無料電子ブック

学習

Elasticsearch

Free unaffiliated eBook created from
Stack Overflow contributors.

#elasticsear

ch

1: Elasticsearchをいめる

Elasticsearchは、Luceneにづいて Javaでかれたなオープンソースサーバーです。

これは、HTTP REST APIをしてアクセスな、テキスト、なテキスト、クエリベースおよびジオロケーションベースのをしてします。

バージョン

バージョン	
5.2.1	2017-02-14
5.2.0	2017-01-31
5.1.2	2017-01-12
5.1.1	2016-12-08
5.0.2	2016-11-29
5.0.1	2016-11-15
5.0.0	2016-10-26
2.4.0	2016-08-31
2.3.0	2016-03-30
2.2.0	2016-02-02
2.1.0	2015-11-24
2.0.0	2015-10-28
1.7.0	2015-07-16
1.6.0	2015-06-09
1.5.0	2015-03-06
1.4.0	2014-11-05
1.3.0	2014-07-23
1.2.0	2014-05-22
1.1.0	2014-03-25

バージョン	
1.0.0	2014-02-14

Examples

Ubuntu 14.04でのElasticsearchのインストール

Elasticsearchをするには、マシンにJava Runtime EnvironmentJREがです。 Elasticsearchには Java 7があり、 `Oracle JDK version 1.8.0_73`をし`Oracle JDK version 1.8.0_73`。

Oracle Java 8をインストールする

```
sudo add-apt-repository -y ppa:webupd8team/java
sudo apt-get update
echo "oracle-java8-installer shared/accepted-oracle-license-v1-1 select true" | sudo debconf-
set-selections
sudo apt-get install -y oracle-java8-installer
```

Javaバージョンをする

```
java -version
```

パッケージのダウンロードとインストール

バイナリの

1. Elasticsearchののをここからダウンロードしてください 。
2. ファイルをしてする

Linux

```
$ bin/elasticsearch
```

apt-getをう

ウェブサイトからelasticsearchをダウンロードするわりに、 `apt-get`をしてそれをインストールし`apt-get`。

```
wget -qO - https://packages.elastic.co/GPG-KEY-elasticsearch | sudo apt-key add -
echo "deb https://packages.elastic.co/elasticsearch/2.x/debian stable main" | sudo tee -a
/etc/apt/sources.list.d/elasticsearch-2.x.list
sudo apt-get update && sudo apt-get install elasticsearch
sudo /etc/init.d/elasticsearch start
```

elasticsearchバージョン5.xのインストール

```
wget -qO - https://artifacts.elastic.co/GPG-KEY-elasticsearch | sudo apt-key add -
sudo apt-get install apt-transport-https
echo "deb https://artifacts.elastic.co/packages/5.x/apt stable main" | sudo tee -a
/etc/apt/sources.list.d/elastic-5.x.list
sudo apt-get update && sudo apt-get install elasticsearch
```

Linuxでサービスとして

インストール、はしません。サービスとしてするがあります。 Elasticsearchをまたはするは、システムがSysV initをするかsystemdをするかによってなります。のコマンドでできます。

```
ps -p 1
```

あなたのディストリビューションがSysV initをしているは、をするがあります

```
sudo update-rc.d elasticsearch defaults 95 10
sudo /etc/init.d/elasticsearch start
```

それのは、あなたのがsystemdをしています

```
sudo /bin/systemctl daemon-reload
sudo /bin/systemctl enable elasticsearch.service
```

ブラウザまたはRESTクライアントから`CURL`コマンドをして、Elasticsearchがしくインストールされているかどうかをします。

```
curl -X GET http://localhost:9200/
```

WindowsでのElasticsearchのインストール

WindowsのElasticsearchは、このリンクからできます https ://www.elastic.co/downloads/elasticsearch 。のリリースはににあります。

Windowsにインストールするときには、 `.ZIP`アーカイブがです。 [`Downloads:`セクションのリンクをクリックし、ファイルをコンピュータにします。

このバージョンのは「ポータブル」です。つまり、プログラムをするためにインストーラをするはありません。ファイルのをにえているにします。デモンストレーションのために、すべてを`C:\elasticsearch`したとします。

アーカイブにはデフォルトで`elasticsearch-<version>`というのフォルダがあり、そのフォルダを`C:\`にして`elasticsearch`をするか、 `C:\elasticsearch`でしてアーカイブのフォルダののみをすることができますそこへ。

ElasticsearchはJavaでかれているため、Java Runtime Environmentがするがあります。したがって、サーバーをするに、コマンド・プロンプトをいてのようにして、Javaがかどうかをしてください。

```
java -version
```

のようながられるはずです。

```
java version "1.8.0_91"
Java(TM) SE Runtime Environment (build 1.8.0_91-b14)
Java HotSpot(TM) Client VM (build 25.91-b14, mixed mode)
```

わりにのようにされた

> 'java'は、またはのコマンド、なプログラムまたはバッチファイルとしてされません。

Javaがシステムにインストールされていないか、またはしくされていません。 このチュートリアルにうと、 Javaをインストールできます。また、これらのがのにされていることをしてください。

JAVA_HOME	C\ Program Files \ Java \ jre
パス	...; C\ Program Files \ Java \ jre

これらのをべるがわからないは、 このチュートリアルをしてください 。

バッチファイルから

Javaをインストールしたら、 `bin`フォルダをきます。すべてをしたフォルダでつけることができるので、 `c:\elasticsearch\bin`はずです。このフォルダには、コマンドウィンドウでElasticsearchをするためにできる`elasticsearch.bat`というファイルがあります。これは、プロセスによってされたがコマンドプロンプトウィンドウにされることをします。サーバーをするには、 `CTRL C`をすか、にウィンドウをじます。

Windowsサービスとしてする

には、にできないなウィンドウをつことはましくないため、Elasticsearchをサービスとしてするようにすることができます。

Elasticsearchをサービスとしてインストールするに、 `C:\elasticsearch\config\jvm.options`ファイルにをするがあります。

> サービスインストーラは、サービスをインストールするに、スレッドスタックサイズ

のを`jvm.options`するがあります。 32ビットWindowsでは`-Xss320k` [...]をし、64ビットWindowsでは`-Xss1m`を`jvm.options`ファイルにする`-Xss1m`があります。 [ソース]

をえたら、コマンドプロンプトをき、のコマンドをして`bin`ディレクトリにします。

```
C:\Users\user> cd c:\elasticsearch\bin
```

サービスは`elasticsearch-service.bat`によってされます。いバージョンでは、このファイルはに`service.bat`とばれることがあります。なすべてのをするには、をします。

```
C:\elasticsearch\bin> elasticsearch-service.bat

Usage: elasticsearch-service.bat install|remove|start|stop|manager [SERVICE_ID]
```

また、オプションの`SERVICE_ID`があることがわかりますが、はできます。サービスをインストールするには、のコマンドをします。

```
C:\elasticsearch\bin> elasticsearch-service.bat install
```

サービスをインストールした、それぞれのをしてサービスをおよびできます。サービスをするには、

```
C:\elasticsearch\bin> elasticsearch-service.bat start
```

それをするには、する

```
C:\elasticsearch\bin> elasticsearch-service.bat stop
```

わりにGUIをしてサービスをするは、のコマンドをできます。

```
C:\elasticsearch\bin> elasticsearch-service.bat manager
```

これによりElastic Service Managerがきます。これにより、サービスのをカスタマイズし、のタブのにあるボタンをしてサービスを/することができます。

ドキュメントのけと

ElasticsearchはHTTP REST APIをしてアクセスされます。はcURLライブラリをします。サーバーとクライアントユーザーまたはアプリケーションとののメッセージは、JSONのでされます。デフォルトでは、Elasticsearchはポート9200でします。

のでは、ElasticsearchにJSONレスポンスをあらかじめするようにするために`?pretty`がされています。アプリケーションでこれらのエンドポイントをするは、このクエリパラメータをするはありません。

ドキュメントのインデックス

でのをするは、けするにのIDをりてることをおめします。 `megacorp`というのにをするには、`employee`とID `1`タイプをします。

```
curl -XPUT "http://localhost:9200/megacorp/employee/1?pretty" -d'
{
    "first_name" : "John",
    "last_name" :  "Smith",
    "age" :        25,
    "about" :      "I love to go rock climbing",
    "interests": [ "sports", "music" ]
}'
```

```
{
  "_index": "megacorp",
  "_type": "employee",
  "_id": "1",
  "_version": 1,
  "_shards": {
    "total": 2,
    "successful": 1,
    "failed": 0
  },
  "created": true
}
```

