elasticsearch.

und die Java-Welt

Florian Hopf
@fhopf

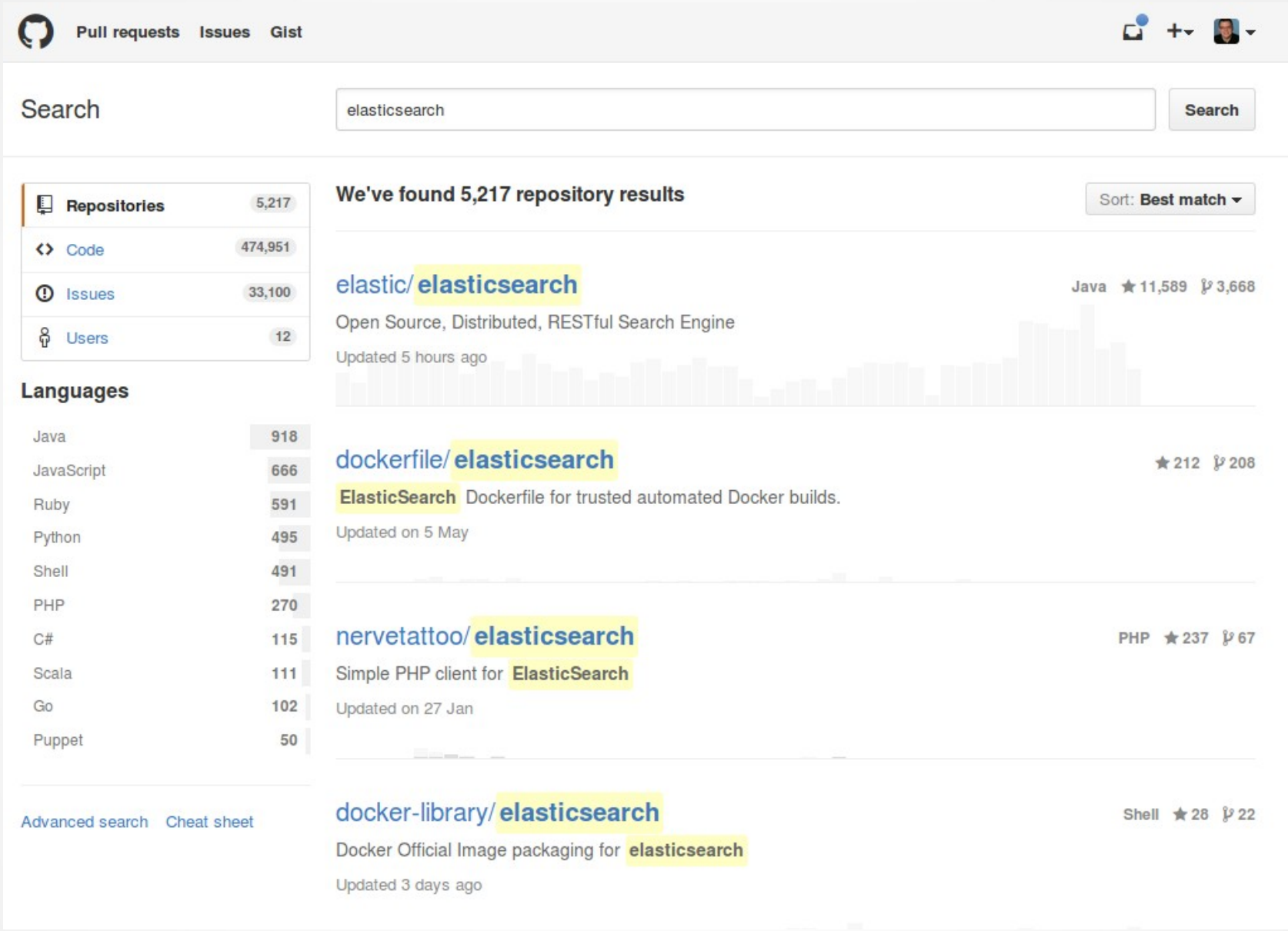
Elasticsearch?

„Elasticsearch is a distributed, open source search and analytics engine, designed for horizontal scalability, reliability, and easy management.“

Elasticsearch?

*„Elasticsearch is a distributed, open source **search and analytics engine**, designed for horizontal scalability, reliability, and easy management.“*

Elasticsearch?



