entwickler.press
shortcuts

Elastic-search

Clientnutzung, Datenvisualisierung und Apache Lucene

Florian Hopf, Stefan Siprell, Achim Nierbeck, Uwe Schindler

Elasticsearch geschuldet und kein Hinweis auf den HTTP-GET-Request: execute() liefert ein Future-Objekt zurück, von dem dann der eigentliche Wert angefordert werden kann.

Auch für die interne Kommunikation, die nicht über die REST-Schnittstelle erfolgt, nutzt Elasticsearch JSON. Der Großteil der Funktionalität ist zwar über den Client und eigene Builder verfügbar, es kann allerdings auch der generische jsonBuilder() verwendet werden, der beliebige JSON-Strukturen erstellen kann.

Eine Alternative zum NodeClient, den wir bis jetzt gesehen haben, ist der TransportClient. Wird dieser

verwendet, tritt der Knoten nicht dem Cluster bei, sondern verbindet sich nur mit diesem mittels des Transportmoduls, das auch für die Kommunikation zwischen den Knoten verwendet wird. Die Verwendung kann sinnvoll sein, wenn die Verwaltung des Clusterzustands in der Anwendung problematisch ist, beispielsweise wenn besondere Anforderungen bezüglich des Speicherverbrauchs bestehen oder wenn die Anwendung häufig neu gestartet wird.

Ein TransportClient erwartet, wie in Listing 1.4 zu sehen, bei der Erzeugung einen oder mehrere URLs

zu Knoten im Cluster. Wenn der Konfigurationsparameter client.transport.sniff gesetzt ist, kann der TransportClient auch automatisch alle weiteren Knoten des Clusters auslesen und diese mittels eines Round-Robin-Verfahrens verwenden.

```
Client client = new TransportClient()
        .addTransportAddress(new InetSocketT
        .addTransportAddress(new InetSocketT
```

Listing 1.4

Der Standardclient ist die richtige Lösung, wenn man alle Features von Elasticsearch verfügbar haben will. Neue Funktionalität ist mit jedem Release sofort verfügbar. Der NodeClient kann Anfragen intelligent

verteilen, während der TransportClient Anfragen an unterschiedliche Knoten des Clusters schickt.

Zu beiden Clients und der grundlegenden Funktionsweise gibt es neben der Dokumentation in der Elasticsearch-Referenz auch weitere Informationen unter [2] und [3]. Bei der Verwendung gilt zu beachten, dass in der Anwendung Elasticsearch in derselben Version verwendet werden sollte wie im Cluster. Ein Upgrade der Elasticsearch-Version erfordert dann auch ein Upgrade in der Anwendung.

Jest

Eine leichtgewichtigere Alternative, bei der weniger Probleme mit Versionsinkompatibilitäten auftreten können, ist Jest [4], eine Implementierung des Elasticsearch-REST-API per Apache HttpComponents.

Das API von Jest verhält sich ähnlich wie das Elasticsearch-API. Über ein Fluent-API mit spezialisierten Buildern steht die Funktionalität von Elasticsearch zur Verfügung. Jegliche Interaktion wird über den JestClient abgewickelt, der, wie in Listing 1.5 zu sehen, über eine Factory erzeugt werden kann.

```
JestClientFactory factory = new JestClientFa
```