インデックスは、PUTびしをするときにしないにされます。

IDなしのインデックス

```
POST /megacorp/employee?pretty
{
    "first_name" :  "Jane",
    "last_name" :   "Smith",
    "age" :         32,
    "about" :       "I like to collect rock albums",
    "interests":  [ "music" ]
}
```

```
{
  "_index": "megacorp",
  "_type": "employee",
  "_id": "AVYg2mBJYy9ijdngfeGa",
  "_version": 1,
  "_shards": {
    "total": 2,
    "successful": 2,
    "failed": 0
  },
  "created": true
}
```

ドキュメントの

```
curl -XGET "http://localhost:9200/megacorp/employee/1?pretty"
```

```
{
  "_index": "megacorp",
  "_type": "employee",
  "_id": "1",
  "_version": 1,
  "found": true,
  "_source": {
    "first_name": "John",
    "last_name": "Smith",
    "age": 25,
    "about": "I love to go rock climbing",
    "interests": [
      "sports",
      "music"
    ]
  }
}
```

`megacorp`インデックスから`employee`というの10のドキュメントをします。

```
curl -XGET "http://localhost:9200/megacorp/employee/_search?pretty"
```

```
{
  "took": 2,
  "timed_out": false,
  "_shards": {
    "total": 5,
    "successful": 5,
    "failed": 0
  },
  "hits": {
    "total": 2,
    "max_score": 1,
    "hits": [
      {
        "_index": "megacorp",
        "_type": "employee",
        "_id": "1",
        "_score": 1,
        "_source": {
          "first_name": "John",
          "last_name": "Smith",
          "age": 25,
          "about": "I love to go rock climbing",
          "interests": [
            "sports",
            "music"
```

```
          ]
        }
      },
      {
        "_index": "megacorp",
        "_type": "employee",
        "_id": "AVYg2mBJYy9ijdngfeGa",
        "_score": 1,
        "_source": {
          "first_name": "Jane",
          "last_name": "Smith",
          "age": 32,
          "about": "I like to collect rock albums",
          "interests": [
            "music"
          ]
        }
      }
    ]
  }
}
```

されたフィールドでにするものをす`match`クエリをしたな

```
curl -XGET "http://localhost:9200/megacorp/employee/_search" -d'
{
    "query" : {
        "match" : {
            "last_name" : "Smith"
        }
    }
}'
```

```
{
  "took": 2,
  "timed_out": false,
  "_shards": {
    "total": 5,
    "successful": 5,
    "failed": 0
  },
  "hits": {
    "total": 1,
    "max_score": 0.6931472,
    "hits": [
      {
        "_index": "megacorp",
        "_type": "employee",
        "_id": "1",
        "_score": 0.6931472,
        "_source": {
          "first_name": "John",
          "last_name": "Smith",
          "age": 25,
          "about": "I love to go rock climbing",
          "interests": [
            "sports",
            "music"
```

```
          ]
        }
      }
    ]
  }
}
```

をいたパラメータ

では、すべてののとしてなインデックスきドキュメントがされます。これは、ソースヒットの_sourceフィールドとばれます。ソースをすことをまない、されるソースのいくつかのフィールドだけをするか、_sourceをfalseにしてフィールドをにすることができます。

のは、account_numberとbalance _sourceの2つのフィールドをaccount_numberからすをしています。

```
curl -XPOST 'localhost:9200/bank/_search?pretty' -d '
{
  "query": { "match_all": {} },
  "_source": ["account_number", "balance"]
}'
```

のは、に_sourceフィールドでされるをらすことにしてください。_sourceという1つのフィールドだけをしますが、account_numberフィールドとbalanceフィールドのみがまれます。

あなたがSQLのバックグラウンドからたのであれば、はSQLのクエリとがています

```
SELECT account_number, balance FROM bank;
```

では、クエリにりましょう。は、match_allクエリをしてすべてのドキュメントをするをてきました。ここでは、マッチクエリとばれるしいクエリをします。これは、なフィールドクエリつまり、のフィールドまたはフィールドセットにしてわれたとえることができます。

このでは、account_numberが20されたアカウントがされます。

```
curl -XPOST 'localhost:9200/bank/_search?pretty' -d '
{
  "query": { "match": { "account_number": 20 } }
}'
```

このでは、address「ミル」というをむすべてのをしaddress。

```
curl -XPOST 'localhost:9200/bank/_search?pretty' -d '
{
  "query": { "match": { "address": "mill" } }
}'
```

このでは、address「mill」または「lane」というをむすべてのアカウントをしaddress。

```
curl -XPOST 'localhost:9200/bank/_search?pretty' -d '
{
  "query": { "match": { "address": "mill lane" } }
}'
```

このは、クエリをにし、おいにじにある`address`のすべてのをむドキュメントのみをす`match` `match_phrase` のです[1] 。

```
curl -XPOST 'localhost:9200/bank/_search?pretty' -d '
{
  "query": { "match_phrase": { "address": "mill lane" } }
}'
```

booleanクエリをしましょう。 boolクエリをすると、ブーールをして、よりさいクエリをよりきなクエリにすることができます。

このでは、2つのクエリをし、アドレスに「mill」と「lane」をむすべてのアカウントをします。

```
curl -XPOST 'localhost:9200/bank/_search?pretty' -d '
{
  "query": {
    "bool": {
      "must": [
        { "match": { "address": "mill" } },
        { "match": { "address": "lane" } }
      ]
    }
  }
}'
```

のでは、bool `must`は、ドキュメントがとみなされるためにでなければならないすべてのクエリをします。

に、このでは、2つのクエリをし、 `address` 「mill」または「lane」をむすべてのアカウントをし`address`。

```
curl -XPOST 'localhost:9200/bank/_search?pretty' -d '
{
  "query": {
    "bool": {
      "should": [
        { "match": { "address": "mill" } },
        { "match": { "address": "lane" } }
      ]
    }
  }
}'
```

のでは、bool `should`は、がするとみなされるためにはでなければならないせのリストをします。

このでは、2つのクエリをし、 `address` 「ミル」も「レーン」もまないすべてのアカウントをし`address`。

```
curl -XPOST 'localhost:9200/bank/_search?pretty' -d '
{
  "query": {
    "bool": {
      "must_not": [
        { "match": { "address": "mill" } },
        { "match": { "address": "lane" } }
      ]
    }
  }
}'
```

のでは、bool must_notは、ドキュメントがとみなされるためにはでなければならないクエリのリストをします。

boolクエリのでmust、should、must_notをにみわせることができます。さらに、これらのブーのいずれかのでブールクエリをして、なマルチレベルブールをすることができます。

このでは、に40で、ワシントンにんでいないにするすべてのアカウントをします `WA`はい。

```
curl -XPOST 'localhost:9200/bank/_search?pretty' -d '
{
  "query": {
    "bool": {
      "must": [
        { "match": { "age": "40" } }
      ],
      "must_not": [
        { "match": { "state": "WA" } }
      ]
    }
  }
}'
```

CentOS 7にElasticsearchとKibanaをインストールする

Elasticsearchをするには、マシンにJava Runtime EnvironmentJREがです。 Elasticsearchには Java 7があり、 `Oracle JDK version 1.8.0_73`をし`Oracle JDK version 1.8.0_73`。

したがって、システムにJavaがあるかどうかをしてください。そうでないは、のにいます。

```
# Install wget with yum
yum -y install wget

# Download the rpm jre-8u60-linux-x64.rpm for 64 bit
wget --no-cookies --no-check-certificate --header "Cookie:
gpw_e24=http%3A%2F%2Fwww.oracle.com%2F; oraclelicense=accept-securebackup-cookie"
"http://download.oracle.com/otn-pub/java/jdk/8u60-b27/jre-8u60-linux-x64.rpm"

# Download the rpm jre-8u101-linux-i586.rpm for 32 bit
wget --no-cookies --no-check-certificate --header "Cookie:
gpw_e24=http%3A%2F%2Fwww.oracle.com%2F; oraclelicense=accept-securebackup-cookie"
"http://download.oracle.com/otn-pub/java/jdk/8u101-b13/jre-8u101-linux-i586.rpm"

# Install jre-.*.rpm
```

```
rpm -ivh jre-.*.rpm
```

JavaはCentOSシステムにインストールするがあります。あなたはそれをチェックすることができます

```
java -version
```

elasticsearchのダウンロードとインストール

```
# Download elasticsearch-2.3.5.rpm
wget
https://download.elastic.co/elasticsearch/release/org/elasticsearch/distribution/rpm/elasticsearch/2.3.
2.3.5.rpm

# Install elasticsearch-.*.rpm
rpm -ivh elasticsearch-.*.rpm
```