Installation

```
# download archive
wget https://download.elastic.co/elasticsearch
                    /elasticsearch/elasticsearch-1.6.0.zip

# zip is for windows and linux
unzip elasticsearch-1.6.0.zip

# on windows: elasticsearch.bat
elasticsearch-1.6.0/bin/elasticsearch
```

Zugriff per HTTP

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

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

Indizierung

```
curl -XPOST "http://localhost:9200/library/book" -d'
{
    "title": "Elasticsearch in Action",
    "author": [ "Radu Gheorghe",
                "Matthew Lee Hinman",
                "Roy Russo" ],
    "published": "2015-06-30T00:00:00.000Z",
    "publisher": {
        "name": "Manning",
        "country": "USA"
    }
}'
```

Suche

```
curl -XPOST "http://localhost:9200/library/book/_search
              ?q=elasticsearch"

{
   [...]
   "hits": {
      "hits": [
         {
            "_index": "library",
            "_type": "book",
            "_source": {
               "title": "Elasticsearch in Action",

            [...]
```

Suche per Query DSL

```
curl -XPOST "http://localhost:9200/library/book/_search"
-d'
{
    "query": {
        "filtered": {
            "query": {
                "match": {
                    "title": "elasticsearch"
                }
            },
            "filter": {
                "term": {
                    "publisher.name": "manning"
                }
            }
        }
    }
}'
```

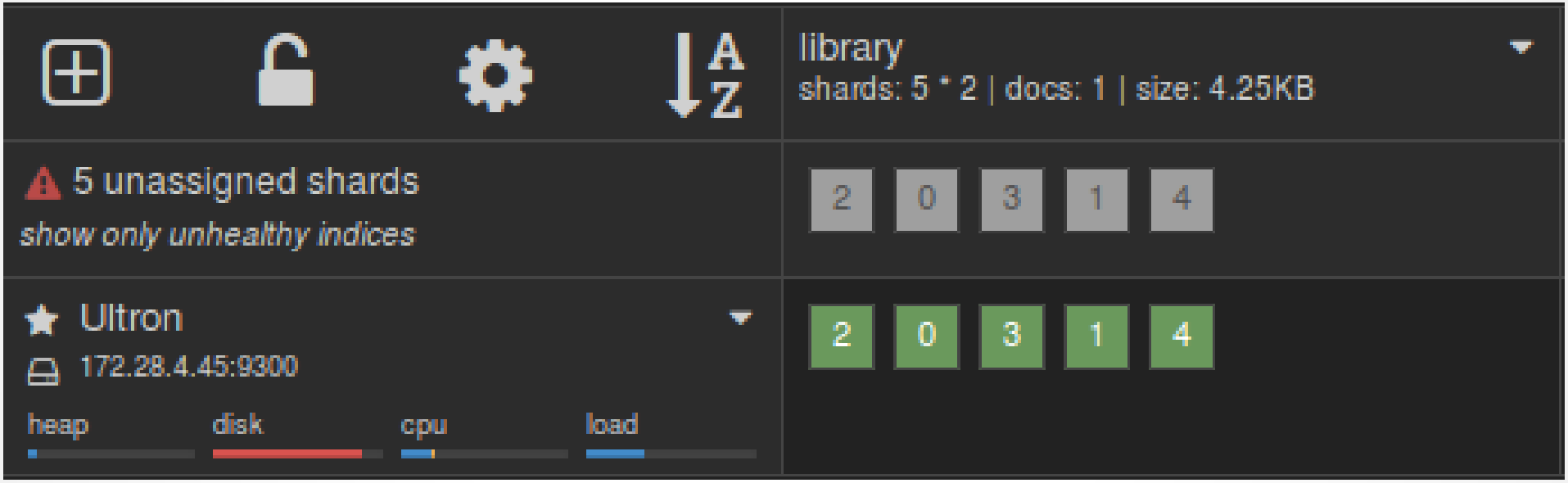
Recap

- Auf Java basierender Suchserver
- HTTP und JSON
- Suche und Filterung über Query-DSL
- Facettierung über Aggregations
- Highlighting, Suggestions, ...

