

NoSQL Datenbanken

Vorlesung - Hochschule Mannheim

Elasticsearch

Inhaltsverzeichnis

- ▶ Einführung
- ▶ Logische/ physikalische Infrastruktur
- ▶ API-Abfragen via REST
- ▶ Percolator
- ▶ Analyzers



elasticsearch.



mozilla



Elasticsearch

- ▶ Suchmaschine auf Basis von Lucene
 - Programmierlibrary zur Volltextsuche
 - Near Realtime (NRT)
- ▶ Entwickelt von Elastic
- ▶ Erscheinungsjahr: 2010
- ▶ Aktuelle Version: 2.0.0 (21.Oktober 2015)
- ▶ Plattformunabhängig
- ▶ Apache-Lizenz

Elasticsearch

- ▶ Schema-frei
 - Suchanfragen sind JSON Dokumente
 - Documentstore
- ▶ Zero Configuration
 - Elasticsearch als Bundle herunterladen und entpacken
 - Kann gestartet werden ohne Konfiguration
- ▶ In Java geschrieben
 - Voraussetzung: JDK 7

Anforderung an Suchmaschinen

- ▶ Strukturierte Suche
 - nach Programmiersprache
 - nach Tags
- ▶ Sortierung
 - Nach Relevanz
 - Nach Datentypen
 - Auf und absteigend
- ▶ Pagination
- ▶ Echtzeit

API

▶ HTTP RESTful API

- Aufruf via HTTP-Verben
- Vorgegebenes URL Schema

▶ Native Java API

```
import static org.elasticsearch.common.xcontent.XContentFactory.*;

XContentBuilder builder = jsonBuilder()
    .startObject()
        .field("user", "kimchy")
        .field("postDate", new Date())
        .field("message", "trying out Elasticsearch")
    .endObject()
```


Logische Infrastruktur



Index

- ▶ Besteht aus mehreren Typen
- ▶ Wie eine Datenbank
- ▶ Beinhaltet alle Daten die durchsucht werden sollen

Typen

- ▶ Wie eine Tabelle in der Datenbank
- ▶ Klassifizierung von Daten
- ▶ Beinhaltet mehrere Dokumente

Dokumente mit Attributen

- ▶ Hier werden die eigentlichen Daten zur Suche gespeichert
- ▶ Hat immer die Attribute **id** und **source**
 - *Id*: eindeutiger Identifier des Dokuments
 - *Source*: Beinhaltet die Originaldaten anhand derer weitere Attribute entstanden sind

Physikalische Infrastruktur

- ▶ Datenspeicherung in einem Cluster
- ▶ Bessere Lastverteilung



Physikalische Infrastruktur

- ▶ Jeder Index wird in 5 Shards unterteilt (Default)
- ▶ Automatisches generieren von Replica



Physikalische Infrastruktur



Abfrage-API

- ▶ Alle Abfragen mit den HTTP-Verben
 - GET (Select)
 - PUT (Update, Insert)
 - DELETE (Remove)

```
curl -X<METHOD> 'url' (-d '<bodyAsJSON>')
```

```
curl -XGET 'http://localhost:9200.'
```


Dokumenten-API

► Url-Schema:

/twitter/**user**/**cookiengineer**

Index

Type

Document

Dokumente abfragen (Get-API)

```
curl -XGET 'http://localhost:9200/twitter/user/kimchy?pretty=true'
```

```
curl -XGET 'http://localhost:9200/twitter/tweet/1?pretty=true'
```

- ▶ pretty=true liefert formatiertes JSON

```
martina@Deathwing:~$ curl -XGET 'http://localhost:9200/twitter/user/kimchy?pretty=true'
{
  "_index" : "twitter",
  "_type" : "user",
  "_id" : "kimchy",
  "_version" : 1,
  "found" : true,
  "_source":{ "name" : "Shay Banon" }
}
martina@Deathwing:~$
```

Dokumente anlegen (Update-API)

```
curl -XPUT 'http://localhost:9200/twitter/user/kimchy'  
-d '{ "name" : "Shay Banon" }'
```

```
curl -XPUT 'http://localhost:9200/twitter/tweet/1' -d  
{  
  "user": "kimchy",  
  "postDate": "2009-11-15T13:12:00",  
  "message": "Trying out Elasticsearch, so far so  
good?"  
}'
```

Dokumente löschen (Delete-API)

```
curl -XDELETE 'http://localhost:9200/twitter/tweet/1'
```

```
{
  "_shards" : {
    "total" : 10,
    "failed" : 0,
    "successful" : 10
  },
  "found" : true,
  "_index" : "twitter",
  "_type" : "tweet",
  "_id" : "1",
  "_version" : 2
}
```

Search-API

- ▶ über die URL
 - Mithilfe eines `_search` Flags

`/twitter/user/_search?`

`q=user:kimchy&pretty=true`

```
martina@Deathwing:~$ curl -XGET 'http://localhost:9200/twitter/tweet/
&pretty=true'
{
  "took" : 15,
  "timed_out" : false,
  "_shards" : {
    "total" : 5,
    "successful" : 5,
    "failed" : 0
  },
  "hits" : {
    "total" : 1,
    "max_score" : 0.30685282,
    "hits" : [ {
      "_index" : "twitter",
      "_type" : "tweet",
      "_id" : "1",
      "_score" : 0.30685282,
      "_source":
    {
      "user": "kimchy",
      "postDate": "2009-11-15T13:12:00",
      "message": "Trying out Elasticsearch, so far so good?"
    }
  } ]
}
```

Search-API

► über den Request-Body

```
curl -XGET 'http://localhost:9200/twitter/tweet/_search?
pretty=true' -d '
{
  "query" : {
    "match" : { "user": "kimchy" }
  }
}'
```