にシステムサービスとして**elasticsearch**をする

```
sudo systemctl daemon-reload
sudo systemctl enable elasticsearch
sudo systemctl start elasticsearch

# check the current status to ensure everything is okay.
systemctl status elasticsearch
```

をする

rpmでにGPGキーをインポートする

```
sudo rpm --import http://packages.elastic.co/GPG-KEY-elasticsearch
```

に、ローカルリポジトリ`kibana.repo`します`kibana.repo`

```
sudo vi /etc/yum.repos.d/kibana.repo
```

そしてのコンテンツをします

```
[kibana-4.4]
name=Kibana repository for 4.4.x packages
baseurl=http://packages.elastic.co/kibana/4.4/centos
gpgcheck=1
gpgkey=http://packages.elastic.co/GPG-KEY-elasticsearch
enabled=1
```

のコマンドでkibanaをインストールします。

```
yum -y install kibana
```

それをめる

```
systemctl start kibana
```

ステータスをするには

```
systemctl status kibana
```

スタートアップサービスとしてすることができます。

```
systemctl enable kibana
```

オンラインでElasticsearchをいめるをむ
https://riptutorial.com/ja/elasticsearch/topic/941/elasticsearchをいめる

2: Python インタフェース

パラメータ

パラメータ	
ホスト	キーhostとportをむオブジェクトののホストの。デフォルトのhostは 'localhost'で、 portは9200です。 サンプルエントリは`[{"host": "ip of es server", "port": 9200}]`
sniff_on_start	ブールクライアントがにノードをスニッフィングするようにする、スニッフィングはelasticsearchクラスタのノードのリストをすることをします
sniff_on_connection_fail	クライアントがアクティブなときにがしたのスニッフィングをトリガーするブール
sniffer_timeout	スニフの
sniff_timeout	スニッフィングの1ののための
retry_on_timeout	Booelanのためにクライアントがのelasticsearchノードにするか、ちょうどエラーをスローするようにタイムアウトするべきか
http_auth	なhttpは`username:password`でここにでき`username:password`

Examples

ドキュメントのけつまり、サンプルの

なPythonライブラリをインストールする

```
$ pip install elasticsearch
```

Elasticsearchにし、をしデータ、Elasticsearchをしてを「け」します。

```
from datetime import datetime
from elasticsearch import Elasticsearch

# Connect to Elasticsearch using default options (localhost:9200)
es = Elasticsearch()

# Define a simple Dictionary object that we'll index to make a document in ES
doc = {
    'author': 'kimchy',
```

```
    'text': 'Elasticsearch: cool. bonsai cool.',
    'timestamp': datetime.now(),
}

# Write a document
res = es.index(index="test-index", doc_type='tweet', id=1, body=doc)
print(res['created'])

# Fetch the document
res = es.get(index="test-index", doc_type='tweet', id=1)
print(res['_source'])

# Refresh the specified index (or indices) to guarantee that the document
#  is searchable (avoid race conditions with near realtime search)
es.indices.refresh(index="test-index")

# Search for the document
res = es.search(index="test-index", body={"query": {"match_all": {}}})
print("Got %d Hits:" % res['hits']['total'])

# Show each "hit" or search response (max of 10 by default)
for hit in res['hits']['hits']:
    print("%(timestamp)s %(author)s: %(text)s" % hit["_source"])
```

クラスタへの

```
es = Elasticsearch(hosts=hosts, sniff_on_start=True, sniff_on_connection_fail=True,
sniffer_timeout=60, sniff_timeout=10, retry_on_timeout=True)
```

のインデックスのとマッピングの

このでは、マッピングをすることによってのインデックスをしますインデックスなしのドキュメント。

まず、 `ElasticSearch`インスタンスをし、したマッピングをします。に、インデックスがするかどうかをし、しないは、それぞれインデックスとマッピングのをむ`index`と`body`パラメータをして`index`をします。

```
from elasticsearch import Elasticsearch

# create an ElasticSearch instance
es = Elasticsearch()
# name the index
index_name = "my_index"
# define the mapping
mapping = {
    "mappings": {
        "my_type": {
                "properties": {
                    "foo": {'type': 'text'},
                    "bar": {'type': 'keyword'}
                }
            }
        }
    }
```

```
# create an empty index with the defined mapping - no documents added
if not es.indices.exists(index_name):
    res = es.indices.create(
        index=index_name,
        body=mapping
    )
    # check the response of the request
    print(res)
    # check the result of the mapping on the index
    print(es.indices.get_mapping(index_name))
```

とによる

ななのがなにされます。つまり、のでは、 `doc_id`ののフィールド`name`は 'John'にされます。フィールドがないは、フィールドにされます。

```
doc = {
    "doc": {
        "name": "John"
    }
}
es.update(index='index_name',
          doc_type='doc_name',
          id='doc_id',
          body=doc)
```

によるをたすをするがあるにします。つまり、のでは、 `name`フィールドが 'John'とするのをします。

```
q = {
  "script": {
    "inline": "ctx._source.age=23",
    "lang": "painless"
  },
  "query": {
    "match": {
        "name": "John"
    }
  }
}

es.update_by_query(body=q,
                   doc_type='doc_name',
                   index='index_name')
```

オンラインでPythonインタフェースをむ https://riptutorial.com/ja/elasticsearch/topic/2068/pythonインタフェース

3: アナライザ

アナライザは、フィールドからテキストをりし、クエリにされるトークンをします。

アナライザはのでします。

- `CharFilters` ゼロ
- `Tokenizer` 1
- `TokenFilters` ゼロ

アナライザは、フィールドにインデックスがけられたときに、ではなくトークンでわれるように、マッピングにできます。するとき、もアナライザーをしてされます。したがって、アナライザでテキストをすると、クエリにされていないがまれていてもにします。

Examples

マッピング

アナライザは「アナライザ」をしてマッピングにできます。デフォルトでは、「」アナライザがされています。あるいは、トークンやがにたないためにアナライザをしたくないは、 "index"をすることができます "not_analyzed"

```
PUT my_index
{
  "mappings": {
    "user": {
      "properties": {
        "name": {
          "type": "string"
          "analyzer":  "my_user_name_analyzer"
        },
        "id": {
          "type": "string",
          "index": "not_analyzed"
        }
      }
    }
  }
}
```

マルチフィールド

によっては、なるアナライザでフィールドののなるインデックスをつことがなもあります。マルチフィールドをすると、そうすることができます。

```
PUT my_index
{
  "mappings": {
```

```
    "user": {
      "properties": {
        "name": {
          "type": "string"
          "analyzer": "standard",
          "fields": {
            "special": {
               "type": "string",
               "analyzer": "my_user_name_analyzer"
            },
            "unanalyzed": {
              "type": "string",
              "index": "not_analyzed"
            }
          }
        }
      }
    }
  }
}
```

クエリをするには、に "user.name"このはスタンダードアナライザをしますをにするのではなく、 "user.name.special"または "user.name.unanalyzed"をできます。ドキュメントはされずにることにしてください。これはけにのみします。

アナライザ

elasticsearchのは、のデータをしたいときにコンテキストになります。

アナライザをすると、のをできます。

-
- ステミング
- ミス

たちは、それぞれをていきます。

1. をして、elasticsearchにたちのデータの、すなわちdr => Doctorをどのようにうかをえてもらえます。たちのでdoctorキーワードをすると、elastsearchはdrでされたもします。

2. ステミング

   アナライザーでステミングをすると、されたのようなベース・ワードをすることができます。

ワード	
する	、

3. タイプミス

アナライザはまた、「Resurrection」というのをしているかどうかをべているにタイプミングをいます。それで、elasticsearchはタイプミスのをします。これはresurection、ressurectionをじようにし、をします。

ワード	
	、

のアナライザ

1.
2. シンプル
3.
4. やめる
5. キーワード
6. パターン
7.
8. スノーボール

ケースアナライザをする

によっては、ドキュメントのにして、クエリのケースをするがあるがあります。この、にをするためにをすることができる。フィールドは、このアナライザーをプロパティにめて、させるがあります

```
"settings": {
        "analysis": {
            "analyzer": {
                "case_insensitive": {
               "tokenizer": "keyword",
               "filter": ["lowercase"]
            }
            }
        }
    }
```

