you.

To get started with Kibana, consult the Kibana docs.

1.2 Installing the Library

You can use pip to install *elastalk*.

```
pip install elastalk
```

CHAPTER

TWO

CONFIGURING YOUR CONNECTION

2.1 Configuration from Objects

If you're wanting to configure your connection from a python object, you're likely using *Flask*. There is another article on that subject called *Configuration from Objects*.

2.2 Configuration from TOML

In addition to *configuring from objects*, you can also configure *elastalk* connections using TOML.

> TOML aims to be a minimal configuration file format that's easy to read due to obvious semantics. TOML is designed to map unambiguously to a hash table.
>
> —the TOML project's README.md

2.2.1 A Sample TOML Configuration

```
[blobs]
excluded = ["owner_", "group_"]

[indexes.cats]
mappings = "cats/mappings.json"

[indexes.dogs.blobs]
enabled = True
excluded = ["name", "breed"]
```

2.2.2 Options

seeds a list, or comma-separated string containing the Elasticsearch seed hosts

See also:

Elastalk.seeds

sniff_on_start See Sniffing on startup and *ElastalkConf.sniff_on_start*

sniff_on_connection_fail See Sniffing on connection failure and *ElastalkConf.sniff_on_connection_fail*

sniffer_timeout See Python Elasticsearch Client and *ElastalkConf.sniffer_timeout*

maxsize the maximum number of concurrent connections the client may make

See also:

Elastalk.maxsize

mapping_field_limit the maximum number of fields in an index

Note: Field and object mappings, as well as field aliases count towards this limit.

See also:

- *ElastalkConf.mapping_field_limit*
- Elasticsearch Mapping

CHAPTER

THREE

BLOBS

This section contains global configuration options that control how, when, and which data is converted to binary representations (see *Blobbing*).

enabled indicates whether or not blobbing is enabled

excluded the names of attributes that are never included in binary representations when a document is packed using the `ElastalkConnection.pack()` method

key the key that stores blobbed values in packed documents

CHAPTER

FOUR

INDEXES

This section contains information about specific Elasticsearch Indexes. In the *example* above there are two configured indexes: *cats* and *dogs*. You can configure individual index preferences by adding creating a new section and appending the index name to *indexes*.

blobs index-level blob configuration (See *blobs*.)

mappings a path to a file that contains an index mapping definition (See *Mappings*.)

CHAPTER

FIVE

SEEDING ELASTICSEARCH INDEXES

The `connect` module contains a convenience function called `seed` that you can use to initialize Elasticsearch indexes.

5.1 Directory Structure

When you call the `seed` function you only need to provide a path to the directory that contains your seed data, however the directory must conform to particular structure.

And example seed data directory structure in the project is shown below.

```
seed
|-- config.toml
`-- indexes
    |-- cats
    |   |-- cat
    |   |   |-- 5836327c-3592-4fcb-a925-14a106bdcdab
    |   |   `-- 9b31890a-28a1-4f59-a448-1f85dd2435a3
    |   `-- mappings.json
    `-- dogs
        `-- dog
            |-- 564e74ba-1177-4d3c-9160-a08e116ad9ff
            `-- de0a76e7-ecb9-4fac-b524-622ed8c344b8
```

5.1.1 The Base Directory (*"seed"*)

This is the base directory that contains all the seed data. If you're creating your own seed data set you may provide another name.

5.1.2 Indexes

All of the Elasticsearch indexes are defined in a subdirectory called *indexes*. An Elasticsearch index will be created for each subdirectory and the name of the subdirectory will be the name of the index.

5.1.3 Document Types

Within each *index* directory there are directories that define document types. The name of the subdirectory will be the name of the document type.

5.1.4 Documents

Within each *document type* directory are individual files that represent the individual documents that will be indexed. The name of the file will be the id of the document.

5.1.5 Extra Configuration

You can supply additional information about the seed data in an index by supplying a *config.toml* file in the *Indexes* directory.

Note: The `seed` function supports a parameter called *config* if, for some reason, you have a reason not to call your configuration files *"config.toml"*.

Mappings

If your index has a static mapping you can include a *mappings* key in the *index configuration file*. The value of this key should match what you would provide in the *mappings* if you were creating the index directly.