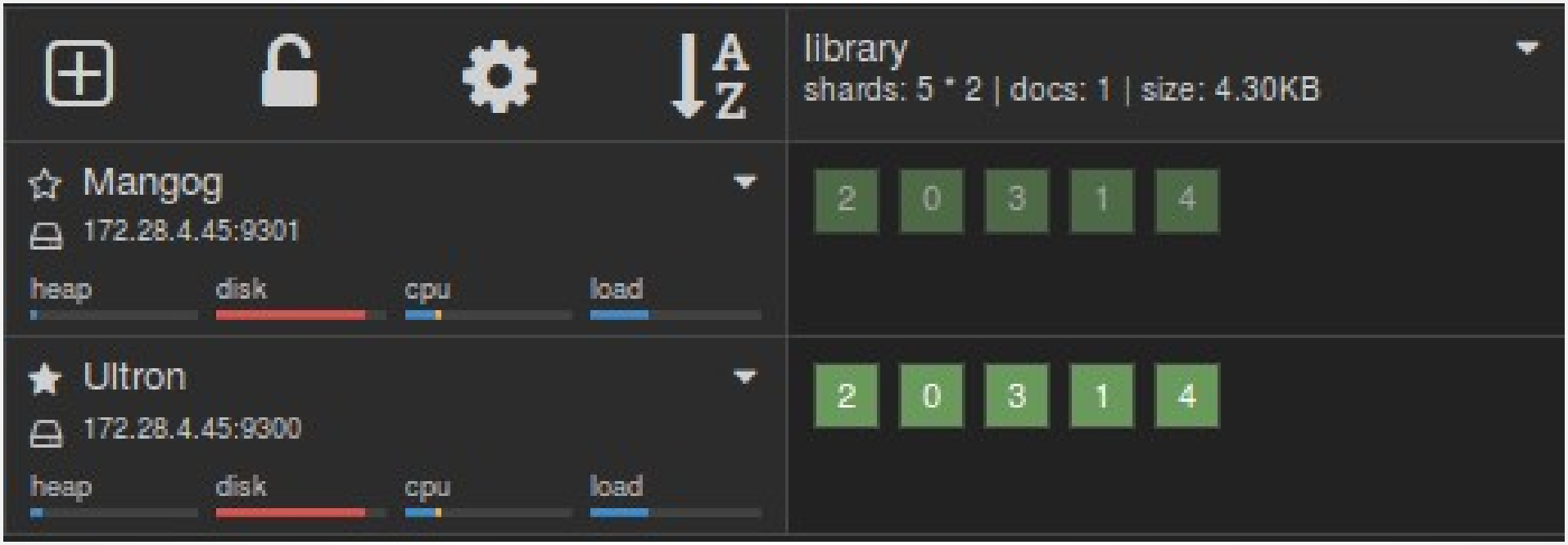
Elasticsearch?

*„Elasticsearch is a **distributed**, open source search and analytics engine, **designed for horizontal scalability, reliability**, and easy management.“*

Verteilung


library
shards: 5 * 2 | docs: 1 | size: 4.25KB
5 unassigned shards
show only unhealthy indices
2
0
3
1
4
Ultron
172.28.4.45:9300
heap
disk
cpu
load
2
0
3
1
4

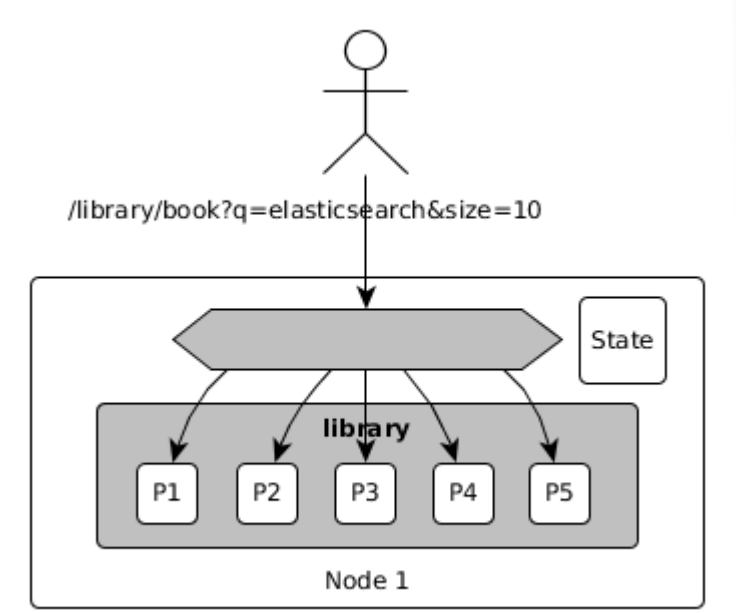
Verteilung


library
shards: 5 * 2 | docs: 1 | size: 4.30KB
Mangog
172.28.4.45:9301
heap
disk
cpu
load
2
0
3
1
4
Ultron
172.28.4.45:9300
heap
disk
cpu
load
2
0
3
1
4