→ Alle Dokumente dessen Attribut User == “kimchy” ist

Search-API

► über den Request-Body

```
curl -XGET 'http://localhost:9200/twitter/tweet/_search?pretty=true' -d '{
  "query" : {
    "matchAll" : {}
  }
}'
```

→ Alle Dokumente

Search-API

```
curl -XGET 'http://localhost:9200/twitter/_search?
pretty=true' -d '
{
  "query" : {
    "range" : {
      "postDate" : { "from" : "2009-11-15T13:00:00",
"to" : "2009-11-15T14:00:00" }
    }
  }
}'
```

→ Alle Dokumente deren Postdate zwischen 13 und 14Uhr lagen

Weitere APIs

▶ Count-API (*_count*)

- Zählt alle Dokumente des Users kimchy im Type Tweet

```
curl -XGET 'http://localhost:9200/twitter/tweet/_count?  
q=user:kimchy'
```

• Validate-API (*__validate*)

- Validiert einen Query ohne ihn wirklich auszuführen

```
curl -XGET 'http://localhost:9200/twitter/_validate/query?  
q=user:foo'
```

→ {"valid":true,"_shards":{"total":1,"successful":1,"failed":0}}

Weitere APIs

- ▶ Search Exists-API (exists?)
 - Überprüft ob es ein gegebenen Query ein Ergebnis zurückgeben kann

```
curl -XGET  
'http://localhost:9200/twitter/tweet/_search/exists?  
q=user:kimchy'
```

→

```
{  
  "exists": true  
}
```

Percolator

- ▶ Datenbankentwicklung:
 - 1) Daten sammeln
 - 2) Daten strukturieren / Schema festlegen
 - 3) In die Datenbank speichern
 - 4) Daten manipulieren / anlegen / löschen

Percolator

- ▶ In Elasticsearch sind Queries auch in JSON notiert
- ▶ Queries können als Document gespeichert werden
- ▶ Indizierung von Queries in */_percolator*
- ▶ Zuerst Queries, dann die Daten
→ reversed Search

Percolator - Use Case

- ▶ Clientseitige Subscription
 - Price Monitoring
 - News alerts
 - Logs monitoring
- ▶ Client drückt mithilfe eines JSON Queries aus, was er abonnieren möchte

Indizierung von Queries

```
-XPUT 'http://localhost:9200/_percolator/twitter/es-tweets  
-d  
{  
  "query": {  
    "match" : { "tweet": "katzen"}  
  }  
}'
```

reservierter Index, Index indem der Query gespeichert wird

Query Identifier, Query

Dokumente Percolaten

```
-XGET 'http://localhost:9200/twitter/tweet/_percolator -d  
{  
  "doc": {  
    "tweet" : "katzen sind die intelligentesten Tiere",  
    "nick" : "cookiengineer",  
  }  
}'
```

API Endpoint, **Index** indem percoliert wird

Query Identifier, **Document**

Dokumente Percolaten

```
-XGET 'http://localhost:9200/twitter/tweet/_percolator -d
```

```
{
```

```
  (...)
```

```
}
```

```
→
```

```
{
```

```
  "ok": true,
```

```
  "matches": ["es-tweets"]
```

```
}
```


Analyzers

- ▶ Zur Analyse von Documents
- ▶ Filterung anhand verschiedener Pattern
- ▶ Zusammensetzung von einem oder mehreren Tokenizer
- ▶ Elasticsearch beinhaltet Prebuilt-Analyzers
- ▶ Eigene *custom Analyzers* können gebaut werden

Analyzers

- ▶ Whitespace Analyzer
 - Splittet Text nach Leerzeichen
- ▶ Stop Analyzer
 - Splittet Text nach *stopwords* (selbst definierte Wörter)
- ▶ Pattern Analyzer
 - Separiert Text nach Regulären Ausdrücken

Beispiel

```
PUT /test
{
  "settings": {
    "analysis": {
      "analyzer": {
        "whitespace": {
          "type": "pattern",
          "pattern": "\\s+"
        }
      }
    }
  }
}

GET /test/_analyze?analyzer=whitespace&text=foo,bar baz

# "foo,bar", "baz"
```

Plugins / weitere Produkte

- ▶ Shield
 - Securityerweiterungen für Elasticsearch
 - Per Default keine Authentifizierung
- ▶ Marcel
 - Clustermonitoringtool
 - Registrieren von Ressourcenprobleme
- ▶ Watcher
 - Notificationserver bei Ausfall oder Problemen
- ▶ Kibana
 - Visuelle Darstellung von analysierten Daten in Dashboards

Getting started

- ▶ Java 7 → `java -version | echo $JAVA_HOME`
- ▶ Download und Entpacken:
`https://www.elastic.co/downloads/elasticsearch`
- Ausführen: (Elasticsearch unter `http://localhost:9200/`)
 - `cd elasticsearch/bin`
 - Linux: `./elasticsearch`
 - Windows: `elasticsearch.bat`

Hands on

- ▶ Download Elasticsearch und starte den Server
- ▶ Lege einen Index ***University*** an
- ▶ Speichere in diesem Index 5 Professoren und 5 Studenten
- ▶ Führe mit *curl* folgende API-Calls aus:
 - Finde alle Professoren
 - Gebe die Anzahl der Studierenden aus
 - Finde einen Studierenden anhand seines Vornamens
 - Suche alle Professoren die älter sind als 40
 - (dazu müssen Sie natürlich vorher die Professoren mit dem Attribut Alter versehen haben)
 - Lösche die Professoren, die jünger sind als 35