Florian Hopf

Elasticsearch

Ein praktischer Einstieg

dpunkt.verlag

Florian Hopf arbeitet als freiberuflicher Softwareentwickler in Karlsruhe. Über diverse Content-Management-Systeme auf der Java-Plattform kam er schon früh mit Suchlösungen auf Basis von Lucene in Kontakt. Er ist verantwortlich für kleine und große Suchlösungen, im Intranet und Internet, für Web-Inhalte und anwendungsspezifische Daten, basierend auf Lucene, Solr und Elasticsearch. Er ist einer der Organisatoren der Java User Group Karlsruhe und des Search Meetup Karlsruhe.

Florian Hopf

Elasticsearch

Ein praktischer Einstieg

dpunkt.verlag

Florian Hopf

Lektorat: René Schönfeld
Copy-Editing: Sandra Gottmann, Münster-Nienberge
Satz: Da-TeX, Leipzig
Herstellung: Frank Heidt
Umschlaggestaltung: Helmut Kraus, www.exclam.de
Druck und Bindung: M.P. Media-Print Informationstechnologie GmbH, 33100 Paderborn

Bibliografische Information der Deutschen Nationalbibliothek
Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über http://dnb.d-nb.de abrufbar.

ISBN:
Buch 978-3-86490-289-5
PDF 978-3-86491-826-1
ePub 978-3-86491-827-8
mobi 978-3-86491-828-5

1. Auflage

Wieblinger Weg 17
69123 Heidelberg

5 4 3 2 1 0

Inhaltsverzeichnis

1 Einführung

1.1 Motivation

In den letzten Jahren ist für viele Anwendungen das Suchfeld immer wichtiger geworden. War die Suche früher noch ein Teilaspekt einer Anwendung, wird heutzutage immer mehr Funktionalität über Suchserver wie Elasticsearch abgewickelt. Viele Benutzer greifen auf Web-Inhalte nur noch über die Suche zu, das Menü als Hilfsmittel für die Navigation verliert an Bedeutung. Dabei soll alles möglichst so gut auffindbar sein, wie man es von Diensten wie Google gewohnt ist.

Gleichzeitig können Suchserver auch zur Lösung anderer Probleme genutzt werden. Sie sind oftmals für lesenden Zugriff optimiert und können skalieren. So bieten sie sich für Aufgaben an, die klassischerweise von einer Datenbank übernommen wurden. Anwendungen verkraften dadurch hohe Zugriffszahlen, Lastspitzen können abgefangen werden.

Durch die Verfügbarkeit von flexiblen Suchservern wie Elasticsearch ergeben sich jedoch auch ganz neue Anwendungsfälle – das zentralisierte Logging, bei dem alle anfallenden Lognachrichten einer Anwendung in einem zentralen Speicher zur Auswertung geschrieben werden, kann Zeit sparen und Entwicklernerven schonen.

Neben den Anwendungs-Logs kann das Ganze dann auch noch auf Business-Daten übertragen werden. Durch die flexiblen Auswertungsmöglichkeiten können Daten analysiert und visualisiert werden. Dabei kann der Suchserver bestehende Data-Warehouse-Lösungen ergänzen oder gar ersetzen.

Bevor wir uns in einem ersten Praxisbeispiel anschauen, wie sich die Nutzung von Elasticsearch anfühlt, werfen wir noch einen Blick auf die Geschichte von Elasticsearch, die stark mit der zugrunde liegenden Bibliothek Apache Lucene zusammenhängt.

1.2 Geschichte von Elasticsearch

Wenn es um die Volltextsuche geht, kam man lange Jahre nicht an der Java-Bibliothek Lucene vorbei, die 1999 von Doug Cutting entworfen und bei SourceForge unter einer Open-Source-Lizenz veröffentlicht wurde. Seit 2001 wird das

Projekt bei der Apache Software Foundation weiterentwickelt. Lucene implementiert einen sogenannten invertierten Index, der die Basis für viele Suchlösungen bildet. Damit können Daten performant und flexibel durchsucht werden. Neben der Implementierung auf der JVM in Java entstanden über die Jahre auch noch unterschiedliche Portierungen in anderen Programmiersprachen, beispielsweise PyLucene für Python oder Lucene.NET für die .NET-Runtime [1].

Einen Wandel in der Nutzung gab es in den Jahren nach 2006, als der hauptsächlich von Yonick Seeley geschriebene Suchserver Solr von CNET als Open-Source-Projekt an die Apache Software Foundation übergeben wurde. Apache Solr baut auf Lucene auf und stellt die zugrunde liegende Funktionalität über eine HTTP-Schnittstelle zur Verfügung. [2] Damit kann auch außerhalb der JVM die Funktionalität von Lucene genutzt werden. Durch weitere Features wie die ursprünglich nur in Solr verfügbare Facettierung und den konfigurativen Ansatz wurde Solr schnell ein großer Erfolg.

Im Jahre 2010 schließlich veröffentlichte Shay Banon die erste Version von Elasticsearch als Open-Source-Projekt unter der Apache-2.0-Lizenz. Viele der Funktionen basieren auf seinen Erfahrungen bei der Entwicklung des Lucene-Such-Frameworks Compass [3]. Auf den ersten Blick sieht Elasticsearch dabei ähnlich wie Apache Solr aus – es baut ebenfalls auf Lucene auf und stellt die Funktionalität per HTTP zur Verfügung. Im Detail gibt es jedoch große Unterschiede, etwa eine kompromisslose Fokussierung auf Verteilung. Durch die transparente Datenverteilung und die einfachen, modernen Bedienkonzepte wurde Elasticsearch schnell erfolgreich und ist mittlerweile nicht nur bei jüngeren Unternehmen wie GitHub oder Stackoverflow im Einsatz, sondern auch in konservativeren wie Banken oder unterschiedlichen amerikanischen Behörden. Seit 2012 existiert mit elastic (vormals Elasticsearch) ein Unternehmen, das Support und kommerzielle Erweiterungen für Elasticsearch anbietet.