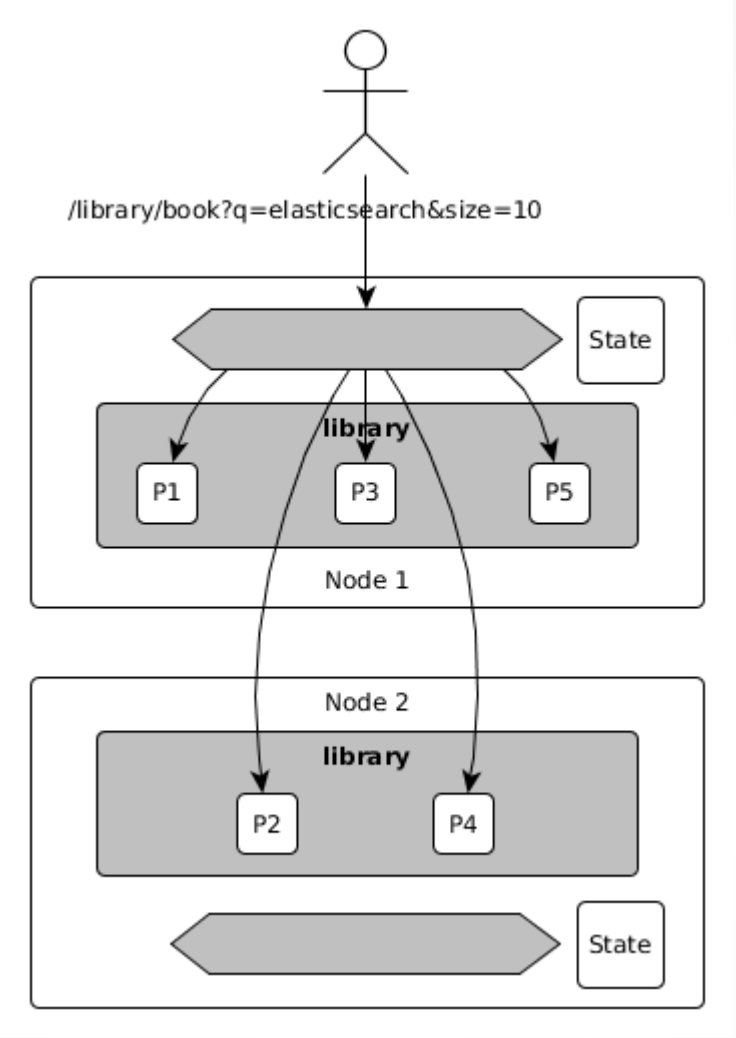
Verteilung


library
shards: 5 * 2 | docs: 1 | size: 4.26KB
Mangog
172.28.4.45:9301
heap
disk
cpu
load
4
3
1
Ultron
172.28.4.45:9300
heap
disk
cpu
load
4
0
3
2
Unseen
172.28.4.45:9302
heap
disk
cpu
load
0
1
2

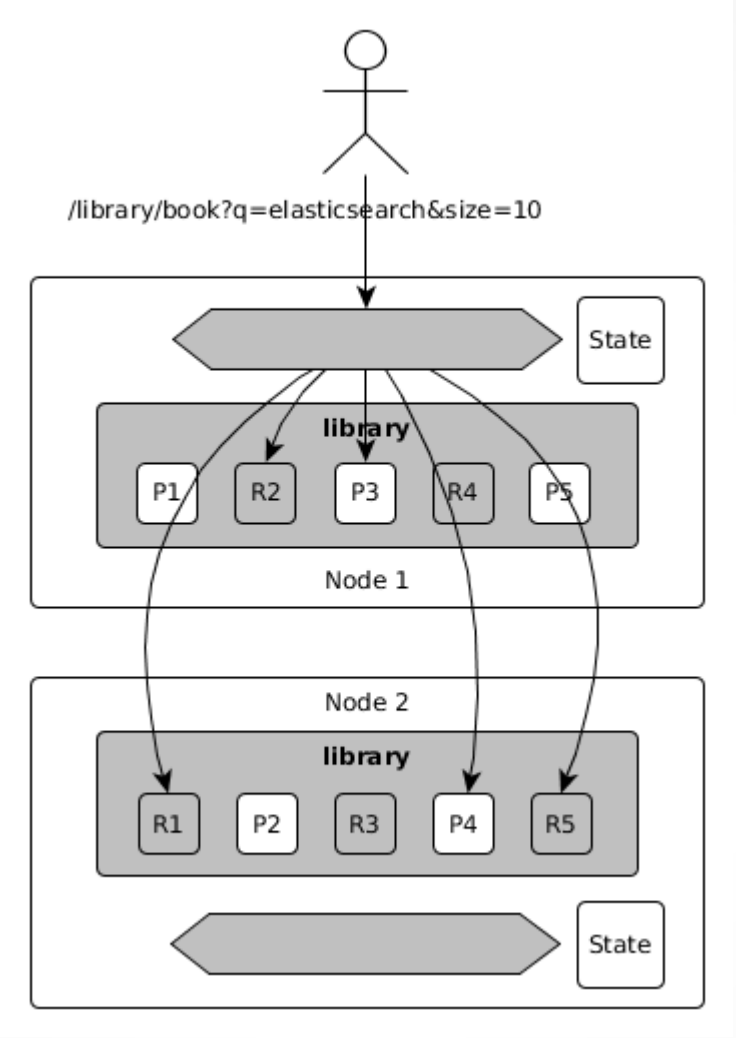
Suche im Cluster



Suche im Cluster



Suche im Cluster



Recap

- Knoten können Cluster bilden
- Datenverteilung durch Sharding
- Replicas zur Lastverteilung und Ausfallsicherheit
- Verteilte Suche
- Cluster-Zustand auf jedem Knoten