オンラインでアナライザをむ https://riptutorial.com/ja/elasticsearch/topic/6232/アナライザ

4: カールコマンド

- curl -X <VERB> '<PROTOCOL>// <HOST><PORT> / <PATH><QUERY_STRING>' -d '<BODY>'
-
- VERBなHTTPメソッドまたはGET、POST、PUT、HEAD、またはDELETE
- PROTOCOLhttpまたはhttpsのいずれかElasticsearchのにhttpsプロキシがある
- HOSTElasticsearchクラスタののノードのホスト、またはローカルマシンのノードのローカルホスト。
- PORTElasticsearch HTTPサービスをするポート。は9200です。
- PATHAPIエンドポイントたとえば、_countはクラスタのドキュメントをします。パスに_cluster / statsや_nodes / stats / jvmなどののコンポーネントがまれているがあります
- QUERY_STRINGオプションのクエリパラメータprettyはみやすくするためにJSONレスポンスをきれいにします
- BODYJSONでエンコードされたリクエストリクエストにな
- リファレンス Elasticsearchとのしい Elasticsearch Docs

Examples

Curl クラスタのをカウントするコマンド

```
curl -XGET 'http://www.example.com:9200/myIndexName/_count?pretty'
```

```
{
  "count" : 90,
  "_shards" : {
    "total" : 6,
    "successful" : 6,
    "failed" : 0
  }
}
```

には90のがあります。

リンク ここに

IDでドキュメントをする

```
curl -XGET 'http://www.example.com:9200/myIndexName/myTypeName/1'
```

```
{
    "_index" : "myIndexName",
    "_type" : "myTypeName",
    "_id" : "1",
    "_version" : 1,
    "found": true,
    "_source" : {
        "user" : "mrunal",
        "postDate" : "2016-07-25T15:48:12",
        "message" : "This is test document!"
    }
}
```

リンク ここに

インデックスをする

```
curl -XPUT 'www.example.com:9200/myIndexName?pretty'
```

```
{
  "acknowledged" : true
}
```

リンク ここに

すべてのインデックスをする

```
curl 'www.example.com:9200/_cat/indices?v'
```

```
health status index               pri rep docs.count docs.deleted store.size pri.store.size
green  open   logstash-2016.07.21   5   1       4760            0      4.8mb          2.4mb
green  open   logstash-2016.07.20   5   1       7232            0      7.5mb          3.7mb
green  open   logstash-2016.07.22   5   1      93528            0    103.6mb           52mb
green  open   logstash-2016.07.25   5   1      20683            0     41.5mb         21.1mb
```

リンク ここに

インデックスをする

```
curl -XDELETE 'http://www.example.com:9200/myIndexName?pretty'
```

```
{
  "acknowledged" : true
}
```

リンク ここに

インデックスのすべてのをする

```
curl -XGET http://www.example.com:9200/myIndexName/_search?pretty=true&q=*:*
```

これは`Search` APIをし、インデックス`myIndexName`のすべてのエントリをします。

リンク ここに

オンラインでカールコマンドをむ https://riptutorial.com/ja/elasticsearch/topic/3703/カールコマンド

5: キバナをった

き

Kibanaはelasticsearchのフロントエンドデーータツーールです。キバナをインストーールするには、キバナのドキュメントをしてください。 localhostでkibanaをするには、 https// localhost5601にき、kibana consoleにします。

Examples

をってあなたのクラスターをする

コマンドのはのようになります。

```
<REST Verb> /<Index>/<Type>/<ID>
```

のコマンドをし、Kibana Consoleをしてelasticsearchクラスタをします。

- クラスタのをするには

```
GET /_cat/health?v
```

- すべてのインデックスをする

```
GET /_cat/indices?v
```

- でをする

```
 PUT /car?pretty
```

- id 1をして、のcarでをけする

```
PUT /car/external/1?pretty
{
  "name": "Tata Nexon"
}
```

のクエリのはのようになります。

```
{
  "_index": "car",
  "_type": "external",
  "_id": "1",
  "_version": 1,
  "result": "created",
```

```
  "_shards": {
    "total": 2,
    "successful": 1,
    "failed": 0
  },
  "created": true
}
```

- のドキュメントをするには、のコマンドをします。

```
GET /car/external/1?pretty
```

- インデックスをする

```
DELETE /car?pretty
```

elasticsearchデ―タをする

Elasticsearchは、ほぼリアルタイムでデ―タとデ―タをします。このでは、、、およびバッチがあります。

- じをしています。すでに/ car / external / 1にをけしているとします。その、デ―タをけするコマンドをすると、のがきえられます。

```
PUT /car/external/1?pretty
{
  "name": "Tata Nexa"
}
```

"Tata Nexon"というのid 1ののののドキュメントはしい "Tata Nexa"でされます

- なIDでデ―タをけする

```
POST /car/external?pretty
{
  "name": "Jane Doe"
}
```

  IDなしでをけするために、 **PUT**のわりに**POST**をします。 IDをしないと、elasticsearchはランダムなIDをし、それをってドキュメントのインデックスをします。

- Idでにのをする。

```
POST /car/external/1/_update?pretty
{
  "doc": { "name": "Tata Nex" }
}
```

-

でドキュメントをする

```
POST /car/external/1/_update?pretty
{
  "doc": { "name": "Tata Nexon", "price": 1000000 }
}
```

- なスクリプトをしてドキュメントをします。

```
POST /car/external/1/_update?pretty
{
  "script" : "ctx._source.price += 50000"
}
```

ctx._sourceは、しようとしているのソースをします。のスクリプトは、にされるスクリプトを1つだけします。

- ドキュメントの

```
DELETE /car/external/1?pretty
```

クエリAPIによるをしてすべてのドキュメントをするよりも、インデックスをするほうがです

バッチ

elasticsearchは、ドキュメントのとのインデックスとはに、 **_bulk** APIをしてのをバッチでするもします。

- **_bulk** APIをしてのドキュメントをする

```
POST /car/external/_bulk?pretty
{"index":{"_id":"1"}}
{"name": "Tata Nexon" }
{"index":{"_id":"2"}}
{"name": "Tata Nano" }
```

- **_bulk** APIをしてドキュメントをおよびする

```
POST /car/external/_bulk?pretty
{"update":{"_id":"1"}}
{"doc": { "name": "Tata Nano" } }
{"delete":{"_id":"2"}}
```

がすると、バルクAPIはしません。すべてのをし、にすべてののレポートをします。

オンラインでキバナをったをむ https://riptutorial.com/ja/elasticsearch/topic/10058/キバナをった

6: クラスタ

クラスタヘルスは、りてられたシャードの「アクティブ」、りてられていないされているシャードなど、クラスタにするくのをします。さらに、クラスタののノードとデータノードをします。これにより、しているノードをポーリングすることができますたとえば、 `15`になるとされますが、 `14` 、 。

Elasticsearchについてっているには、「りてみ」と「りて」のがのにつことがあります。

クラスタヘルスからもよくされるフィールドは、の3つののいずれかになることができる`status`です。

-
-
-

はそれぞれ1つだけです - になもの

1. は、 なくとも 1つのプライマリーシャードがないことをします。
   - 1シャードがつからないということは、ほとんどの、をしてしいデータをきむけすることができないことをします。
     - には、そのでなプライマリーにはまだけすることができますが、には、えられたをけるシャードをにしないためできません。
     - いクラスタにしてもはですが、するインデックスにシャードがないにながられることをします。
   - は、プライマリシャードがりてられていることをし`initializing_shards` `initializing_shards` 。
   - ノードをクラスタからったばかりのたとえば、しているマシンがをったなど、シャードがにしていることがにかなっています。
     - このプライマリシャードのレプリカシャードは、このシナリオではプライマリシャードにされます。
2. はすべてのプライマリーがアクティブであることをしますが、なくとも 1つのレプリカはありません。
   - しているレプリカは、 でインデックスにをえるがあるにのみ、インデックスにします。
     - デフォルトでは、のプライマリにして1つのレプリカしかせず、1つのしたレプリカでインデックスけがわれるがあります。
   - は、シャードがりてられていることをし`initializing_shards` `initializing_shards` 。
   - レプリカをにした1つのノードクラスタはに でです。シャードがまだりてられていないは、になることがあります。
     - のノードしかっていないは、レプリカをしないようにするため、レプリカをにすることはにかなっています。それはにすることができます。
3. は、すべてのシャードがアクティブであることをします。
   - のクラスタでされるのシャードは、 `relocating_shards`です。
   -