Elasticsearch stellt viele unterschiedliche Funktionen zur Verfügung. Dazu gehören unterschiedliche Möglichkeiten zur Indizierung von Inhalten und der Suche darauf sowie unterstützende Funktionalitäten wie die Relevanzberechnung oder Möglichkeiten zur automatischen Vervollständigung von Inhalten. Einzelne Elasticsearch-Instanzen bilden einen Cluster und Daten können in einzelne Shards aufgeteilt werden, die auf unterschiedliche Knoten verteilt werden können. Durch die verteilte Natur ist es deutlich einfacher, als mit vielen anderen Systemen auch mit großen Datenmengen umzugehen, was beispielsweise für Anwendungsfälle wie das zentralisierte Logging wichtig ist. Aggregationen können schließlich neue Blickwinkel auf Daten ermöglichen.

Dieses Buch soll einen Einblick in die Nutzung von Elasticsearch geben und zeigen, wie der Suchserver für unterschiedliche Anwendungen zum Einsatz kommen kann. Der Start mit Elasticsearch fällt glücklicherweise ganz leicht, wie wir im folgenden Beispiel sehen, bei dem wir Elasticsearch installieren, Daten darin speichern und direkt durchsuchen.

1.3 Ein erstes Beispiel

Voraussetzung für die Installation von Elasticsearch ist lediglich eine aktuelle Java VM, die unter http://java.com für unterschiedliche Systeme heruntergeladen werden kann. Auf der Elasticsearch-Homepage stehen unter https://www.elastic.co/downloads/elasticsearch unterschiedliche Archive zur Verfügung, die direkt entpackt werden können. In diesem Buch wird durchgehend Elasticsearch in der Version 1.6.0 eingesetzt. Elasticsearch kann nach dem Entpacken über ein Skript gestartet werden[1].

```
wget https://download.elastic.co/elasticsearch
    ↪/elasticsearch/elasticsearch-1.6.0.zip[2]
unzip elasticsearch-1.6.0.zip
elasticsearch-1.6.0/bin/elasticsearch
```

Die Funktionalität steht über eine HTTP-Schnittstelle zur Verfügung. Ob die Anwendung gestartet wurde, sehen wir nicht nur an den Logausgaben, sondern auch durch einen Zugriff auf den Server per HTTP, etwa über das Kommandozeilenwerkzeug cURL.

```
curl -XGET "http://localhost:9200"
```

Elasticsearch und HTTP

Da HTTP mittlerweile allgegenwärtig ist, können unterschiedliche Werkzeuge für den Zugriff verwendet werden. In diesem Buch wird wie in vielen anderen Dokumentationsquellen der Kommandozeilen-Client cURL [4] verwendet, der für unterschiedliche Systeme zur Verfügung steht. Es ist jedoch auch problemlos eine Nutzung anderer Werkzeuge möglich, beispielsweise von Browser-Plugins wie Postman [5] für Chrome oder RESTClient [6] für Firefox.

Die Kommunikation mit Elasticsearch über HTTP wird oft als RESTful bezeichnet. REST ist ein Architekturstil, bei dem die Funktionalität einer Anwendung über Ressourcen abgebildet wird, die über HTTP angesprochen werden können. Ressourcen sind durch URIs identifizierbar und können unterschiedliche Repräsentationen haben. Zustände werden über Links zwischen den Ressourcen modelliert. [7]

Die Diskussion, ob die Schnittstellen von Elasticsearch als RESTful anzusehen sind, wird in diesem Buch nicht geführt. Wichtig ist, dass die Nutzung von HTTP zur Kommunikation einen einfachen Umgang mit Elasticsearch sowohl für den Betrieb als auch für die Entwicklung erlaubt.

[1] Im Buch wird Elasticsearch in einer Linux-Umgebung verwendet. Ein Betrieb ist jedoch auch problemlos unter Windows möglich.

[2] Lange Ein- und Ausgaben auf der Kommandozeile, die eigentlich durchgängig zu schreiben sind, werden aus Platzgründen im Buch gelegentlich auf mehrere Zeilen umbrochen. Dies ist dann mit dem Zeichen ↪ gekennzeichnet.

Sense
Gerade für Einsteiger kann neben der Nutzung eines einfachen Tools zur Kommunikation über HTTP auch der Einsatz von Marvel [8] eine Alternative sein, einem Monitoring-Plugin für Elasticsearch. Dieses integriert das Werkzeug Sense, mit dem komfortabel Abfragen formuliert werden können. Neben der kontextsensitiven Autovervollständigung von Query-Bestandteilen bietet das Werkzeug Syntax-Highlighting, eine History vergangener Abfragen und die Möglichkeit, Abfragen von und in cURL-Aufrufe umzuwandeln.

Bei Marvel handelt es sich um ein kostenpflichtiges Plugin, das allerdings zur Entwicklungszeit kostenlos genutzt werden kann. Auf `elasticsearch-buch.de` findet sich eine kurze Einführung zur Nutzung von Sense in Marvel.