For example, if you would create the index by submitting the following *PUT* request to Elasticsearch...

```
PUT my_index
{
  "mappings": {
    "_doc": {
      "properties": {
        "title":    { "type": "text"  },
        "name":     { "type": "text"  },
        "age":      { "type": "integer" },
        "created":  {
          "type":   "date",
          "format": "strict_date_optional_time||epoch_millis"
        }
      }
    }
  }
}
```

...your configuration file should include a *mappings* key that looks like this...

```
{
  "mappings": {
    "_doc": {
      "properties": {
        "title": {
          "type": "text"
        },
        "name": {
          "type": "text"
        },
        "age": {
          "type": "integer"
        },
        "created": {
```

(continues on next page)

(continued from previous page)

```
            "type": "date",
            "format": "strict_date_optional_time||epoch_millis"
          }
        }
      }
    }
}
```

CHAPTER

SIX

"BLOBBING"

In order to minimize database overhead, some applications may want to store non-searchable document content in binary form in a field called *blob* so once a document has been indexed.

If you want to store some (or all) of your seed data as a single base-64 BLOB, you can add a *blobs* key to your *index configuration file*.

You can *configure* blobbing behavior in your `ElastalkConnection` via the `ElastalkConf`.

CHAPTER

SEVEN

ELASTALK AND *FLASK*

When we built this library we figured you just might want to use it in your Flask applications. With that in mind we've provided a few conveniences.

7.1 Configuration from Objects

Flask applications objects support a convention that allows you to configure the application using the name of a Python class via a method on the application's configuration object called from_object().

Sticking with that convention, the *ElastalkConnection* in this library has *config* attribute which returns an *ElastalkConf*. Like the Flask configuration object, this configuration object supports a method called *from_object* that mirrors the behavior of the Flask method.

7.1.1 Options

This section describes the configuration options you can use when configuring your Elasticsearch settings from an object.

ESHOSTS a Python list, or comma-separated string containing the Elasticsearch seed hosts

See also:

ElasticsearchConf.seeds

ES_SNIFF_ON_START See Sniffing on startup and *ElastalkConf.sniff_on_start*

ES_SNIFF_ON_CONNECTION_FAIL See Sniffing on connection failure and *ElastalkConf.sniff_on_connection_fail*

ES_SNIFFER_TIMEOUT See Python Elasticsearch Client and *ElastalkConf.sniffer_timeout*

ES_MAXSIZE the maximum number of concurrent connections the client may make

See also:

ElasticsearchConf.maxsize

ES_MAPPING_FIELD_LIMIT the maximum number of fields in an index

Note: Field and object mappings, as well as field aliases count towards this limit.

See also:

- *ElastalkConf.mapping_field_limit*
- Elasticsearch Mapping

CHAPTER

EIGHT

ELASTALK LIBRARY VERSIONS

The *elastalk* version numbers reflect the Elasticsearch with which they work. The *major* and *minor* version numbers reflect the Elasticsearch version, while the *patch* indicates a revision to the *elastalk* library.

For example, version *6.5.1* is the *first* version of the *elastalk* library (indicated by the patch version) built to work with version *6.5* of Elasticsearch.

gc_elasticsearch Version	Elasticsearch Version
6.5.1	6.5

CHAPTER

NINE

USING THE COMMAND LINE APPLICATION

This project contains a command line application (*elastalk*) based on Click.

9.1 Installation

The command line application is installed automatically when the package is installed.

9.2 Running the CLI in the Development Environment

If you need to run the application from within the project's own development environment, you can use the *make build* target.

```
make build
```

9.3 Getting Help

The command line application has a help function which you can access with the *–help* flag.

```
elastalk --help
```

CHAPTER

TEN

API DOCUMENTATION

Simple Conveniences for Talking to Elasticsearch

10.1 elastalk.config

Make things work the way you want!

class elastalk.config.**BlobConf**(*enabled: bool = None*, *excluded: Set[str] = <factory>*, *key: str = None*)

Bases: object

Define blobbing parameters.

__init__(*enabled: bool = None*, *excluded: Set[str] = <factory>*, *key: str = None*) → None

enabled = None
indicates whether or not blobbing is enabled.

exclude(**keys*)
Add to the set of excluded document keys.

Parameters **keys** – the excluded document keys

excluded = None
the excluded top-level document keys

key = None
the key that stores blobbed values in packed documents

classmethod load(*dict_: Dict*) → elastalk.config.BlobConf
Create an instance of the class from a dictionary.