しいインデックス、したがってしいシャードは、シャードがりてられたときにクラスター―をから、にえますはにに、であればをにします。

- Elasticsearch 5.xでは、りてにがかかりすぎるまで、しいインデックスはクラスタをくしません 。

Examples

がめるのクラスタヘルス

では、なHTTPをします。このの<#>は、コピ―するときにするがあります。

_cat APIをすると、さまざまなでがめるのをることができます。

```
GET /_cat/health?v <1>
```

1. ?vはオプションですが、 ""ながなことをします。

_cat/healthはElasticsearch 1.xしていましたが、ここではElasticsearch 5.xのです

の

```
epoch      timestamp cluster       status node.total node.data shards pri relo init unassign
pending_tasks max_task_wait_time active_shards_percent
1469302011 15:26:51  elasticsearch yellow          1         1     45  45    0    0       44
0                   -                  50.6%
```

ヘッダ―のない、がめる、のクラスター―ヘルス

では、なHTTPをします。このの<#>は、コピ―するときにするがあります。

_cat APIをすると、さまざまなでがめるのをることができます。

```
GET /_cat/health <1>
```

_cat/healthはElasticsearch 1.xしていましたが、ここではElasticsearch 5.xのです

なし

```
1469302245 15:30:45 elasticsearch yellow 1 1 45 45 0 0 44 0 - 50.6%
```

したヘッダをつ、がめる、のクラスタヘルス

では、なHTTPをします。このの<#>は、コピ―するときにするがあります。

Elasticsearchのほとんどの_cat APIとに、APIはデフォルトのフィ―ルドセットでにします。ただし、にじて、APIののフィ―ルドがします。

```
GET /_cat/health?help <1>
```

1. `?help`と、APIはなとにフィーールドおよびいをします。

`_cat/health`はElasticsearch 1.xしていましたが、ここではElasticsearch 5.xのです

こののになフィーールド

```
epoch                | t,time                                  | seconds since 1970-01-01
00:00:00
timestamp            | ts,hms,hhmmss                           | time in HH:MM:SS

cluster              | cl                                      | cluster name

status               | st                                      | health status

node.total           | nt,nodeTotal                            | total number of nodes

node.data            | nd,nodeData                             | number of nodes that can
store data
shards               | t,sh,shards.total,shardsTotal           | total number of shards

pri                  | p,shards.primary,shardsPrimary          | number of primary shards

relo                 | r,shards.relocating,shardsRelocating    | number of relocating nodes

init                 | i,shards.initializing,shardsInitializing | number of initializing
nodes
unassign             | u,shards.unassigned,shardsUnassigned    | number of unassigned shards

pending_tasks        | pt,pendingTasks                         | number of pending tasks

max_task_wait_time   | mtwt,maxTaskWaitTime                    | wait time of longest task
pending
active_shards_percent | asp,activeShardsPercent                | active number of shards in
percent
```

これをって、これらのフィーールドだけをすることができます

```
GET /_cat/health?h=timestamp,cl,status&v <1>
```

1. `h=...`は、すフィーールドのリストをします。
2. `v` は、ヘッダーをすることをします。

Elasticsearch 5.xのインスタンスからの

```
timestamp cl            status
15:38:00  elasticsearch yellow
```

JSONベースのクラスタヘルス

では、なHTTPをします。こののは、コピーするときにするがあります。

_cat APIは、くの、がクラスタにするをで_catするのにです。しかし、しばしば、ソフトウェアでするなをしてしがる。に、JSON APIはこののためにされています。

```
GET /_cluster/health
```

_cluster/healthはElasticsearch 1.xしていましたが、ここではElasticsearch 5.xのです

```
{
  "cluster_name": "elasticsearch",
  "status": "yellow",
  "timed_out": false,
  "number_of_nodes": 1,
  "number_of_data_nodes": 1,
  "active_primary_shards": 45,
  "active_shards": 45,
  "relocating_shards": 0,
  "initializing_shards": 0,
  "unassigned_shards": 44,
  "delayed_unassigned_shards": 0,
  "number_of_pending_tasks": 0,
  "number_of_in_flight_fetch": 0,
  "task_max_waiting_in_queue_millis": 0,
  "active_shards_percent_as_number": 50.56179775280899
}
```

オンラインでクラスタをむ https://riptutorial.com/ja/elasticsearch/topic/2069/クラスタ

7: リレーショナルデータベースとのいき

これは、からて、をびたいのためのものです。このトピックでは、リレーショナルデータベースがなオプションではないユースケースをします。

Examples

の

リレーショナルデータベース	
データベース	インデックス
	タイプ
/レコード	
	フィールド

のは、リレーショナルデータベースとelasticsearchのなをにしています。

セットアップ

リレーショナルデータベースのにうことをする

```
create databse test;

use test;

create table product;

create table product (name varchar, id int PRIMARY KEY);

insert into product  (id,name) VALUES (1,'Shirt');

insert into product  (id,name) VALUES (2,'Red Shirt');

select * from product;

name       | id
-----------+----
Shirt      |  1
Red Shirt |  2
```

エラスティックサーチ

```
POST test/product
{
    "id" : 1,
    "name" : "Shirt"
}

POST test/product
{
    "id" : 2,
    "name" : "Red Shirt"
}

GET test/product/_search


"hits": [
     {                                          ==============
        "_index": "test",                       ===> index    |
        "_type": "product",                     ===>type      |
        "_id": "AVzglFomaus3G2tXc6sB",                        |
        "_score": 1,                                          |
        "_source": {                                          |===> document
           "id": 2,                           ===>field      |
           "name": "Red Shirt"                ===>field      |
        }                                                     |
     },                                         ==============
     {
        "_index": "test",
        "_type": "product",
        "_id": "AVzglD12aus3G2tXc6sA",
        "_score": 1,
        "_source": {
           "id": 1,
           "name": "Shirt"
        }
     }
  ]
```

リレーショナルデータベースがでない

- のはそのにある。もが、ながにされるようにをすることをんでいます。リレーショナルデータベースにはこのようなはありません。、Elasticsearchは、デフォルトでについてをします。

  セットアップ

  のでしたのとじです。

  ユーザーが*shirts*をしたいとっていて、 *red*のシャツにがあるとします。その、 *red*と*shirts*キーワードをむがにされます。その、のシャツのがされます。

  リレーショナルデータベースクエリをしたソリューション

  `select * from product where name like '%Red%' or name like '%Shirt%'`。

```
name       | id
-----------+----
Shirt      |  1
Red Shirt  |  2
```

ソリュ―ション

```
POST test/product/_search
{
     "query": {
          "match": {
            "name": "Red Shirt"
          }
     }
}
```

```
"hits": [
  {
     "_index": "test",
     "_type": "product",
     "_id": "AVzglFomaus3G2tXc6sB",
     "_score": 1.2422675,              ===> Notice this
     "_source": {
        "id": 2,
        "name": "Red Shirt"
     }
  },
  {
     "_index": "test",
     "_type": "product",
     "_id": "AVzglD12aus3G2tXc6sA",
     "_score": 0.25427115,             ===> Notice this
     "_source": {
        "id": 1,
        "name": "Shirt"
     }
  }
 ]
```

でてきたように、Relational Databaseはランダムなでをしましたが、Elasticsearchはについてされた`_score`でをします。

- をしている、ってしまうがあります。ユ―ザ―がったパラメ―タをするがあります。リレ―ショナルデ―タベ―スはこのようなケ―スをしません。のためのの。

  セットアップ

  のでしたのとじです。

  ユ―ザ―がをしたい*`shirts`*が、はったにり*`shrt`*ってを。ユ―ザ―はまだシャツのをることをしています 。

  リレ―ショナルデ―タベ―スクエリをしたソリュ―ション

```
select * from product where name like '%shrt%'
```

```
No results found
```

ソリューション

```
POST /test/product/_search

 {
    "query": {
      "match": {
        "name": {
          "query": "shrt",
          "fuzziness": 2,
          "prefix_length": 0
         }
      }
    }
 }
```

```
 "hits": [
  {
     "_index": "test",
     "_type": "product",
     "_id": "AVzglD12aus3G2tXc6sA",
     "_score": 1,
     "_source": {
        "id": 1,
        "name": "Shirt"
     }
  },
  {
     "_index": "test",
     "_type": "product",
     "_id": "AVzglFomaus3G2tXc6sB",
     "_score": 0.8784157,
     "_source": {
        "id": 2,
        "name": "Red Shirt"
     }
  }
]
```

のように、リレーショナルデータベースはったのをしませんでしたが、Elasticsearchはな`fuzzy`クエリをしてをします。

オンラインでリレーショナルデータベースとのいをむ
https://riptutorial.com/ja/elasticsearch/topic/10632/リレーショナルデータベースとのい