Wenn die Anwendung erfolgreich gestartet wurde, werden einige Informationen angezeigt.

```
{
  "status" : 200,
  "name" : "Wolf",
  "cluster_name" : "elasticsearch",
  "version" : {
    "number" : "1.6.0",
    "build_hash" : "cdd3ac4dde4f69524ec0a14de3828cb95bbb86d0",
    "build_timestamp" : "2015-06-09T13:36:34Z",
    "build_snapshot" : false,
    "lucene_version" : "4.10.4"
  },
  "tagline" : "You Know, for Search"
}
```

Elasticsearch verwendet zur ein- und ausgehenden Kommunikation ausschließlich JSON-Dokumente, im Beispiel ein Dokument, das Informationen zum Server wie den Namen, die Version und Informationen zur verwendeten Lucene-Version zusammenfasst. Zur Speicherung der Daten können ebenfalls JSON-Dokumente verwendet werden, die beliebig aufgebaut sein können. Über eine POST-Anfrage kann ein neues Dokument hinzugefügt werden.

```
curl -XPOST "http://localhost:9200/example/doc" -d'
{
    "title": "Hallo Welt",
    "tags": ["example", "elasticsearch"]
}'
```

Diese gespeicherten Daten können im Anschluss sofort durchsucht werden.

```
curl -XGET "http://localhost:9200/_search?q=welt&pretty"
```

Als Ergebnis erhalten wir für den Suchbegriff *welt* unser gerade hinzugefügtes Dokument zurück.

```
{
  "took" : 231,
  [...],
  "hits" : {
    "total" : 1,
    "max_score" : 0.15342641,
    "hits" : [ {
      "_index" : "example",
      "_type" : "doc",
      "_id" : "AU3g6jiVOZtCV272fVut",
      "_score" : 0.15342641,
      "_source":{
    "title": "Hallo Welt",
    "tags": ["example", "elasticsearch"]
}

    } ]
  }
}
```

Wir haben in wenigen Schritten eine Möglichkeit geschaffen, unsere Daten durchsuchen zu können.

Formatierung der Ausgabe
Viele der JSON-Dokumente, die von Elasticsearch zurückgegeben werden, sind nicht formatiert und deshalb schwer lesbar. Über den optionalen Parameter `pretty`, der an die URL angehängt werden kann, wird die Rückgabe formatiert. Im Buch werden die Ausgaben oftmals formatiert dargestellt, der Parameter wird aus Gründen der Lesbarkeit aber nicht explizit angegeben.

1.4 Anwendungsfälle

Elasticsearch kann für die unterschiedlichsten Einsatzgebiete nützlich sein. Oftmals wenden Anwendungen unterschiedliche Aspekte an und kombinieren diese. Zu den wichtigsten Einsatzszenarien gehören die folgenden.

Volltextsuche Durch den Unterbau Apache Lucene ist Elasticsearch prädestiniert zur Volltextsuche. Dabei kann innerhalb von Texten nach einzelnen Schlüsselwörtern, Phrasen oder unvollständigen Begriffen gesucht werden. Gerade auf Webseiten oder bei Anwendungen, die große Datenmengen zur Verfügung stellen, kann die Volltextsuche ein elementares Werkzeug für die Nutzer sein.

Abfrage strukturierter Daten Neben der Suche in Textinhalten können in Elasticsearch Daten auch strukturiert abgefragt werden. Es werden unterschiedliche Datentypen für numerische Daten, Datumswerte und sogar Geodaten unterstützt, die auf unterschiedliche Art, auch kombiniert, abgefragt werden können.

Analytics Mit den Aggregationen hat Elasticsearch ein mächtiges Werkzeug für Analytics integriert. Damit können Besonderheiten und Gemeinsamkeiten in Datensätzen auch aus großen Datenmengen performant und trotzdem flexibel extrahiert werden. Durch die Sharding-Funktionalität können auch sehr große Datenmengen in Echtzeit gespeichert, durchsucht und aggregiert werden.

Verarbeitung hoher Leselast Elasticsearch kann durch die horizontale Skalierung gut mit vielen Leseanfragen umgehen. Daten können auf unterschiedliche Knoten repliziert und damit auch eine hohe Abfragelast verarbeitet werden.

Datenkonsolidierung Durch die einfache Dokumentenstruktur können Daten aus unterschiedlichen Quellen in Elasticsearch zusammengeführt und gemeinsam abgefragt werden.

Logfile-Analyse Durch die Verteilung unserer Anwendungen in unterschiedliche Prozesse und auf unterschiedliche Maschinen ist die Analyse im Fehlerfall oder auch die Kontrolle der Logdaten deutlich komplexer geworden. Elasticsearch ist vor allem in Verbindung mit den Werkzeugen Logstash und Kibana sehr beliebt als zentraler Datenspeicher für Log-Events.

1.5 Wann Elasticsearch?

Neben den unterschiedlichen Anwendungsfällen bietet Elasticsearch noch weitere Vorteile.

Einfacher Start Die Nutzung von Elasticsearch gestaltet sich auch für Einsteiger sehr einfach. Es ist wenig Konfiguration notwendig, Elasticsearch arbeitet in vielen Fällen mit sinnvollen Standardwerten.

Programmiersprachunabhängigkeit Durch die Nutzung von HTTP und JSON kann Elasticsearch aus so gut wie jeder Programmiersprache angesprochen werden. Zusätzlich existieren zahlreiche Client-Bibliotheken für die verbreitetsten Programmiersprachen.

Open-Source-Projekt Da Elasticsearch unter der bewährten Apache-Lizenz veröffentlicht wird, ist der Einsatz im kommerziellen Umfeld kein Problem. Zusätzlich kann der Quelltext genutzt werden, um die Funktionsweise zu verstehen und auch um eigene Fehlerbehebung oder Erweiterungen durchzuführen.

Eine Nutzung von Elasticsearch sollte in vielen Fällen keine Probleme darstellen. Jedoch sollen auch noch die folgenden Aspekte erwähnt werden, die unter Umständen für andere Systeme sprechen können.

Stärke durch Flexibilität Elasticsearch spielt seine Stärke bei den flexiblen Abfragen aus. Wenn immer nur über eine ID auf die Dokumente zugegriffen werden soll, können andere Systeme wie Apache Cassandra eine bessere Wahl sein.

Komplexität Elasticsearch bringt als von Grund auf verteiltes System inhärente Komplexität mit sich. Vieles davon ist vor dem Anwender versteckt, im Ernstfall muss man sich jedoch mit unterschiedlichen Eigenheiten auseinandersetzen. Wenn die Verteilung nicht benötigt wird, kann eine Nutzung anderer Systeme einfacher sein.