Standard-Client

Java!

```
dependencies {
    compile group: 'org.elasticsearch',
            name: 'elasticsearch',
            version: '1.6.0'
}
```

Java!

```
TransportAddress address =
    new InetSocketTransportAddress("localhost", 9300);

Client client = new TransportClient().
    addTransportAddress(address);
```

Suche per Query DSL

```
curl -XPOST "http://localhost:9200/library/book/_search"
-d'
{
    "query": {
        "filtered": {
            "query": {
                "match": {
                    "title": "elasticsearch"
                }
            },
            "filter": {
                "term": {
                    "publisher.name": "manning"
                }
            }
        }
    }
}'
```

Suche über Java-Client

```
SearchResponse searchResponse = client
        .prepareSearch("library")
        .setQuery(filteredQuery(
              matchQuery("title", "elasticsearch"),
              termFilter("publisher.name", "manning")))
        .execute().actionGet();

assertEquals(1, searchResponse.getHits().getTotalHits());

SearchHit hit = searchResponse.getHits().getAt(0);
assertEquals("Elasticsearch in Action",
              hit.getSource().get("title"));
```

Indizierung über Java-Client

```
XContentBuilder jsonBuilder = jsonBuilder()
        .startObject()
            .field("title", "Elasticsearch")
            .field("author", "Florian Hopf")
            .startObject("publisher")
                .field("name", "dpunkt")
            .endObject()
        .endObject();

client.prepareIndex("library", "book")
        .setSource(jsonBuilder)
        .execute().actionGet();
```

Indizierung über Java-Client

```
client.prepareIndex("testindex", "book").
```

```
setSource(byte[] source)                          IndexRequestBuilder
setSource(byte[] source, int offset, int …        IndexRequestBuilder
setSource(BytesReference source)                  IndexRequestBuilder
setSource(Map<String, Object> source, XCo…        IndexRequestBuilder
setSource(Object... source)                       IndexRequestBuilder
setSource(String field1, Object value1)           IndexRequestBuilder
setSource(String field1, Object value1, S…        IndexRequestBuilder
setSource(String field1, Object value1, S…        IndexRequestBuilder
setSource(String field1, Object value1, S…        IndexRequestBuilder
setSource(String source)                          IndexRequestBuilder
setSource(XContentBuilder sourceBuilder)          IndexRequestBuilder
```

Dot, semicolon and some other keys will also close this lookup and be inserted into editor

