Cockatrice Documentation

Release 0.1.0

Minoru Osuka

Oct 11, 2018

Contents:

1 What's Cockatrice? 1

2 Source Codes 3

3 Requirements 5

4 Features 7

4.1 Setup 7

4.2 Usage 7

4.3 Schema 12

5 Indices and tables 13

CHAPTER 1

What's Cockatrice?

Cockatrice is the open source search and indexing server written in Python that provides scalable indexing and search, faceting, hit highlighting and advanced analysis/tokenization capabilities.

Indexing and search are implemented by Whoosh. Cockatrice provides it via the RESTful API using Flask.

In cluster mode, Cockatrice uses Raft Consensus Algorithm by PySyncObj to achieve consensus across all the instances of the nodes, ensuring that every change made to the system is made to a quorum of nodes.

CHAPTER 2

Source Codes

https://github.com/mosuka/cockatrice

CHAPTER 3

Requirements

Python 3.x interpreter

CHAPTER 4

Features

- Full-text search and indexing
- Faceting
- Result highlighting
- Easy deployment
- Bringing up cluster
- Index replication
- An easy-to-use RESTful API

4.1 Setup

4.1.1 Setting up Cockatrice

Cockatrice is not registered to PyPI just yet, so you may not install it via pip command at the moment.

```
$ git clone https://github.com/mosuka/cockatrice.git
$ cd cockatrice
$ pip install -e .
```

4.2 Usage

4.2.1 Running Cockatrice in standalone mode

Running a Cockatrice node is easy. Starting Cockatrice in standalone mode by the following command:

```
$ ./bin/cockatrice server --http-port=8080 --index-dir=/tmp/cockatrice/index --schema-
↪file=./conf/schema.yaml
```

You can now index, get, search and delete the document(s) via HTTP.

4.2.2 Indexing a document

Indexing a document by the following command:

```
$ curl -s -X PUT -H "Content-Type:application/json" http://localhost:8080/rest/doc/1 -
↪d @./example/doc1.json | jq .
```

You can see the result in JSON format. The result of the above command is:

```
{
  "status": {
    "code": 202,
    "description": "Request accepted, processing continues off-line",
    "phrase": "Accepted"
  },
  "time": 0.00015020370483398438
}
```

4.2.3 Getting a document

Getting a document by the following command:

```
$ curl -s -X GET http://localhost:8080/rest/doc/1 | jq .
```

You can see the result in JSON format. The result of the above command is:

```
{
  "doc": {
    "fields": {
      "contributor": "43.225.167.166",
      "id": "1",
      "text": "A search engine is an information retrieval system designed to help␣
↪find information stored on a computer system. The search results are usually␣
↪presented in a list and are commonly called hits. Search engines help to minimize␣
↪the time required to find information and the amount of information which must be␣
↪consulted, akin to other techniques for managing information overload. The most␣
↪public, visible form of a search engine is a Web search engine which searches for␣
↪information on the World Wide Web.",
      "timestamp": "20180704054100",
      "title": "Search engine (computing)"
    }
  },
  "status": {
    "code": 200,
    "description": "Request fulfilled, document follows",
    "phrase": "OK"
  },
  "time": 0.011947870254516602
}
```

4.2.4 Deleting a document

Deleting a document by the following command:

```
$ curl -s -X DELETE http://localhost:8080/rest/doc/1 | jq .
```

You can see the result in JSON format. The result of the above command is:

```
{
  "status": {
    "code": 202,
    "description": "Request accepted, processing continues off-line",
    "phrase": "Accepted"
  },
  "time": 6.699562072753906e-05
}
```

4.2.5 Indexing documents in bulk

Indexing documents in bulk by the following command:

```
$ curl -s -X PUT -H "Content-Type:application/json" http://localhost:8080/rest/bulk -
↪d @./example/bulk_index.json | jq .
```

You can see the result in JSON format. The result of the above command is:

```
{
  "status": {
    "code": 202,
    "description": "Request accepted, processing continues off-line",
    "phrase": "Accepted"
  },
  "time": 0.00018596649169921875
}
```

4.2.6 Searching documents

Searching documents by the following command:

```
$ curl -s -X GET http://localhost:8080/rest/search?query=search | jq .
```