8:

Elasticsearchには、のためのれたエクスペリエンスをするデフォルトのセットがしています。のステートメントは、それがずしものニーズにわせてられなければならず、したがってすることができない、にとってずしもらしいものではないということです。

のでは 、をすることなくじマシンでのノードをにダウンロードしてできます。

はどこですか

Elasticsearchのインストールのには`config/elasticsearch.yml`ます。そこにはのがありま す

- `cluster.name`
  - ノードがしているクラスターの。じクラスタのすべてのノードは、じをするがあります。
  - 、デフォルトは`elasticsearch`です。
- `node.*`
  - `node.name`
    - しないは、ノードがするたびにランダムながされます 。これはしいかもしれませんが、にはしていません。
    - はであるはありませんが、であるがあります。
  - `node.master`
    - ブールの。 `true`、ノードはなマスターノードであり、されたマスターノードであるがあります。
    - デフォルトは`true`です。これは、すべてのノードがなマスターノードであることをします。
  - `node.data`
    - ブールの。 `true`、ノードがデータをし、アクティビティをすることをします。
    - デフォルトは`true`です。
- `path.*`
  - `path.data`
    - ノードのファイルがきまれる。 すべてのノードはこのディレクトリをしてメタデータをしますが、データノードはこれをしてドキュメントを/けします。
    - デフォルトは`./data`です。
      - つまり、Elasticsearchディレクトリのに`config`するピアディレクトリとして`data`がされます。
  - `path.logs`
    - ログファイルがきまれる。
    - デフォルトは`./logs`です。
- `network.*`
  - `network.host`
    - デフォルトは`_local_`にされてい`_local_` 、に`localhost`です。
      - つまり、デフォルトでは、のマシンのからノードをすることはできません 。

    - `network.bind_host`
        - にです。これはElasticsearchにソケットをバインドするのマシンのアドレスをします。
            - このリストは、マシンののマシンたとえば、クラスタののノ―ドがこのノ―ドとできるようにします。
        - デフォルトは`network.host`です。
    - `network.publish_host`
        - このノ―ドとのなをのノ―ドにするためにされるのホスト。
            - `network.bind_host`にをする、これはノ―ドにされる*1*つのホストでなければなりません。
        - デフォルトはnetwork.host`です。
- `discovery.zen.*`
    - `discovery.zen.minimum_master_nodes`
        - マスター―のをします。これは 、のをしてするがあります。 `(M / 2) + 1`ここで、`M`はなマスター―ノ―ドのです `node.master: true`をするノ―ド`node.master: true`またはに`node.master: true`。
        - デフォルトは`1`で、のノ―ドクラスタにしてのみです。
    - `discovery.zen.ping.unicast.hosts`
        - このノ―ドをクラスタのりのにするためのメカニズム。
        - これにより、ノ―ドがクラスタのりのをつけることができるようになマスター―ノ―ドがリストされます。
        - ここでするがあるは、のノ―ドの`network.publish_host`です。
        - デフォルトは`localhost`になります。つまり、ロ―カルマシンでクラスタをすることのみをべます。

どのようながありますか

Elasticsearchは3つのなるタイプのをします

- クラスタの
    - これらは、すべてのノ―ドやすべてのインデックスなど、クラスタのすべてにされるです。
- ノ―ド
    - これらは、のノ―ドだけにされるです。
- インデックスの
    - これらは、インデックスだけにされるです。

にじて、のようにできます。

- ににされました
- インデックスのクロ―ズ/オ―プンにされました
    - インデックスレベルのによっては、インデックスをじてオ―プンするはありませんが、インデックスをにマ―ジしてするがあります。
        -

のレベルは、このタイプののです。それはにすることができますが、しいセグメントのみがをします。したがって、インデックスがされない、インデックスをにセグメントをしないり、をすることはありません。

- ノードのにされた
- クラスタのにされた
- されていない

でができるかできないかについては、にElasticsearchのバージョンのドキュメントをチェックしてください。

どのようにをできますか

にはいくつかのがありますが、そのうちのいくつかはされていません。

- コマンドライン

Elasticsearch 1.xおよび2.xでは、ほとんどのをJavaシステムプロパティとしてプレフィックスとして`es.`

```
$ bin/elasticsearch -Des.cluster.name=my_cluster -Des.node.name=`hostname`
```

Elasticsearch 5.xでは、 `-Des.`のわりに`-E`カスタムをするわりに、Java System Propertiesをしないようにされています`-Des.`

```
$ bin/elasticsearch -Ecluster.name=my_cluster -Enode.name=`hostname`
```

Puppet、Chef、またはAnabilitiesなどのツールをしてクラスタをおよびする、このをするこのアプローチはです。しかし、でうとにうまくしません。

- YAMLの
  - にす
-
  - にす

がされるは、もなです

1. な
2. な
3. コマンドラインの
4. YAML

が2、いずれかのレベルで1されている、レベルがになります。

Examples

な

ElasticsearchはYAMLYet Another Markup Languageファイルをします。このファイルは、デフォルトのElasticsearchディレクトリ RPMとDEBインストールではこのがされますにあります。

`config/elasticsearch.yml`でをうことができます

```
# Change the cluster name. All nodes in the same cluster must use the same name!
cluster.name: my_cluster_name

# Set the node's name using the hostname, which is an environment variable!
# This is a convenient way to uniquely set it per machine without having to make
#  a unique configuration file per node.
node.name: ${HOSTNAME}

# ALL nodes should set this setting, regardless of node type
path.data: /path/to/store/data

# This is a both a master and data node (defaults)
node.master: true
node.data: true

# This tells Elasticsearch to bind all sockets to only be available
#  at localhost (default)
network.host: _local_
```

なクラスタ

クラスター—がすでにしたににをするがあり、ににできるは、 `_cluster/settings` APIをして`_cluster/settings`できます。

なは、な2つのタイプのクラスター—のの1つです。なは 、なクラスタのもします。

すべてのをにできるわけではありません。たとえば、クラスター—のをにすることはできません。ほとんどのノ—ドレベルは、にタ—ゲティングすることはできないため、にすることもできません。

これは、インデックスレベルのにするAPIではありません 。はインデックスでまるがあるため、インデックスレベルのであることがわかり`index.`。 が`indices.`の`indices.`らはすべてのインデックスにされるため、クラスタのです 。

```
POST /_cluster/settings
{
  "persistent": {
    "cluster.routing.allocation.enable": "none"
  }
}
```

Elasticsearch 1.xおよび2.xでは、なをできません。

なことに、これはElasticsearch 5.xでされました。これで、 `null`することでをできるようになりました。

```
POST /_cluster/settings
{
  "persistent": {
    "cluster.routing.allocation.enable": null
  }
}
```

のは、デフォルト、またはのいレベルでされたコマンドラインなどにります。

なクラスタ

クラスター—がすでにしたににをするがあり、ににできるは、 `_cluster/settings` APIをして `_cluster/settings`できます。

なは、な2つのタイプのクラスター—のの1つです。なは、なクラスタではしません 。

すべてのをにできるわけではありません。たとえば、クラスター—のをにすることはできません。ほとんどのノ—ドレベルは、にタ—ゲティングすることはできないため、にすることもできません。

これは、インデックスレベルのにするAPIではありません 。はインデックスでまるがあるため、インデックスレベルのであることがわかり`index.`。 が`indices.`の`indices.`らはすべてのインデックスにされるため、クラスタのです 。

```
POST /_cluster/settings
{
  "transient": {
    "cluster.routing.allocation.enable": "none"
  }
}
```

Elasticsearch 1.xおよび2.xでは、クラスタをにせずになをすることはできません。

なことに、これはElasticsearch 5.xでされました。これで、nullにすることでをできるようになりました。

```
POST /_cluster/settings
{
  "transient": {
    "cluster.routing.allocation.enable": null
  }
}
```

をすると、デフォルト、またはよりいでされたたとえば、 `persistent`にります。

インデックスの

インデックスは、のインデックスにされるです。このようなは`index.`まり`index.`。 こののは`number_of_shards`と`number_of_replicas`であり、 `index.number_of_shards`と`index.number_of_replicas`で

もします。

がすように、インデックスレベルのはのインデックスにされます。のは、インデックスのプライマリのをする`index.number_of_shards`など、にできないため、にするがあります。

```
PUT /my_index
{
  "settings": {
    "index.number_of_shards": 1,
    "index.number_of_replicas": 1
  }
}
```