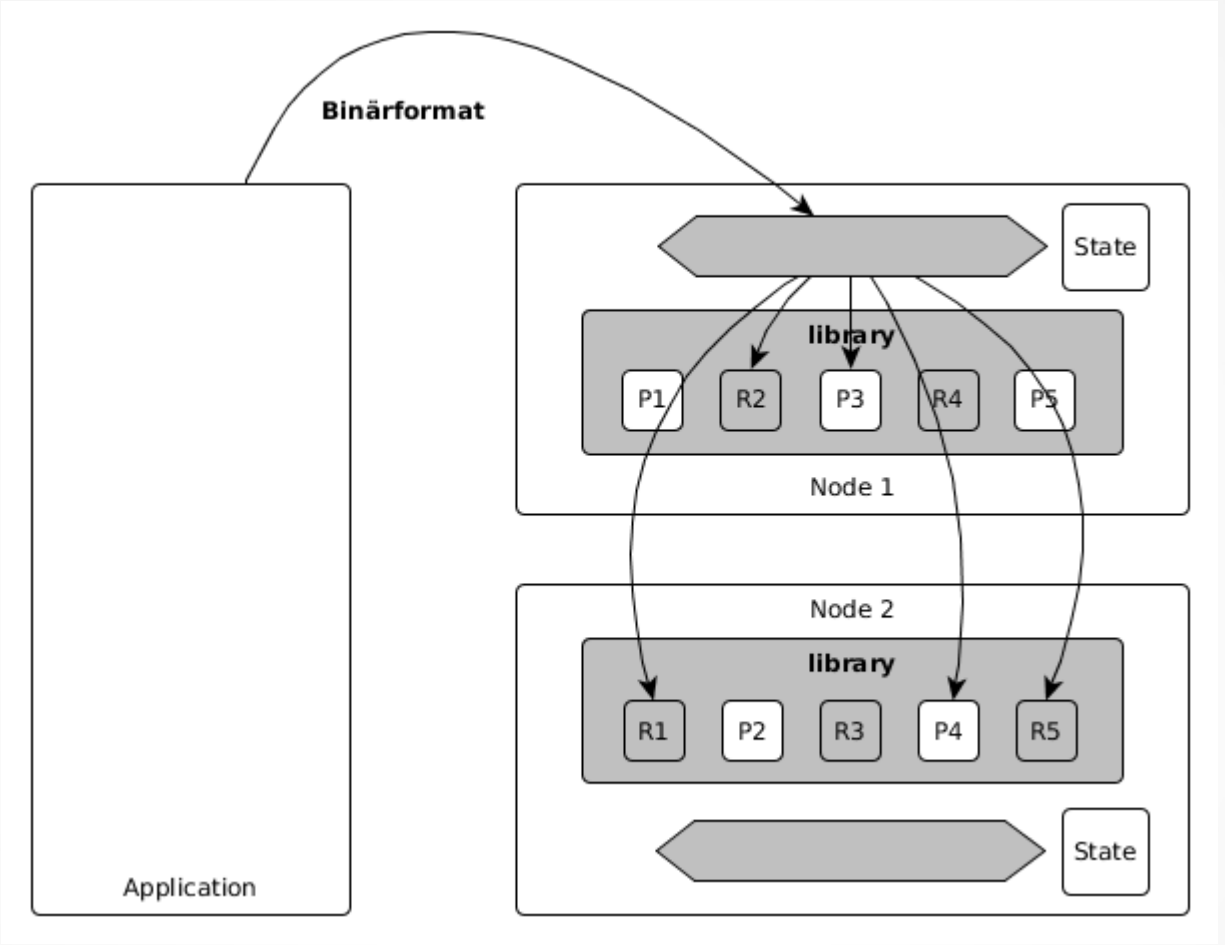
Transport-Client

- Verbindet sich mit bestehendem Cluster
- Seed-Urls zur Erzeugung

```
TransportAddress address =
    new InetSocketTransportAddress("localhost", 9300);

Client client = new TransportClient().
    addTransportAddress(address);
```

Transport-Client



Node-Client

```
Node node = nodeBuilder().client(true).node();

Client client = node.client();
```