Parameters **dict** – the dictionary

Returns the instance

class elastalk.config.**ElastalkConf**(*seeds: Iterable[str] = <factory>*, *sniff_on_start: bool = True*, *sniff_on_connection_fail: bool = True*, *sniffer_timeout: int = 60*, *maxsize: int = 10*, *mapping_field_limit: int = 1000*, *blobs: elastalk.config.BlobConf = BlobConf(enabled=None, excluded=set(), key=None)*, *indexes: Dict[str, elastalk.config.IndexConf] = <factory>*)

Bases: object

Configuration options for an Elastalk and the Elasticsearch client.

__init__(*seeds: Iterable[str] = <factory>*, *sniff_on_start: bool = True*, *sniff_on_connection_fail: bool = True*, *sniffer_timeout: int = 60*, *maxsize: int = 10*, *mapping_field_limit: int = 1000*, *blobs: elastalk.config.BlobConf = BlobConf(enabled=None, excluded=set(), key=None)*, *indexes: Dict[str, elastalk.config.IndexConf] = <factory>*) → None

blob_exclusions
Get the full set of top-level document properties that should be excluded from blobs for a given index. If you don't supply the *index* parameter, the method returns the global exclusions.

Parameters **index** – the name of the index

Returns the set of excluded property names

blob_key(*index: str = None*) → str
Get the configured document key for blobbed data. (If you don't supply the index, the method returns the global configuration value. If there is no global configuration value, the method returns the default.)

Parameters **index** – the name of the index

Returns the blobbed data key

blobs = BlobConf(enabled=None, excluded=set(), key=None)
global BLOB behavior configuration

blobs_enabled
Determine whether or not blobbing is enabled for an index.

Parameters **index** – the name of the index

Returns *True* if blobbing is enabled, otherwise *False*

from_object(*o: str*) → elastalk.config.ElastalkConf
Update the configuration from an object.

Parameters **o** – the configuration object

from_toml(*toml_: pathlib.Path*) → elastalk.config.ElastalkConf
Update the configuration from a TOML configuration.

Parameters **toml** – the path to the file or the TOML configuration string

indexes = None
index-specific configurations

mapping_field_limit = 1000
the maximum number of mapped fields

maxsize = 10
the maximum number of connections

seeds = None
the Elasticsearch seed hosts

sniff_on_connection_fail = True
Sniff when the connection fails?

sniff_on_start = True
Start sniffing on startup?

sniffer_timeout = 60
the sniffer timeout

exception elastalk.config.**ElastalkConfigException**
Bases: Exception

Raised when a configuration error is detected.

class elastalk.config.**IndexConf**(*blobs: elastalk.config.BlobConf = BlobConf(enabled=None, excluded=set(), key=None), mappings: str = None*)

Bases: object

Define index-specific configuration settings.

__init__(*blobs: elastalk.config.BlobConf = BlobConf(enabled=None, excluded=set(), key=None), mappings: str = None*) → None

blobs = BlobConf(enabled=None, excluded=set(), key=None)
blobbing configuration for the index

classmethod load(*dict_: Dict*) → elastalk.config.IndexConf
Create an instance of the class from a dictionary.

Parameters **dict** – the dictionary

Returns the instance

mappings = None
the path to Elasticsearch mappings for the configuration

mappings_document(*root: pathlib.Path = None*) → dict
Get the contents of the index mapping document (if one is defined).

Parameters **root** – the root path that contains the document file

Returns the index mapping document (or *None* if one isn't defined)

10.2 elastalk.connect

Start a conversation with Elasticsearch!

class elastalk.connect.**ElastalkConnection**(*config: elastalk.config.ElastalkConf = None*)
Bases: object

Defines an Elasticsearch environment.

__init__(*config: elastalk.config.ElastalkConf = None*)

Parameters **config** – the configuration

property client
Get the Elasticsearch client.

Returns the Elasticsearch client

Raises **ElasticsearchConfigurationException** – if there is an error in the current configuration

property config
Get the connection configuration.

Returns the connection configuration

static default(*cnx: Optional[elastalk.connect.ElastalkConnection] = None*) → elastalk.connect.ElastalkConnection
Set and/or retrieve the default connection object.

Parameters **cnx** – Provide a new connection object if you want to change the default. Otherwise, leave this argument out to retrieve the current object.

Returns the default connection object

pack(*doc: Dict*, *index: str*) → Dict[str, Any]
Convert a document object into a BLOB document.