よりなで、キープレフィックスをそれぞれみわせることができます.

```
PUT /my_index
{
  "settings": {
    "index": {
      "number_of_shards": 1,
      "number_of_replicas": 1
    }
  }
}
```

のでは、されたでインデックスがされます。 index `_settings`エンドポイントをして、インデックスごとのをにすることができます。えば、ここでは、にslowlogを のレベルのために

```
PUT /my_index/_settings
{
  "index": {
    "indexing.slowlog.threshold.index.warn": "1s",
    "search.slowlog.threshold": {
      "fetch.warn": "500ms",
      "query.warn": "2s"
    }
  }
}
```

Elasticsearch 1.xおよび2.xでは、インデックスレベルのをにしていませんでした。タイプミスがあったや、にをったは、それをにけれるでしょうが、そうでなければします。 Elasticsearch 5.xはにをし、なタイプミスまたはプラグインがないためのインデックスをしようとするみをします。のステートメントは、にされるインデックスおよびにされます。

のインデックスのインデックスをに

「`Index Settings`にされているじを、 すべてののインデックスに1のリクエストですることも、それらのサブセットをすることもできます。

```
PUT /*/_settings
{
  "index": {
```

```
    "indexing.slowlog.threshold.index.warn": "1s",
    "search.slowlog.threshold": {
      "fetch.warn": "500ms",
      "query.warn": "2s"
    }
  }
}
```

または

```
PUT /_all/_settings
{
  "index": {
    "indexing.slowlog.threshold.index.warn": "1s",
    "search.slowlog.threshold": {
      "fetch.warn": "500ms",
      "query.warn": "2s"
    }
  }
}
```

または

```
PUT /_settings
{
  "index": {
    "indexing.slowlog.threshold.index.warn": "1s",
    "search.slowlog.threshold": {
      "fetch.warn": "500ms",
      "query.warn": "2s"
    }
  }
}
```

よりにしたいは、すべてをせずにをすることができます。

```
PUT /logstash-*,my_other_index,some-other-*/_settings
{
  "index": {
    "indexing.slowlog.threshold.index.warn": "1s",
    "search.slowlog.threshold": {
      "fetch.warn": "500ms",
      "query.warn": "2s"
    }
  }
}
```

オンラインでをむ https://riptutorial.com/ja/elasticsearch/topic/3411/

9: とのい

`index`がSQLデータベースであるSQLデータベースのテーブルのような`type` sをるのはです。しかし、これは`type` sにづくいではありません。

すべてのタイプについて

、はり、Elasticsearch `_type`によってドキュメントにされたなるメタデータフィールド`_type`。のでは、`my_type`と`my_other_type` 2がされています。つまり、にけられたドキュメントには、`"_type": "my_type"`ようににされたなフィールドがあります`"_type": "my_type"`;これはドキュメントでけされているため、またはフィルタなフィールドになりますが、のドキュメントにはしません。したがって、アプリケーションでにするはありません。

すべてのタイプはじインデックスにするため、じインデックスにされます。ディスクレベルであっても、じファイルにします。2のタイプをするのはなものである。であるかどうかにかかわらず、すべてのタイプがマッピングにするがあり、それらのマッピングはすべてクラスタでするがあります。これはメモリをし、タイプがにされると、マッピングがするとパフォーマンスがします。

したがって、にのタイプをとしないり、のタイプのみをすることがベストプラクティスです。のタイプがましいシナリオをるのがです。たとえば、のインデックスがあるとします。のタイプでするとです。

- BMW
- シエビー
- ホンダ
- マツダ
- メルセデス
-
- レンジローバー
- トヨタ
- ...

こので、すべてのをするか、オンデマンドでメーカーがすることができます。これら2つののいは、のようにです。

```
GET /cars/_search
```

そして

```
GET /cars/bmw/_search
```

Elasticsearchのしいユーザーにとってらかでないことは、2のフォームがのフォームのであること

です。それはどおりきされます

```
GET /cars/_search
{
  "query": {
    "bool": {
      "filter": [
        {
          "term" : {
            "_type": "bmw"
          }
        }
      ]
    }
  }
}
```

が`bmw` `_type`フィ―ルドでけされていないすべてのをに`bmw`ます。すべてのドキュメントは`_type`フィ―ルドとしてそのでけされるため、これはかなりなフィルタとしてします。どちらのでものがされていた、フィルタはにじてフルにされます。

そのため、がであれば、のこのでは`manufacturer`などをし、それをにするがはるかにれています。に、で、 `make` というのフィ―ルドまたはのをにし、したいときはいつでもでフィルター―をします。これにより、マッピングのサイズが`1/n`にします。ここで、 `n`はのタイプのです。そうでなければされたマッピングののために、ドキュメントにのフィ―ルドをします。

Elasticsearch 1.xと2.xでは、このようなフィ―ルドはのようにするがあります。

```
PUT /cars
{
  "manufacturer": { <1>
    "properties": {
      "make": { <2>
        "type": "string",
        "index": "not_analyzed"
      }
    }
  }
}
```

1. はです。
2. はで、あなたがむならそれはとするかもしれません。

Elasticsearch 5.xでは、はしますがされません、よりいはをすることです

```
PUT /cars
{
  "manufacturer": { <1>
    "properties": {
      "make": { <2>
        "type": "keyword"
      }
    }
  }
```

```
}
```

1. はです。
2. はで、あなたがむならそれはとするかもしれません。

はインデックスでえめにするがあります。、インデックスのマッピングをらませるためです。あなたはなくとも1つはっていなければなりませんが、のものをたなければならないということはもありません。

よくある

- ほとんどじ2つまたはそれのタイプがあり、タイプごとにのフィールドがいくつかあるはどうなりますか

インデックス·レベルでは、まばらにされているいくつかのフィールドとし、のタイプはフィールドをしたことがないというされていないでスパースフィールドのをするののでされているいずれかのタイプにいはありませんs。

いえれば、になく 、まばらにされるフィールドはインデックスでです。スパースは、それがのタイプでされているというだけでインデックスにをもたらしません。

これらのをみわせて、のフィールドをするだけです。

- 々のはくじでフィールドをするがあるのはなぜですか

フィールドはにLuceneレベルで1しかされていないので、そこにいくつのがあるかにはありません。タイプがくしないというは、Elasticsearchのであり、なにぎない 。

- 々にされたじフィールドをつ々のをできますか

いいえ、ES 2.xでうがつかったは、バグレポートをくがあります 。のでしたように、Luceneはそれらをすべて1つのフィールドとみなしているため、にさせるはありません。

ES 1.xはこれをのとしてしました。これにより、インデックスの1つのシャードのマッピングがじインデックスののシャードとになるをすることができました。これはであり、しないにつながるがあります。

ルールの

- /ドキュメントでは 、じインデックスでのをするがあります。
  - は1つのタイプでらしています。
  - はのタイプでらしていますしかし、はそのとじシャードにんでいます。
- のインデックスをすることがましくなく、スパースフィールドのがよりもましいにニッチな 。
  - たとえば、Elasticsearchプラグインmarvel1.xおよび2.xまたはX-Pack Monitoring5.x +

は、クラスタ、ノード、インデックス、のインデックスインデックスレベルのをElasticsearchでします。さらには。されたマッピングのがにじですが、したインデックスのそれはユニークなマッピングをっているか、それがインデックスノートをすることにより、クラスタのをするために、ベストプラクティスにするがあり、それらのをするために、5+インデックスをすることができます。nから1にする。
    - これはなシナリオですが、タイプでフィールドのをするがあります。

Examples

をつをにする

では、cURLやそののHTTPアプリケーションににできるHTTPをしています。また、 Senseとし、Kibana 5.0のConsoleにがされます。

このでは、にをすためにをしています。それらをコピーするとするがあります

```
PUT /my_index <1>
{
  "mappings": {
    "my_type": { <2>
      "properties": {
        "field1": {
          "type": "long"
        },
        "field2": {
          "type": "integer"
        },
        "object1": {
          "type": "object",
          "properties": {
            "field1" : {
              "type": "float"
            }
          }
        }
      }
    }
  },
  "my_other_type": {
    "properties": {
      "field1": {
        "type": "long" <3>
      },
      "field3": { <4>
        "type": "double"
      }
    }
  }
}
```

1. これは、create indexエンドポイントをしてindexをしています。
2. これはtypeしています。
3. じindexのtype sのフィールドは、じをするがあります。 ES 1.xはにこのをしませんでしたが

、のでした。 ES 2.xではこのがにされます。
4. `type`ののフィールドはです。

インデックスまたはインデックスにはがまれます。タイプはドキュメントをするなメカニズムですが、するタイプのマッピングを/またはにするがあります。インデックスに15のタイプをすると、15ののマッピングがあります。