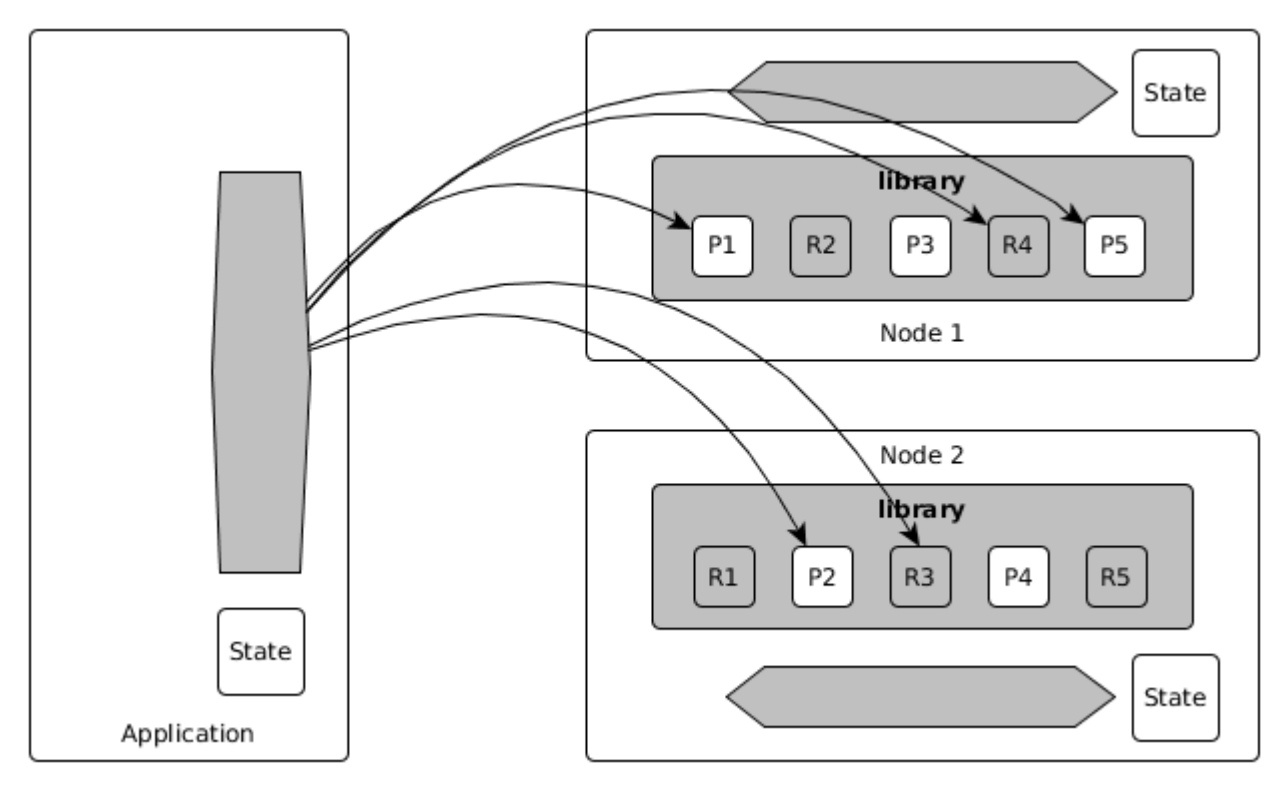
Node-Client

- Startet eigenen Knoten
- Anwendung wird Teil des Clusters

```
Node node = nodeBuilder().client(true).node();

Client client = node.client();
```

Node-Client


State
library
P1
R2
P3
R4
P5
Node 1
Node 2
library
R1
P2
R3
P4
R5
State
State
Application

Standard-Client

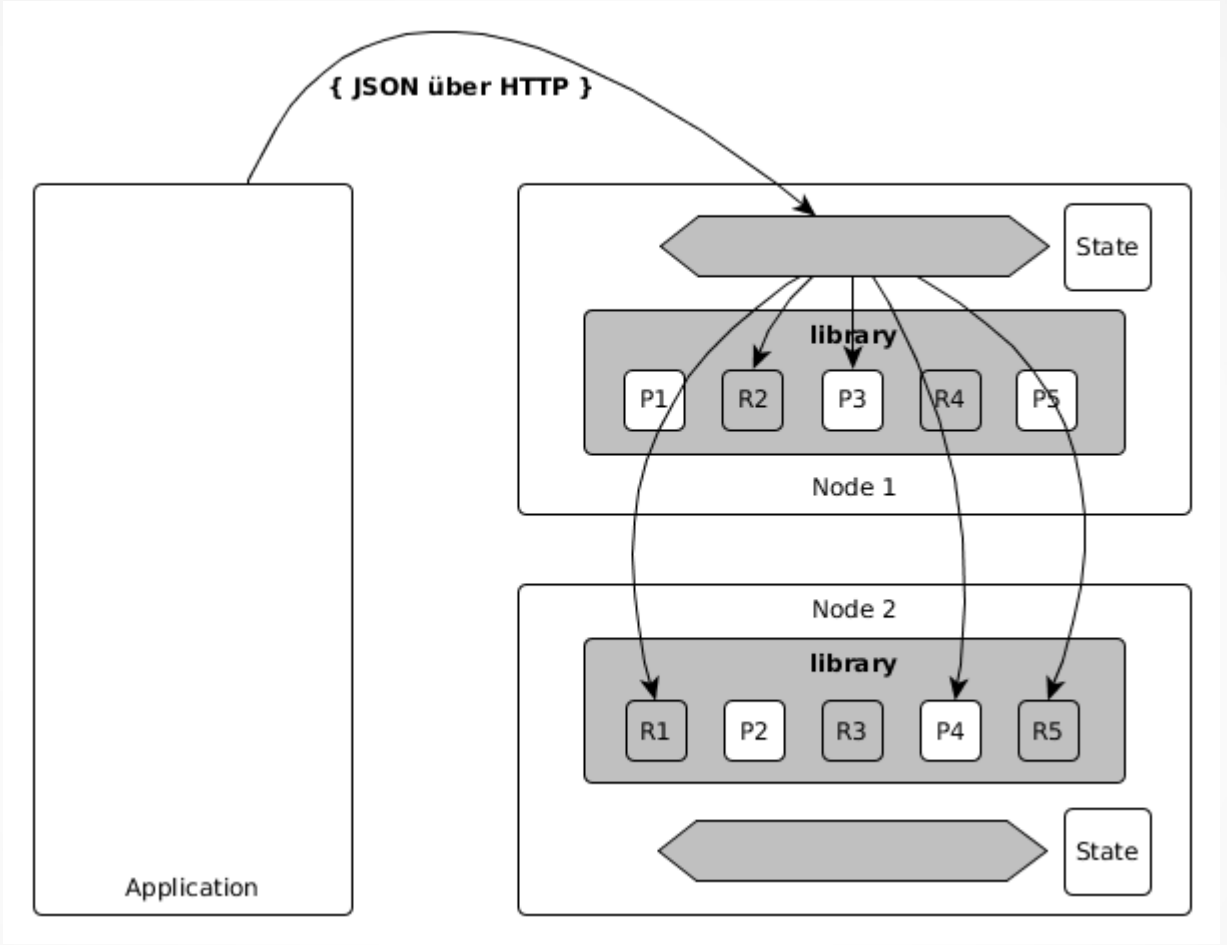
- Volle API-Unterstützung
- Effiziente Kommunikation
- Node-Client
  - Cluster-State spart unnötige Hops