Parameters

- **doc** – the original document object
- **index** – the name of the index *(This is optional, but if you supply it the behavior configured for the index can be used.)*

Returns the BLOB document

reset()
Reset the connection.

unpack(*doc: Dict*, *index: str = None*) → Dict
Convert a *packed* document to its original form.

Parameters

- **doc** – the packed document
- **index** – the name of the index for which the packed document came

Returns the unpacked document

class elastalk.connect.**ElastalkMixin**
Bases: object

Mix this into your class to get easy access to the Elasticsearch client.

property es
Get the Elasticsearch client.

property es_cnx
Get the Elastalk connection object.

Returns the Elastalk connection object

10.3 elastalk.seed

Prepare your Elasticsearch store with seed data!

elastalk.seed.**seed**(*root: str*, *config: str = 'config.toml'*, *force: bool = False*)
Populate an Elasticsearch instance with seed data.

Parameters

- **root** – the root directory that contains the seed data
- **config** – the path to the configuration
- **force** – delete existing indexes and replace them with seed data

Raises

- **FileNotFoundError** – if the path does not exist
- **NotADirectoryError** – if the path is not a directory

10.4 elastalk.version

This module contains project version information.

CHAPTER

ELEVEN

DEVELOPMENT

11.1 Getting Started

This section provides instructions for setting up your development environment. If you follow the steps from top to bottom you should be ready to roll by the end.

11.1.1 Get the Source

The source code for the *elastalk* project lives at github. You can use *git clone* to get it.

```
git clone https://github.com/patdaburu/elastalk
```

11.1.2 Create the Virtual Environment

You can create a virtual environment and install the project's dependencies using *make*.

```
make venv
make install
source venv/bin/activate
```

11.1.3 Try It Out

One way to test out the environment is to run the tests. You can do this with the *make test* target.

```
make test
```

If the tests run and pass, you're ready to roll.

11.1.4 Getting Answers

Once the environment is set up, you can perform a quick build of this project documentation using the *make answers* target.

```
make answers
```

11.2 Using the *Makefile*

This project includes a `Makefile` that you can use to perform common tasks such as running tests and building documentation.

11.2.1 Targets

This section contains a brief description of the targets defined in the `Makefile`.

`clean`

Remove generated packages, documentation, temporary files, *etc*.

`lint`

Run pylint against the project files.

`test`

Run the unit tests.

`docs`

Build the documentation for production.

`answers`

Perform a quick build of the documentation and open it in your browser.

`package`

Build the package for publishing.

`publish`

Publish the package to your repository.

`build`

Install the current project locally so that you may run the command-line application.

`venv`

Create a virtual environment.

install

Install (or update) project dependencies.

licenses

Generate a report of the projects dependencies and respective licenses.

Note: If project dependencies change, please update this documentation.

11.3 Publishing the Package

As you make changes to the project, you'll probably want to publish new version of the package. *(That's the point, right?)*

11.3.1 Publishing

The actual process of publishing the project is just a matter of running the *publish* target.

```
make publish
```

11.3.2 Installing

If you just need to install the library in your project, have a look at the *general tutorial* article.

CHAPTER

TWELVE

INDICES AND TABLES

- genindex
- modindex
- search

CHAPTER

THIRTEEN

PYTHON MODULE DEPENDENCIES

The `requirements.txt` file contains this project's module dependencies. You can install these dependencies using `pip`.

```
pip install -r requirements.txt
```

13.1 requirements.txt

```
click>=7.0,<8
dataclasses
elasticsearch>=6.3.1,<7
pip-check-reqs>=2.0.1,<3
pip-licenses>=1.7.1,<2
pylint>=1.8.4,<2
pytest>=3.4.0,<4
pytest-cov>=2.5.1,<3
pytest-pythonpath>=0.7.2,<1
setuptools>=38.4.0
Sphinx==1.7.2
sphinx-rtd-theme==0.3.0
toml>=0.10.0,<1
tox>=3.0.0,<4
twine>=1.11.0,<2
```

13.2 Runtime Dependencies and Licenses

Name	Version	License	URL
Click	7.0	BSD	https://palletsprojects.com/p/click/
elasticsearch	6.3.1	Apache License, Version 2.0	https://github.com/elastic/elasticsearch-py

CHAPTER

FOURTEEN

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

e

INDEX

Symbols

B

C

D

E

F

I

K

L

M

P

R

S

U