eldam Documentation

Release 1.2.0

Srikanth Bemineni

Feb 10, 2018

Contents

1 Some requirements kept in mind while designing this library **3**

2 Transaction process **5**
2.1 Two step commit 5

3 API Documentation **7**
3.1 `connect(settings,default_index = "")` 7
3.2 `connection()` 7
3.3 `refresh(index="_all")` 8
3.4 `add(item)` 8
3.5 `remove(item, check_existence=False)` 9
3.6 `update(item, check_existence=False)` 9
3.7 `update_by_query(item)` 10
3.8 `delete_by_query(item)` 11

Elasticsearch data manager (ELDAM) with zope transaction support. Other transaction data manager like SQLAlchemy use the database atomic operation feature to rollback if an error occurs during the commit process. This is not achievable in elastic search. To overcome this ELDAM finalizes the records in the commit call of the two-phase transaction process. At the same time takes backup of existing data in the index, if it involves removing or updating a record.

ELDAM now supports Elastic Search 6.0. One significant change in 6.0 is the lack for doc types in a single index. You can read about this here Removal of mapping types.

ELDAM will automatically detect this, if any of the nodes in the elastic search cluster is version 6.x and above.

CHAPTER 1

Some requirements kept in mind while designing this library

- This module is not atomic. If any of the other transactions fail, then there is a brief period where the data still resides in Elasticsearch and will be removed during the `tcp_abort` call.
- **Please use it at your own discretion if Elasticsearch is your primary data store**. This library is perfect if Elasticsearch is used only for searching in your application.

CHAPTER 2

Transaction process

`abort` - If needed, If any of the previous data managers aborted. This is before even beginning ELDAM data manager transaction process. Elasticsearch data manager doesn't do anything on this call. Since we haven't committed anything into ElasticSearch

`tpc_abort` - If any of the other managers voted no after Elasticsearch data manager has committed, then this function will be called for cleanup. We just revert all the data.

2.1 Two step commit

- `tpc_begin` - Prepare for the transaction. Elasticsearch does nothing in this call
- `commit` - Elasticsearch data manager commits the data, at the same time takes backup of the existing data in case of delete or update. Raises an exception if the any of the commits fail.
- `tpc_vote` - After committing, vote and tell the transaction manager, that I am okay to go or not. Elasticsearch data manager votes yes
- `tpc_finish` - Final commit, no turning back after this. Elasticsearch data manager has already committed the data.

CHAPTER 3

API Documentation

3.1 connect(settings,default_index = "")

Connect to ElasticSearch database

Parameter

- settings - Dictionary with elasticsearch connections details
  - 'elasticsearch_hosts' : array of Elasticsearch hosts [localhost, localhost:443, 16.74.45.322]
- default_index - Default index to store documents. If not specified, then add, remove and update input need to mention the index details. **In 6.x there should always be one mapping type in a single index**.

```
from eldam.elasticdatamanger import ElasticDataManager

edm = new ElasticDataManager()
edm.connect({
              "elasticsearch_hosts":["16.83.62.232:9200","16.83.63.114:9200"]
            },
            "test")
```

3.2 connection()

Get the stable connection to Elastic search.

```
from eldam.elasticdatamanger import ElasticDataManager

edm = new ElasticDataManager()
edm.connect({
              "elasticsearch_hosts":["16.83.62.232:9200","16.83.63.114:9200"]
            },
            "test")
```

```
# you can directly run api's defined from python ElasticSearch 5.0.0 and above
print(edm.connection.cluster.health())
```

3.3 refresh(index="_all")

Refreshes the indices. If we include a document, and immediately update or delete the document before the internal indexes are refreshed, then the operation can fail. refreshing the indexes will make sure that newly included records are available for modifying or deleting them.

Parameters

- index- Index to refresh, by default refreshes all indices.

3.4 add(item)

Add a document to be included into Elasticsearch once the transaction is committed. If this instance of data manager was not included into the transaction manager, then this call will automatically add it to the current transaction. **In 6.x ELDAM will automatically create the indexes, if it doesn't exist.** If the transaction fails, the index will be deleted, only if the index was established in the current transaction.

Parameters

- item - Dictionary of the document to be added
  - '_index' - Index to which this document needs to be included. This is optional if default_index is set during connect.
  - '_type' - Type of the document.
  - '_id' - Id of the document.
  - '_source' - Source of the document.

```
import transaction
from eldam.elasticdatamanger import ElasticDataManager


edm = ElasticDataManager()
edm.connect({
              "elasticsearch_hosts":["16.83.62.232:9200","16.83.63.114:9200"]
            },
            "test")
edm.add({'_type':'group',
         '_id':'2',
         '_source':{
                       "gid": 234,
                       "owner": "bemineni",
                       "name": "Sammy",
                       "grp_hash": "456678",
                       "description": "Sammy group"
                       }
         })
transaction.commit()
```

3.5 remove(item, check_existence=False)

Remove an existing document from Elasticsearch, once the transaction is committed. If this instance of data manager was not added to the transaction manager, then this call will automatically add it to the current transaction.