Standard-Client Gotchas

- Elasticsearch-Version Client == Server
- Node-Client
  - Bei Start und Stop wird Cluster-State übertragen
- Speicherverbrauch
- Elasticsearch-Dependency

Jest – Http Client

Jest


{ JSON über HTTP }
Application
State
library
P1
R2
P3
R4
P5
Node 1
Node 2
library
R1
P2
R3
P4
R5
State

Jest

```
dependencies {
    compile group: 'io.searchbox',
            name: 'jest',
            Version: '0.1.6'
}
```

Client

```
JestClientFactory factory = new JestClientFactory();
factory.setHttpClientConfig(new HttpClientConfig
                            .Builder("http://localhost:9200")
                            .multiThreaded(true)
                            .build());

JestClient client = factory.getObject();
```

Suche mit Jest

```
String query = jsonStringThatMagicallyAppears;

Search search = new Search.Builder(query)
        .addIndex("library")
        .build();

SearchResult result = client.execute(search);
assertEquals(Integer.valueOf(1), result.getTotal());
```

Suche mit Jest

```
JsonObject jsonObject = result.getJsonObject();
JsonObject hitsObj = jsonObject.getAsJsonObject("hits");
JsonArray hits = hitsObj.getAsJsonArray("hits");
JsonObject hit = hits.get(0).getAsJsonObject();

// ... more boring code
```

Suche mit Jest

```
public class Book {

    private String title;
    private List<String> author;
    private Publisher publisher;

    public Publisher getPublisher() {
        return publisher;
    }

    public void setPublisher(Publisher publisher) {
        this.publisher = publisher;
    }

    // more code
}
```

Suche mit Jest

```
Book book = result.getFirstHit(Book.class).source;

assertEquals("Elasticsearch in Action", book.getTitle());
```

Jest

- Alternative HTTP-Implementierung
- Queries als String oder per Elasticsearch-Builder
- Indizieren und Suchen per Java-Beans

Spring Data Elasticsearch

Spring Data

- Abstraktionen für Data-Stores
- Spezialitäten bleiben erhalten
- Dynamische Repository-Implementierung
- Populäre Implementierungen
  - Spring Data JPA
  - Spring Data MongoDB

Dependency

```
dependencies {
      compile group: 'org.springframework.data',
              name: 'spring-data-elasticsearch',
              version: '1.2.0.RELEASE'
}
```

Entity

```
@Document(
        indexName = "library",
        type = "book")
public class Book {

    private String id;
    private String title;
    private List<String> author;
    private Publisher publisher;

    // more code

}
```

Repository

```
public interface BookRepository
  extends ElasticsearchCrudRepository<Book, String> {

    public Iterable<Book> findByTitle(String title);

}
```

Configuration

```
<elasticsearch:node-client id="client" />

<bean name="elasticsearchTemplate"
  class="o.s.d.elasticsearch.core.ElasticsearchTemplate">
    <constructor-arg name="client" ref="client"/>
</bean>

<elasticsearch:repositories
  base-package="de.fhopf.elasticsearch.springdata" />
```

Indexing

```
Book book = new Book();
book.setTitle("Elasticsearch");
book.setAuthor(Arrays.asList("Florian Hopf"));
book.setPublisher(new Publisher("dpunkt"));

repository.save(book);
```

Searching

```
Iterable<Book> books = repository.findAll();

books = repository.findByTitle("Elasticsearch");
```

Recap

- Abstraktion über Elasticsearch
- Entity-Beans
- Dynamische Repository-Implementierung

- Junges Projekt

Recap

- Standard-Client
  - Node- oder Transport-Client
- Jest
  - Http-Client mit Java-Bean-Unterstützung
- Spring-Data-Elasticsearch
  - Annotationzentrierte Konfiguration, dynamische Repository-Implementierungen

JVM?

- Clojure: Elastisch
  - Standard-Client oder HTTP
- Groovy: elasticsearch-groovy
  - Standard-Client, Closure-Unterstützung
- Scala: große Auswahl, z.B. elastic4s
  - Standard-Client, Type-Safety

Links

- https://www.found.no/foundation/java-clients-for-elasticsearch/
- http://elastic.co
- https://github.com/searchbox-io/Jest
- https://github.com/spring-projects/spring-data-elasticsearch
- http://www.florian-hopf.de

Weitere Infos