このコンセプトのとタイプをするかどうかについては、をしてください。

タイプでインデックスをにする

では、cURLやそののHTTPアプリケーションににできるHTTPをしています。また、 Senseとし、Kibana 5.0のConsoleにがされます。

このでは、にをするために`<#>`をしています。それらをコピーするとするがあります

```
DELETE /my_index <1>

PUT /my_index/my_type/abc123 <2>
{
  "field1" : 1234, <3>
  "field2" : 456,
  "object1" : {
    "field1" : 7.8 <4>
  }
}
```

1. にするののため、をします。
2. を`my_index`に、タイプ`my_type` ID `abc123` でもですが、にですにけします。
   - デフォルトでは、のは、をけするだけでになります。これはにはですが、ではずしもいとはりません。
3. このフィールドはであるため、にされるときはマッピングするがあります。 Elasticsearchはにすべてのタイプにしてもいタイプをしているため、 `integer`または`short`ではなく`long`としてマッピングされますとも`1234`と`456`むがあります。
4. このフィールドでもじことがてはまります。あなたがむかもしれないように、 `float`ではなく`double`としてマップされます。

このにされるインデックスとタイプは、のでされたマッピングとほぼします。ただし、 `<3>`および`<4>`がにされたマッピングにどのようにするかをすることはです。

じにさらにのタイプをにすることで、これをできます。

```
PUT /my_index/my_other_type/abc123 <1>
{
  "field1": 91, <2>
  "field3": 4.567
}
```

1. タイプはのとののいです。 IDはじで、ですこれは、とはありません`abc123`じインデックスで

あることをこるということに。
2. field1すでににするため、のでされているフィールドとじタイプのフィールドでなければなりません。かでないを"field1": "this is some text"するとします"field1": "this is some text"または"field1": 123.0。

これは、のマッピングしmy_other_typeじインデックス、my_index。

Elasticsearchにインデックスににさせるのではなく、マッピングをにするがにです。

ののけのは、のとていますが、フィールドのタイプはなるため、わずかにです。

```
GET /my_index/_mappings <1>
{
  "mappings": {
    "my_type": { <2>
      "properties": {
        "field1": {
          "type": "long"
        },
        "field2": {
          "type": "long" <3>
        },
        "object1": {
          "type": "object",
          "properties": {
            "field1" : {
              "type": "double" <4>
            }
          }
        }
      }
    }
  },
  "my_other_type": { <5>
    "properties": {
      "field1": {
        "type": "long"
      },
      "field3": {
        "type": "double"
      }
    }
  }
}
```

1. これは_mappingsエンドポイントをして、したインデックスからマッピングをします。
2. こののステップでは、 my_typeをにしmy_typeた。
3. field2は、もってしていなかったので、 integerではなく、 longとなりました。これは 、ディスクストレージではであることがわかります。
4. object1.field1は3とじでdoubleになり、3とじになりました。
    - には、くの、 longがです。ただし、 doubleはであるためできません。
5. このの2のステップでは、 my_other_typeもにしmy_other_typeた。たちがにlongとdoubleをしていたので、そのマッピングはじになりました。
    - ことをえておいてくださいfield1からとしなければなりません my_type そしてそれがあ

りません。
- `field3`はこのタイプになので、そのようなはありません。

オンラインでとのいをむ https://riptutorial.com/ja/elasticsearch/topic/3412/とのい

10: API

き

APIをすると、クエリをし、クエリにするヒットをすことができます。クエリは、なクエリをパラメータとしてするか、またはリクエストをしてできます。

Examples

ルーティング

をすると、すべてのインデックス/インデックスシャードにブロードキャストされますレプリカのラウンドロビン。どのシャードがされるかは、ルーティングパラメータをすることでできます。たとえば、ツイートのインデックスをする、ルーティングにはユーザーをできます。

```
curl -XPOST 'localhost:9200/twitter/tweet?routing=kimchy&pretty' -d'
{
    "user" : "kimchy",
    "postDate" : "2009-11-15T14:12:12",
    "message" : "trying out Elasticsearch"
}'
```

リクエストボディをしてする

DSLをしてelasticsearchでをうこともできます。のqueryによって、Query DSLをしてクエリをすることができます。

```
GET /my_index/type/_search
{
    "query" : {
        "term" : { "field_to_search" : "search_item" }
    }
}
```

マルチ

multi_searchオプションをうと、にのフィールドでクエリをすることができます。

```
GET /_search
{
  "query": {
    "multi_match" : {
      "query":    "text to search",
      "fields": [ "field_1", "field_2" ]
    }
  }
}
```

ブースト^をしてのフィールドのスコアをげたり、フィールド*にワイルドカードをすることもできます

```
GET /_search
    {
      "query": {
        "multi_match" : {
          "query":    "text to search",
          "fields": [ "field_1^2", "field_2*" ]
        }
      }
    }
```

URI、ハイライト

は、パラメータをすることによってURIをしてにすることができる。このモードをしてをすると、すべてのオプションがされるわけではありませんが、すばやく「カールテスト」にです。

```
GET Index/type/_search?q=field:value
```

されるもう1つのなは、ドキュメントのヒットをすることです。

```
GET /_search
{
    "query" : {
        "match": { "field": "value" }
    },
    "highlight" : {
        "fields" : {
            "content" : {}
        }
    }
}
```

の、ヒットごとにのフィールドがされます

オンラインでAPIをむ https://riptutorial.com/ja/elasticsearch/topic/8625/api

11:

- ""{ - ">"{ - "_タイプ>"{ - <_> - } - [、 "メタ"{[<meta_data_body>]}] - [、 "aggregations" {[<sub_aggregation>] +}] - } - [、 "<aggregation_name_2>"{...}] * - }

Examples

これは、されたからされたのをするメトリックです。

```
POST /index/_search?
{
    "aggs" : {
        "avd_value" : { "avg" : { "field" : "name_of_field" } }
    }
}
```

のでは、すべてののがされます。タイプはavgで、フィールドはがされるのフィールドをします。はをします

```
{
    ...
    "aggregations": {
        "avg_value": {
            "value": 75.0
        }
    }
}
```

アグリゲーションののavg_gradeは、されたレスポンスからアグリゲーションをするためのキーとしてもします。

カーディナリティ

のおおよそのをするメトリック。は、ののフィールドからすることも、スクリプトによってすることもできます。

```
POST /index/_search?size=0
{
    "aggs" : {
        "type_count" : {
            "cardinality" : {
                "field" : "type"
            }
        }
    }
}
```

```
{
    ...
```

```
    "aggregations" : {
        "type_count" : {
            "value" : 3
        }
    }
}
```

されたからされたについてをするメトリック。これらのは、ののフィールドからするか、またはされたスクリプトによってすることができます。

extended_statsは、sum_of_squares、variance、std_deviation、およびstd_deviation_boundsなどののメトリックがされるのバージョンです。

```
{
    "aggs" : {
        "stats_values" : { "extended_stats" : { "field" : "field_name" } }
    }
}
```

サンプル

```
{
    ...

    "aggregations": {
        "stats_values": {
           "count": 9,
           "min": 72,
           "max": 99,
           "avg": 86,
           "sum": 774,
           "sum_of_squares": 67028,
           "variance": 51.55555555555556,
           "std_deviation": 7.180219742846005,
           "std_deviation_bounds": {
            "upper": 100.36043948569201,
            "lower": 71.63956051430799
           }
        }
    }
}
```

オンラインでをむ https://riptutorial.com/ja/elasticsearch/topic/10745/

クレジット

S. No		Contributors
1	Elasticsearchをいめる	Ahsanul Haque, Berto, Community, DJanssens, Dulguun, igo, KartikKannapur, manishrw, mightyteja, noscreenname, Onur, rafa.ferreira, RustyBuckets, sarvajeetsuman, SeinopSys, Shivkumar Mallesappa, Stephan-v, Suhas K, Sumit Kumar, Trilarion
2	Pythonインタフエース	aidan.plenert.macdonald, christinabo, KartikKannapur, pickypg, Sumit Kumar
3	アナライザ	Bhushan Gadekar, Sid1199, Thomas
4	カールコマンド	Fawix, Mrunal Pagnis, Mrunal Pagnis
5	キバナをった	sarvajeetsuman
6	クラスタ	Gerardo Rochín, pickypg
7	リレーショナルデータベースとのい	Richa
8		pickypg
9	とのい	pickypg
10	API	aerokite, Sid1199
11		Sid1199