Projekt wird von einer Firma getrieben Im Gegensatz zu Projekten beispielsweise bei der Apache Software Foundation steht hinter Elasticsearch eine Firma, die die Entwicklung steuert. Dies muss kein Nachteil sein, das Unternehmen elastic stellt viele exzellente Entwickler zur Arbeit an Elasticsearch und auch an Apache Lucene ab. Potenziell hat man jedoch mit einem Projekt bei einer Stiftung mehr Sicherheit, was eine mögliche Einflussnahme in die Entwicklung betrifft.

Weniger gut geeignet für Daten mit hoher Änderungshäufigkeit Elasticsearch und das darunter liegende Lucene gewinnen viel der Geschwindigkeit dadurch, dass einmal geschriebene Daten möglichst nicht mehr verändert werden. Wenn durch die Anwendung bestimmt wird, dass sich bestehende Datensätze sehr häufig ändern sollen, können andere Systeme zur Speicherung besser geeignet sein.

Auch wenn Elasticsearch für viele Anwendungen empfohlen werden kann, ist je nach den eigenen Anforderungen eine Evaluierung unterschiedlicher Systeme sinnvoll.

1.6 Über dieses Buch

Dieses Buch soll den neugierigen Leser in die Grundlagen von Elasticsearch einführen. Dabei werden unterschiedliche Einsatzzwecke, vor allem die Volltextsuche, allerdings auch die Nutzung für Analytics oder als Datenspeicher für zentralisiertes Logging beleuchtet.

Zum Verständnis der Konzepte und Beispiele sind grundlegende Kenntnisse in der Softwareentwicklung und erste Erfahrungen in der Datenmodellierung, beispielsweise für relationale Datenbanksysteme, sinnvoll. Das Buch ist so gestaltet, dass die Kapitel in der angegebenen Reihenfolge gelesen werden. Für Leser, die sich bereits mit Elasticsearch auskennen, können auch einzelne Kapitel als Referenz dienen.

Kapitel 2 zeigt, wie klassischerweise eine Suchanwendung entsteht. Dabei werden unterschiedliche Konzepte wie Indizierung, Mapping, Facettierung über Aggregationen und die Query-DSL erläutert.

Kapitel 3 widmet sich dem Umgang mit Texten. Dabei wird auf unterschiedliche Mechanismen eingegangen, die viel zum Eindruck einer intelligenten Suche beitragen. Einerseits werden der Umgang mit unterschiedlichen Sprachen als auch andererseits nützliche Funktionen wie die Hervorhebung vom Suchbegriff oder Autovervollständigung erläutert.

Kapitel 4 hat die Relevanzberechnung zum Thema, ein großes Unterscheidungsmerkmal zu datenbankbasierten Systemen. Gerade bei großen Datenmengen kann ein Nutzer oftmals sehr viele Ergebnisse erhalten – die Relevanzsortierung hilft dann, die besten Treffer direkt nach vorne zu sortieren. Über unterschiedliche Mechanismen wie Boosting oder die Function-Score-Query kann die Relevanzsortierung auch für eigene Anwendungen beeinflusst werden.

Kapitel 5 In Kapitel 5 geht es um die Indizierung. Dabei werden einige Strategien und Mechanismen zur Anbindung von externen Datenquellen erläutert. Um zu verstehen, wie und wann Dokumente persistiert werden, ist ein Exkurs in einige Lucene-Interna notwendig.

Kapitel 6 dreht sich um alle Aspekte, die für die Verteilung der Daten wichtig sind. Im Zentrum stehen Shards und Replicas und wie diese auf die unterschiedlichen Knoten im Cluster verteilt werden.

Kapitel 7 erläutert unterschiedliche Mechanismen, wie Daten für Elasticsearch modelliert werden. Dazu gehören die elementaren Datentypen, die Gestaltung der Indexstruktur und die Verwaltung von Beziehungen zwischen Inhalten.

Kapitel 8 stellt die unterschiedlichen Aggregationen vor, die genutzt werden können, um Daten anhand bestimmter Eigenschaften zusammenzufassen und um Berechnungen darauf durchzuführen.

Kapitel 9 beschreibt den Zugriff auf Elasticsearch und die Nutzung des Java- und JavaScript-Clients.

Kapitel 10 hat einige für den Betrieb wichtige Themen zum Inhalt. Dazu gehören einerseits Überlegungen, die vor einer Produktivnahme anstehen, als auch Themen wie Monitoring oder Datensicherung.

Kapitel 11 widmet sich schließlich der populären Nutzung von Elasticsearch zur Speicherung von Logdaten mittels der Werkzeuge Logstash und Kibana, die zusammen den ELK-Stack ausmachen. Zusätzlich wird die Alternative Graylog vorgestellt, eine integrierte Lösung, die ebenfalls auf Elasticsearch zur Datenspeicherung setzt.

Kapitel 12 gibt einen Ausblick in die Zukunft von Elasticsearch und stellt in Kürze einige Themen vor, auf die im Buch nicht eingegangen werden konnte.

In den Anhängen finden sich schließlich noch Informationen zu einem Vorgehen zum Neuindizieren von Dokumenten und Hinweise zur Installation des Twitter-Rivers.

Die Beispiele und Erläuterungen in diesem Buch bauen auf der Elasticsearch-Version 1.6.0 auf. Aus Platzgründen musste die Darstellung von Ein- und Ausgaben teilweise umbrochen und abgekürzt werden. Dadurch sind die Codestücke

eventuell nicht immer so wie sie sind, lauffähig. Alle Beispiele in ihrer Originalform, weitere Informationen und mögliche Korrekturen finden sich auf der Homepage elasticsearch-buch.de.

