intake_elasticsearch Documentation

Release 0.0.1

Joseph Crail

Nov 26, 2019

CONTENTS:

This package enables the Intake data access and catalog system to read from the ElasticSearch data analytics platform.

CHAPTER ONE

QUICKSTART

`intake-elasticsearch` provides quick and easy access to data stored in ElasticSearch

This plugin reads ElasticSearch query results without random access: there is only ever a single partition.

1.1 Installation

To use this plugin for intake, install with the following command:

```
conda install -c intake intake-elasticsearch
```

1.2 Usage

1.2.1 Ad-hoc

After installation, the functions `intake.open_elasticsearch_table` and `intake.open_elasticsearch_seq` will become available. They can be used to execute queries on the ElasticSearch server, and download the results as a sequence of dictionaries, or a data-frame.

Three parameters are of interest when defining a data source:

- query: the query to execute, which can be defined either using Lucene or JSON syntax, both of which are to be provided as a string.
- qargs: further arguments to pass along with the query, such as the index(es) to consider, sorting and any filters to apply
- other arguments are passed as parameters to the server connection instance,

In the simplest case, this might look something like:

```
import intake
source = intake.open_elasticsearch_seq("*:*", host='elastic.server', port=9200,
    qargs={'index': 'mydocuments'})
result = source.read()
```

Where `"*:*"` is Lucene syntax for "match all", so this will grab every document within the given index, as a data-frame. The host and port parameters define the connection to the ElasticSearch server.

Further parameters which can be used to modify how the source works are as follows. These are likely not altered often.

- scroll: a text string specifying how long the query remains live on the server

- size: the number of entries to download in a single call; smaller numbers will download slower, but may be more stable.

1.2.2 Creating Catalog Entries

Catalog entries must specify `driver: elasticsearch_seq` for the sequence of dictionaries version, and `driver: elasticsearch_table` for the dataframe version.

Aside from this, the same arguments are available as for ad-hoc usage. Note that queries are commonly multi-line, especially is using JSON syntax, so the YAML `"|"` character should be used to define them within the catalog file. A full entry may look something like:

```
args:
  qargs:
    index: intake_test
    doc_type: entry
  query: |
      {
      "query": {
          "match":
              {"typeid": 1}
          },
      "sort": {
          "price": {"order": "desc"}
          },
      "_source": ["price", "typeid"]
      }
  host: intake_es
```

where we have specified both the index and document types (these could have been lists), the fields to extract and sort order, as well as a matching term, loosely equivalent to `"WHERE typeid = 1"` in SQL.

1.2.3 Using a Catalog

Assuming a catalog file `'cat.yaml'`, and an entry called `'es_data'`, the corresponding dataframe could be fetched as follows:

```
import intake
cat = intake.Catalog('cat.yaml')
result = cat.es_data.read()
```

Since the query cannot be partitioned with this plugin, the other methods of the data source (iterate, read one partition, create Dask data-frame) are not particularly useful here.

CHAPTER

TWO

API REFERENCE

intake_elasticsearch.elasticsearch_table.ElasticSearchTableSource(...)	Data source which executes arbitrary queries on ElasticSearch
intake_elasticsearch.elasticsearch_seq.ElasticSearchSeqSource(query)	Data source which executes arbitrary queries on ElasticSearch

class `intake_elasticsearch.elasticsearch_table.`**`ElasticSearchTableSource`**(**args*, ***kwargs*)

Data source which executes arbitrary queries on ElasticSearch

This is the tabular reader: will return dataframes. Nested return items will become dict-like objects in the output.

Parameters

query: str Query to execute. Can either be in Lucene single-line format, or a JSON structured query (presented as text)

npartitions: int Split query into this many sections. If one, will not split.

qargs: dict Further parameters to pass to the query, such as set of indexes to consider, filtering, ordering. See http://elasticsearch-py.readthedocs.io/en/master/api.html#elasticsearch.Elasticsearch.search

es_kwargs: dict Settings for the ES connection, e.g., a simple local connection may be `{'host': 'localhost', 'port': 9200}`. Other keywords to the Plugin that end up here and are material:

scroll: str how long the query is live for, default `'100m'`

size: int the paging size when downloading, default 1000.

metadata: dict Extra information for this source.

Attributes

cache_dirs

classname

datashape

description

has_been_persisted

`hvplot` Returns a hvPlot object to provide a high-level plotting API.

is_persisted

`plot` Returns a hvPlot object to provide a high-level plotting API.

`plots` List custom associated quick-plots

Methods

`close`(self)	Close open resources corresponding to this data source.
`discover`(self)	Open resource and populate the source attributes.
`export`(self, path, **kwargs)	Save this data for sharing with other people
`persist`(self[, ttl])	Save data from this source to local persistent storage
`read`(self)	Read all data in one go
`read_chunked`(self)	Return iterator over container fragments of data source
`read_partition`(self, i)	Return a part of the data corresponding to i-th partition.
`to_dask`(self)	Turn into dask.dataframe
`to_spark`(self)	Provide an equivalent data object in Apache Spark
`yaml`(self[, with_plugin])	Return YAML representation of this data-source

get_persisted	
set_cache_dir	

`to_dask`(*self*)
Turn into dask.dataframe

`class` `intake_elasticsearch.elasticsearch_seq.`**`ElasticSearchSeqSource`**(*query*, *npartitions=1*, *qargs={}*, *metadata={}*, ***es_kwargs*)

Data source which executes arbitrary queries on ElasticSearch

This is the sequential reader: will return a list of dictionaries.

Parameters

query: str Query to execute. Can either be in Lucene single-line format, or a JSON structured query (presented as text)

npartitions: int Split query into this many sections. If one, will not split.

qargs: dict Further parameters to pass to the query, such as set of indexes to consider, filtering, ordering. See http://elasticsearch-py.readthedocs.io/en/master/api.html#elasticsearch.Elasticsearch.search

es_kwargs: dict Settings for the ES connection, e.g., a simple local connection may be `{'host': 'localhost', 'port': 9200}`. Other keywords to the Plugin that end up here and are material:

scroll: str how long the query is live for, default `'100m'`

size: int the paging size when downloading, default 1000.

metadata: dict Extra information for this source.

Attributes

cache_dirs

classname

datashape

description

has_been_persisted

hvplot Returns a hvPlot object to provide a high-level plotting API.

is_persisted

plot Returns a hvPlot object to provide a high-level plotting API.

plots List custom associated quick-plots

Methods

`close`(self)	Close open resources corresponding to this data source.
`discover`(self)	Open resource and populate the source attributes.
`export`(self, path, **kwargs)	Save this data for sharing with other people
`persist`(self[, ttl])	Save data from this source to local persistent storage
`read`(self)	Read all data in one go
`read_chunked`(self)	Return iterator over container fragments of data source
`read_partition`(self, i)	Return a part of the data corresponding to i-th partition.
`to_dask`(self)	Form partitions into a dask.bag
`to_spark`(self)	Provide an equivalent data object in Apache Spark
`yaml`(self[, with_plugin])	Return YAML representation of this data-source

get_persisted	
set_cache_dir	

read(*self*)
Read all data in one go

to_dask(*self*)
Form partitions into a dask.bag

CHAPTER THREE

INDICES AND TABLES

- genindex
- modindex
- search

INDEX