Parameters

- item - Dictionary of the document to be added
  - '_index' - Index to which this document needs to be included. This is optional if default_index is set during connect.
  - '_type' - Type of the document.
  - '_id' - Id of the document.
- check_existense - checks if the document indicated in the item exists, then adds into the transaction

```
import transaction
from eldam.elasticdatamanger import ElasticDataManager


edm = ElasticDataManager()
edm.connect({
              "elasticsearch_hosts":["16.83.62.232:9200","16.83.63.114:9200"]
            },
            "test")
edm.remove({'_type':'group',
                '_id':'2'})

# commit the transaction
transaction.commit()
```

3.6 update(item, check_existence=False)

Update an existing document from Elasticsearch once the transaction is committed. If the document doesn't exist in the index, then this call will automatically convert to an add call, and a new document will be included into ElasticSearch using the `_source` provided in the `item`. If this instance of data manager was not added to the transaction manager, then this call will automatically add it to the current transaction.

Parameters

- item - Dictionary of the document to be added
  - '_index' - Index to which this document needs to be included. This is optional if default_index is set during connect.
  - '_type' - Type of the document.
  - '_id' - Id of the document.
  - '_source' - Source of the document. Can be partial update attributes of the original document.
- check_existense - checks if the document indicated in the item exists, then adds into the transaction

```
import transaction
from eldam.elasticdatamanger import ElasticDataManager
```

```
edm = ElasticDataManager()
edm.connect({
              "elasticsearch_hosts":["16.83.62.232:9200","16.83.63.114:9200"]
            },
            "test")
edm.update({'_type':'group',
                '_id':'2',
                '_source':{
                            "description": "Sammy study group"
                            }
                })

# commit the transaction
transaction.commit()
```

3.7 update_by_query(item)

Update existing documents from Elasticsearch using a query and script once the transaction is committed. If this instance of data manager was not added to the transaction manager, then this call will automatically add it to the current transaction.

Parameters

- item - Dictionary of the document to be added
  - '_index' - Index to which this document needs to be included. This is optional if default_index is set during connect.
  - '_type' - Type of the document.
  - '_query' - Elasticsearch query using DSL format. https://www.elastic.co/guide/en/elasticsearch/reference/current/query-dsl.html
  - '_script' - Specify the script to update in any language, as long as the language is enabled in the deployed Elasticsearch. By default "painless" lang is supported in Elasticsearch

Below example searches for fields with id starting with 'g', and updates the description and name fields using the script provided. https://www.elastic.co/guide/en/elasticsearch/reference/current/modules-scripting.html

```
import transaction
from eldam.elasticdatamanger import ElasticDataManager


edm = ElasticDataManager()
edm.connect({
              "elasticsearch_hosts":["16.83.62.232:9200","16.83.63.114:9200"]
            },
            "test")
item = {}
item['query'] = {
                 "wildcard": {
                             "_uid": {
                                     "value":"g*"
                                     }
                             }
                }
```

```
item['script'] = {
                "inline":"ctx._source.description = params.description;ctx._source.
↪name = params.name",
                "lang":"painless",
                'params':{
                    "description": "eldam group",
                    "name":"eldam"
                }
            }
edm.update_by_query(item)

# commit the transaction
transaction.commit()

# if you need to access the data immediately, then refresh the indices
edm.connection.indices.refresh(index="_all")
```

3.8 delete_by_query(item)

Delete existing documents from Elasticsearch using a query once the transaction is committed. If this instance of data manager was not included in the transaction manager, then this call will automatically add it to the current transaction.

Parameters

- item - Dictionary of the document to be added
  - '_index' - Index to which this document needs to be included. This is optional if default_index is set during connect.
  - '_type' - Type of the document.
  - '_query' - Elasticsearch query using DSL format. https://www.elastic.co/guide/en/elasticsearch/reference/current/query-dsl.html

Below example deletes all the documents with an id starting with 'g'.

```
import transaction
from eldam.elasticdatamanger import ElasticDataManager


edm = ElasticDataManager()
edm.connect({
              "elasticsearch_hosts":["16.83.62.232:9200","16.83.63.114:9200"]
            },
            "test")
item = {}
item['query'] = {
                 "wildcard": {
                              "_uid": {
                                      "value":"g*"
                                      }
                              }
                }

edm.delete_by_query(item)
```

```
# commit the transaction
transaction.commit()
```