In den Kapiteln 2–5 sind einzelne Aufgaben verstreut, die mit »Zum Selbermachen« markiert sind und direkt zum praktischen Mitarbeiten anregen sollen. Hinweise und mögliche Lösungen zu diesen Aufgaben finden sich ebenfalls auf elasticsearch-buch.de.

1.7 Danksagung

Dieses Buch konnte nur entstehen, weil ich auf die Hilfe vieler Leute zurückgreifen konnte. An erster Stelle möchte ich meinen Reviewern danken, die wertvollen Input zum Manuskript in unterschiedlichen Stadien gegeben haben. Vielen Dank an Tobias Kraft, Andreas Jägle, Bernd Fondermann, Eberhard Wolff, Jochen Schalanda und Dr. Patrick Peschlow.

Vielen Dank auch an das Team beim dpunkt.verlag um meinen Lektor René Schönfeldt und an Niko Köbler, der dafür gesorgt hat, dass ich dieses Buch schreiben konnte.

Ebenfalls vielen Dank an all die Leute, die ihr Wissen in Blogposts, auf Konferenzvorträgen oder im direkten Austausch weitergeben und damit indirekt mitgeholfen haben, dass dieses Buch entstehen konnte. Ohne Anspruch auf Vollständigkeit gehören dazu Alexander Reelsen (der mich auch ursprünglich von Elasticsearch überzeugt hat), Britta Weber, Adrien Grand, Uwe Schindler, Clinton Gormley, Christian Uhl, Jörg Prante, Andrew Cholakian, Michael McCandless, Zachary Tong, Lucian Precup und Alex Brasetvik.

Vielen Dank auch an alle Entwickler hinter Elasticsearch und Apache Lucene, die es ermöglichen, dass die Software für uns alle verfügbar ist.

Zum Schluss noch vielen Dank an meine Eltern, die mich jahrelang unterstützt haben, und vor allem an Lianna, die während der Entstehung dieses Buches oft auf mich verzichten musste, mir aber immer eine Hilfe war.

2 Eine Suchanwendung entsteht

Nachdem die Einführung einen ersten Blick auf Elasticsearch ermöglicht hat, gehen wir in diesem Kapitel direkt zur Praxis über. Wir implementieren eine typische Suchanwendung von Grund auf und lernen dabei wichtige Eigenschaften von Elasticsearch, wie die Indizierung, unterschiedliche Abfragen und die Möglichkeit zur Facettierung kennen.

2.1 Die Beispielanwendung

Wir werden viele Eigenschaften von Elasticsearch an einem durchgehenden Beispiel ansehen, das uns in diesem und den nächsten Kapiteln begleiten wird: eine einfache Anwendung zur Suche nach Vortragsbeschreibungen. Wir können uns vorstellen, dass ein Konferenzveranstalter ein Archiv zur Verfügung stellen will, in dem Details zu den Inhalten der Vorträge und den Metadaten wie Sprecher oder Datum zur Verfügung stehen. Die Benutzer können damit für sie interessante Vorträge finden.

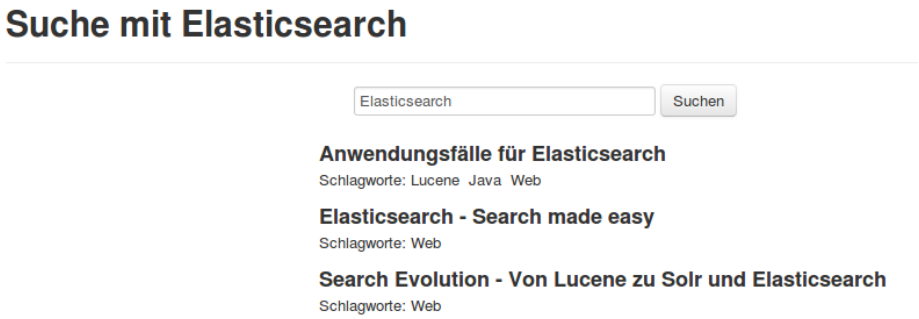

Abbildung 2-1: Screenshot einer klassischen Suchanwendung

In der Einführung haben wir bereits gesehen, wie einfach Elasticsearch installiert werden kann. Kapitel 10 bietet noch weitere Details dazu.

In einer ersten Iteration unserer Anwendung können wir uns eine einfache Suchseite mit Eingabefeld vorstellen, in das der Nutzer einzelne Schlüsselwörter eingeben kann. In einer Liste werden die zugehörigen Vorträge mit einigen ihrer Informationen angezeigt. Abbildung 2-1 zeigt, wie die Anwendung in einer ersten Iteration aussehen könnte.

2.1.1 Die Beispieldaten

Elasticsearch akzeptiert zu speichernde Daten in Form von JSON-Dokumenten. Woher diese Daten ursprünglich kommen, spielt zuerst einmal keine Rolle. Vorerst wird davon ausgegangen, dass die Vortragsdaten bereits im richtigen Format vorliegen. Ein erstes Beispiel kann folgendermaßen aussehen. Die Struktur wird im Laufe des Buches noch erweitert.

```
{
    "title" : "Anwendungsfälle für Elasticsearch",
    "speaker" : "Florian Hopf",
    "date" : "2014-07-17T15:35:00.000Z",
    "tags" : ["Java", "Lucene"],
    "conference" : {
        "name" : "Java Forum Stuttgart",
        "city" : "Stuttgart"
    }
}
```

Wir sehen, dass das Dokument unterschiedliche Eigenschaften hat. Manche davon sind Texte, wie `title` oder `speaker`. Das Datumsfeld `date` ist zwar ein JSON-String, ist allerdings in einem standardisierten Datumsformat hinterlegt. Die Schlagwörter zum Vortrag in dem Feld `tags` sind ein Array, die Konferenz, auf der der Vortrag stattfindet, ist schließlich noch als Subdokument hinterlegt.

Elasticsearch kann mit JSON dieser Art umgehen, das Dokument kann wie es ist gespeichert werden.