You can see the result in JSON format. The result of the above command is:

```
{
  "results": {
    "hits": [
      {
        "doc": {
          "fields": {
            "contributor": "KolbertBot",
            "id": "3",
            "text": "Enterprise search is the practice of making content from␣
↪multiple enterprise-type sources, such as databases and intranets, searchable to a␣
↪defined audience. \"Enterprise search\" is used to describe the software of search␣
↪information within an enterprise (though the search function and its results may␣
↪still be public). Enterprise search can be contrasted with web search, which␣
↪applies search technology to documents on the open web, and desktop search, which␣
↪applies search technology to the content on a single computer. Enterprise search␣
↪systems index data and documents from a variety of sources such as: file systems,␣
↪intranets, document management systems, e-mail, and databases. Many enterprise␣
↪search systems integrate structured and unstructured data in their collections.[3]␣
↪Enterprise search systems also use access controls to enforce a security policy on␣
```

(continues on next page)

(continued from previous page)

```
            "timestamp": "20180129125400",
            "title": "Enterprise search"
          }
        },
        "pos": 0,
        "rank": 0,
        "score": 1.7234593504967473
      },
      {
        "doc": {
          "fields": {
            "contributor": "Nurg",
            "id": "5",
            "text": "Federated search is an information retrieval technology that␣
↪allows the simultaneous search of multiple searchable resources. A user makes a␣
↪single query request which is distributed to the search engines, databases or other␣
↪query engines participating in the federation. The federated search then aggregates␣
↪the results that are received from the search engines for presentation to the user.␣
↪Federated search can be used to integrate disparate information resources within a␣
↪single large organization (\"enterprise\") or for the entire web. Federated search,␣
↪unlike distributed search, requires centralized coordination of the searchable␣
↪resources. This involves both coordination of the queries transmitted to the␣
↪individual search engines and fusion of the search results returned by each of them.
↪",
            "timestamp": "20180716000600",
            "title": "Federated search"
          }
        },
        "pos": 1,
        "rank": 1,
        "score": 1.7042117821338238
      },
      {
        "doc": {
          "fields": {
            "contributor": "Aistoff",
            "id": "2",
            "text": "A web search engine is a software system that is designed to␣
↪search for information on the World Wide Web. The search results are generally␣
↪presented in a line of results often referred to as search engine results pages␣
↪(SERPs). The information may be a mix of web pages, images, and other types of␣
↪files. Some search engines also mine data available in databases or open␣
↪directories. Unlike web directories, which are maintained only by human editors,␣
↪search engines also maintain real-time information by running an algorithm on a web␣
↪crawler. Internet content that is not capable of being searched by a web search␣
↪engine is generally described as the deep web.",
            "timestamp": "20181005132100",
            "title": "Web search engine"
          }
        },
        "pos": 2,
        "rank": 2,
        "score": 1.619574615564863
      },
      {
        "doc": {
          "fields": {
```

(continues on next page)

(continued from previous page)

```
            "contributor": "43.225.167.166",
            "id": "1",
            "text": "A search engine is an information retrieval system designed to␣
↪help find information stored on a computer system. The search results are usually␣
↪presented in a list and are commonly called hits. Search engines help to minimize␣
↪the time required to find information and the amount of information which must be␣
↪consulted, akin to other techniques for managing information overload. The most␣
↪public, visible form of a search engine is a Web search engine which searches for␣
↪information on the World Wide Web.",
            "timestamp": "20180704054100",
            "title": "Search engine (computing)"
          }
        },
        "pos": 3,
        "rank": 3,
        "score": 1.5951006619362313
      },
      {
        "doc": {
          "fields": {
            "contributor": "Citation bot",
            "id": "4",
            "text": "A distributed search engine is a search engine where there is no␣
↪central server. Unlike traditional centralized search engines, work such as␣
↪crawling, data mining, indexing, and query processing is distributed among several␣
↪peers in a decentralized manner where there is no single point of control.",
            "timestamp": "20180930171400",
            "title": "Distributed search engine"
          }
        },
        "pos": 4,
        "rank": 4,
        "score": 1.5232201764110038
      }
    ],
    "is_last_page": true,
    "page_count": 1,
    "page_len": 5,
    "page_num": 1,
    "total": 5
  },
  "status": {
    "code": 200,
    "description": "Request fulfilled, document follows",
    "phrase": "OK"
  },
  "time": 0.010915756225585938
}
```

4.2.7 Deleting documents in bulk

Deleting documents in bulk by the following command:

```
$ curl -s -X DELETE -H "Content-Type:application/json" http://localhost:8080/rest/
↪bulk -d @./example/bulk_delete.json | jq .
```

You can see the result in JSON format. The result of the above command is:

```
{
  "status": {
    "code": 202,
    "description": "Request accepted, processing continues off-line",
    "phrase": "Accepted"
  },
  "time": 0.00232696533203125
}
```

4.3 Schema

4.3.1 Schema Design

CHAPTER 5

Indices and tables

- genindex
- modindex
- search