Was die Datenmenge angeht, werden wir uns in den ersten Kapiteln auf einige wenige Dokumente beschränken, da die Funktionalität im Vordergrund steht. Was zu beachten ist, wenn die Menge der Daten ansteigt, werden wir in späteren Kapiteln genauer betrachten.

2.2 Dokumente indizieren

Der erste Schritt zu einer Suchlösung besteht darin, unsere Dokumente in Elasticsearch zu speichern. Dazu ist es wichtig, die beiden Konzepte Index und Typ zu verstehen.

2.2.1 Index und Typ

Ein Index in Elasticsearch bildet eine logische Einheit zur Sammlung von Dokumenten. Welche Dokumente das sind, wird von der Anwendung bestimmt. In unserem Fall verwendet die Beispielapplikation nur den Index `conference` zur Speicherung aller Dokumente zu einer Konferenz. Wenn wir das Konzept mit der relationalen Welt vergleichen, entspricht ein Index einer Datenbank, die ebenfalls eine logische Sammlung von Entitäten ist. In einigen Bereichen weicht Elasticsearch hier jedoch deutlich ab, beispielsweise kann man mehrere Indizes gleichzeitig abfragen, was bei relationalen Datenbanken normalerweise nicht möglich ist.

Der Typ eines Dokuments beschreibt die Struktur der indizierten Dokumente. Für jede eigenständige Einheit, die wir im Index speichern wollen, existiert ein Typ. Für unsere Anwendung starten wir mit dem Typ `talk`, der Informationen zu den Vorträgen enthält. Ein weiterer möglicher Typ könnte `speaker` für Informationen zur Sprecherin sein. Wenn wir unseren Vergleich mit der relationalen Welt fortführen, sind die Typen vergleichbar mit Tabellen, allerdings gibt es auch hier im Detail starke Abweichungen.

Abbildung 2-2 visualisiert die Beziehung zwischen Index und Typ in einem Elasticsearch-Cluster mit dem Index `conference` zur Speicherung der Konferenzdaten und dem beispielhaften Index `interaction` zur Speicherung der Kommentare und Benutzerwertungen, die auf einer Homepage abgegeben werden können.

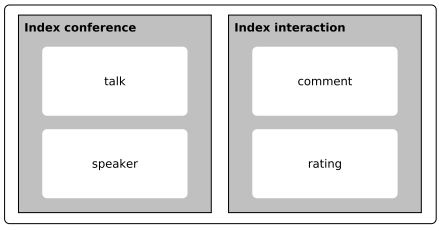

Abbildung 2-2: Mehrere Indizes und Typen in einer Elasticsearch-Instanz

Wie in Kapitel 1 erwähnt, erfolgt der Zugriff auf Elasticsearch meist über HTTP. Die URL enthält dann Informationen zum zu verwendenden Index im ersten Pfadsegment und zum Typ im zweiten Segment. Die gesamte URL entspricht dem Schema `http://{host}:{port}/{index}/{typ}`, also beispielsweise http://localhost:9200/conference/talk/.

Zugriff auf Elasticsearch
Auch wenn der Zugriff über HTTP es nahelegt: Elasticsearch sollte nur in Ausnahmefällen direkt im Internet verfügbar sein. In Kapitel 9 und 10 werden einige Aspekte zur Produktivnahme von Elasticsearch beschrieben.

2.2.2 Indizierung

Wie wir schon in der Einführung gesehen haben, können wir neue Dokumente einfach per HTTP-POST-Request hinzufügen. Index und Typ werden automatisch angelegt, wenn sie noch nicht existieren. Wird eine eigene ID vergeben, beispielsweise eine bestehende Datenbank-ID, können wir ein neues Dokument auch über einen PUT-Request anlegen und die ID mit in der URL übergeben.

```
curl -XPUT "http://localhost:9200/conference/talk/1" -d'
{
    "title" : "Anwendungsfälle für Elasticsearch",
    "speaker" : "Florian Hopf",
    "date" : "2014-07-17T15:35:00.000Z",
    "tags" : ["Java", "Lucene"],
    "conference" : {
        "name" : "Java Forum Stuttgart",
        "city" : "Stuttgart"
    }
}'
```

Wird keine ID angegeben, kann Elasticsearch diese generieren. Der vergebene Wert ist in der Antwort des Indizierungsaufrufs enthalten. Anhand der ID kann per GET-Request auf das Dokument zugegriffen werden.

```
curl -XGET "http://localhost:9200/conference/talk/1"
```

Für viele Anwendungen, in denen Elasticsearch nur zur Suche eingesetzt wird, ist der Zugriff anhand der ID eher selten. Wir können dadurch jedoch direkt sehen, dass unser Dokument vorhanden ist.

2.2.3 Die erste Suche

Wie mittlerweile üblich, soll unseren Benutzern der Zugriff auf die indizierten Vorträge per Suche nach Schlüsselwort möglich sein – die Stärke von Elasticsearch. Eine Nutzerin kann einen oder mehrere Begriffe in das Eingabefeld eingeben und wenn der Begriff im Dokument vorkommt, soll der Vortrag als Treffer in der Liste auftauchen.

Elasticsearch stellt unterschiedliche Suchmöglichkeiten zur Verfügung. Im einfachsten Fall wird ein GET-Request auf den `_search`-Endpunkt ausgeführt. Der Parameter `q` wird verwendet, um den Suchbegriff des Nutzers zu übergeben. Die folgende Abfrage fordert alle Dokumente an, die den Begriff *Elasticsearch* enthalten.

```
curl -XGET "http://localhost:9200/conference/talk/_search?q=Elasticsearch"
```

Elasticsearch schickt uns auf diese Anfrage eine ausführliche Antwort, die auch unser ursprünglich indiziertes Dokument enthält.

```
{
   "took": 35,
   "timed_out": false,
   "_shards": {
      "total": 5,
      "successful": 5,
      "failed": 0
   },
   "hits": {
      "total": 1,
      "max_score": 0.067124054,
      "hits": [
         {
            "_index": "conference",
            "_type": "talk",
            "_id": "1",
            "_score": 0.067124054,
            "_source": {
               "title": "Anwendungsfälle für Elasticsearch",
               "speaker": "Florian Hopf",
               "date": "2014-07-17T15:35:00.000Z",
               "tags": [
                  "Java",
                  "Lucene"
               ],
               "conference": {
                  "name": "Java Forum Stuttgart",
                  "city": "Stuttgart"
               }
            }
         }
      ]
   }
}
```


Im Kopfbereich sehen wir einige Metainformationen wie die Dauer der Bearbeitung in Millisekunden und ob alle Shards erreicht werden konnten.

Shards
Shards unterteilen die Daten eines Elasticsearch-Index in einzelne Datentöpfe, die auch auf unterschiedliche Elasticsearch-Knoten verteilt werden können. Standardmäßig wird ein Index in fünf Shards aufgeteilt und indizierte Dokumente werden darauf verteilt. Die Shards und alles, was mit Verteilung zusammenhängt, sind Thema von Kapitel 6.

Im Subdokument `hits` finden wir Informationen zu unseren Treffern. Der Wert `total` gibt die Gesamtzahl der Treffer im Index an. Im inneren `hits`-Array finden sich dann schließlich die Dokumente, die durch die Suche gefunden wurden, sortiert nach der Relevanz. Wie wir sehen können, steht der gesamte Inhalt des Dokuments in dem Feld `_source` zur Verfügung.

Zum Selbermachen

Indizieren Sie weitere Dokumente für Vorträge. Was passiert, wenn einzelne Felder im Dokument nicht enthalten sind? Können die Werte aller Felder durch eine Suche gefunden werden?

Gewappnet mit dem Wissen über den Zugriff auf Elasticsearch und der einfachen Suchmöglichkeit können wir die Anwendung mit der grundlegenden Suchfunktionalität ausstatten. Da in der Antwort von Elasticsearch das gesamte indizierte Dokument enthalten ist, können wir dem Benutzer Informationen zu den Ergebnissen darstellen. Im folgenden Abschnitt sehen wir, welche Mechanismen Elasticsearch im Hintergrund verwendet, um die bisher gesehene Funktionalität zu ermöglichen.

2.3 Der invertierte Index

Im einführenden Kapitel wurde bereits erwähnt, dass Elasticsearch auf der Bibliothek Lucene aufbaut. Lucene stellt mehrere Datenstrukturen zur Verfügung, unter anderem den zentralen invertierten Index, der auch die Grundlage für unsere bisherige Suche mit Elasticsearch bildet.

Der invertierte Index bildet Suchbegriffe auf Dokumente ab. Wenn beispielsweise zwei Dokumente indiziert werden, Dokument 1 mit dem Titel *Anwendungsfälle für Elasticsearch* und Dokument 2 mit dem Titel *Elasticsearch – Die verteilte Suchmaschine in der Praxis*, wird mit den Standardeinstellungen ein Index ähnlich zu dem in Tabelle 2-1 angegebenen verwendet.

Jeder enthaltene Term ist auf die Nummer des Dokuments abgebildet, in dem er enthalten ist.

Bei einer Suche wird für den entsprechenden Begriff das Dokument direkt nachgeschlagen, beispielsweise Dokument 2 für eine Suche nach *praxis*. Das Prinzip entspricht dem Stichwortverzeichnis in einem Buch, in dem Begriffe auf die zugehörige Seitenzahl abgebildet werden.

Neben der Information, in welchem Dokument ein Begriff enthalten ist, sind meist noch einige zusätzliche Informationen im Index gespeichert, beispielsweise die Häufigkeit eines Terms und die Position im Dokument.

Term	Dokument
anwendungsfälle	1
der	2
die	2
elasticsearch	1,2
für	1
in	2
praxis	2
suchmaschine	2
verteilte	2

Tabelle 2-1: Zuordnung von Termen auf Dokumente in einem invertierten Index

2.3.1 Analyzing

Um die passenden Suchbegriffe auf die Dokumente abzubilden, in denen sie auftauchen, muss der Originalinhalt eines Dokuments in die Struktur für den Index überführt werden. Wenn ein Feld aus Zeichenketten besteht, werden die Terme für den Index über den Analyzing-Prozess extrahiert. In unserem Beispiel wurde der Titel bei der Überführung in den Index an den Leerzeichen aufgespalten. Dadurch kann nach einzelnen Worten im Satz gesucht werden. Zusätzlich wurde jeder Term in Kleinbuchstaben umgewandelt. Dadurch kann eine Suche nach *elasticsearch* den ursprünglich großgeschriebenen Begriff auffinden.

Der konfigurierbare Analyzing-Prozess besteht, wie auch im Beispiel, oft aus mehreren Schritten, die den Text jeweils auf unterschiedliche Weise verarbeiten. Ohne weitere Konfiguration kommt der Standard-Analyzer zum Einsatz, der für westliche Sprachen geeignet ist und Sätze anhand von Leerzeichen und Satztrennzeichen aufspaltet. Zusätzlich werden die Terme wie schon gesehen in Kleinbuchstaben umgewandelt.

Bei einer Suche wird derselbe oder ein kompatibler Analyzing-Prozess auch für den Suchbegriff durchgeführt. Dadurch spielt dann wie im Beispiel Groß- und Kleinschreibung weder im Dokument noch im Suchbegriff eine Rolle. Den Zusammenhang zwischen Analyzing während der Indizierung und der Suche zeigt Abbildung 2-3.

Die Konfiguration des Analyzing erfolgt über das in Abschnitt 2.5 vorgestellte Mapping eines Typs. Wenn wie bisher kein eigenes Mapping erstellt wurde, erstellt Elasticsearch ein passendes aus den indizierten Daten. Neben dem Aufsplitten der Texte und der einfachen Umwandlung wird das Analyzing auch für viele weitere Prozesse verwendet. In Kapitel 3 setzen wir uns ausführlicher damit auseinander.

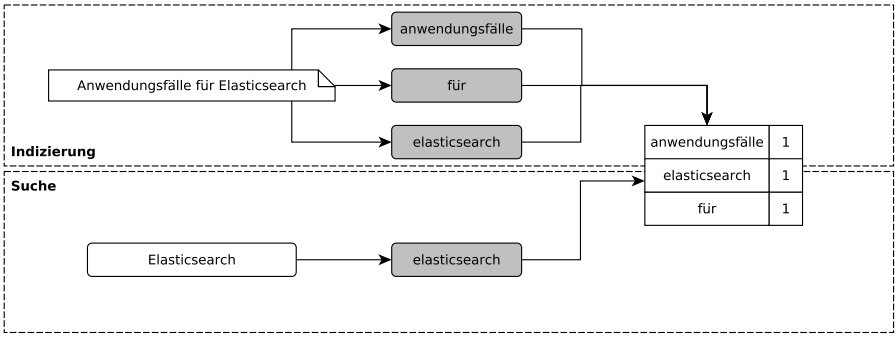

Abbildung 2-3: Analyzing findet zur Index- und zur Suchzeit statt.

2.3.2 Speicherung von Feldern

Die Dokumente, die wir indizieren, bestehen meist aus mehreren Feldern. Da in den Feldern unterschiedliche Daten indiziert werden sollen und eine Suche auch auf ein Feld einschränkbar sein soll, werden die Inhalte jedes Feldes in einer eigenen Indexstruktur vorgehalten. Für jedes Feld kann einzeln entschieden werden, ob und wie es in den Index aufgenommen werden soll. Zusätzlich kann jedes Feld einen anderen Typ haben und anders verarbeitet werden, um bestimmte Suchen zu ermöglichen.

Auch wenn wir in unserem Eingabedokument mehrere Felder hinterlegt haben, wurde bei unserer einfachen Suche bisher lediglich ein Feld von Elasticsearch verwendet, das `_all`-Feld als Standardfeld für die Suche. Es speichert die Daten aller anderen Felder und enthält somit den kompletten Inhalt des Dokuments. Wenn über den `q`-Parameter einfache Terme angegeben werden, wird standardmäßig in diesem Feld gesucht.

Interne Felder
Alle mit einem Unterstrich beginnenden Felder und URL-Bestandteile beschreiben interne Elasticsearch-Funktionalitäten.

2.3.3 Auf Feldern suchen

Standardmäßig legt Elasticsearch neben dem `_all`-Feld bereits alle Felder eines Eingabedokuments einzeln an. Es gibt also in unserem Fall schon ein Feld `title` im Index, das wir durchsuchen können.

Der `q`-Parameter, den wir Elasticsearch bei der ersten Suche übergeben haben, akzeptiert Abfragen, die an die Lucene-Query-Syntax angelehnt sind [9]. Diese spezialisierte Abfragesprache unterstützt Features wie Wildcards und die Verknüpfung von unterschiedlichen Abfragen über boolesche Operatoren. Zu-

sätzlich kann angegeben werden, dass nur auf einem bestimmten Feld gesucht werden soll, indem es vor dem Suchbegriff angegeben wird. Um die Suche auf das `title`-Feld einzuschränken, kann beispielsweise *title:elasticsearch* verwendet werden. Dadurch wird statt des `_all`- das `title`-Feld durchsucht.

```
curl -XGET "http://localhost:9200/conference/talk
    ↪/_search?q=title:elasticsearch"
```

Neben der Suche auf einem direkt im Dokument enthaltenen Feld können wir auch auf Felder in Subdokumenten zugreifen, beispielsweise kann im Namen der Konferenz über *conference.name:java* gesucht werden. Um auf mehreren Feldern zu suchen, könnten wir einfach jedes Feld angeben, auf dem wir suchen wollen: *title:Elasticsearch tags:Elasticsearch*.

Da für Textinhalte der Analyzing-Prozess auch für den Suchbegriff angewendet werden muss, führt Elasticsearch diesen je nach Feld automatisch durch.

Auch wenn einige wichtige Anwendungen unterstützt werden, reicht die Suche über den Query-Parameter nur bedingt aus. Für eine moderne Suchanwendung sollen oft weitere Features wie das Hervorheben von Texten oder die Filterung von Ergebnismengen möglich sein. Die Übergabe mittels Parameter führt dabei zu schwer lesbaren Abfragen. Deshalb setzt Elasticsearch stattdessen auf die ausdrucksstarke Query-DSL.

2.4 Über die Query-DSL zugreifen

Die Steuerung des Verhaltens über Parameter eignet sich für einfache Abfragen gut. Da jedoch auch komplexe Anfragen möglich sein sollen, bietet Elasticsearch eine Alternative: Die Formulierung der Abfragen über die Query-DSL, eine JSON-Struktur, die über den Request-Body übergeben wird. Unsere eingangs verwendete Suche kann mit der Query-DSL folgendermaßen abgebildet werden.

```
curl -XPOST "http://localhost:9200/conference/talk/_search" -d'
{
    "query": {
        "query_string": {
           "query": "Elasticsearch"
        }
    }
}'
```

Wir senden nun einen POST-Request statt eines GET-Requests[1] und übergeben unsere Abfrage im Body des Requests. Eingeleitet wird sie mit dem `query`-Schlüsselwort. Über `query_string` geben wir an, dass wir wie im vorherigen Beispiel die Lucene-Query-Syntax verwenden. Der eigentliche Suchbegriff wird über das Feld `query` übergeben.

[1] Der Zugriff funktioniert auch über GET, POST ist allerdings der empfohlene Weg.