You know, for search

February 18th, 2011, RivieraJUG
About me

- Lukáš Vlček (@lukasvlcek)
- Java developer since 2001
- Joined Red Hat (JBoss division) in 2010
- Member of JBoss.org team, focusing on search
In the beginning there was...
Zzzzz...

...Shay Banon is not sleeping but dreaming!
Highly-available

RESTful Search Engine

Distributed

Asynchronous

Open Sourced (ASL2)

... buzzword dreaming!
... and @ElasticSearch was born!
Dreams do come true

https://github.com/elasticsearch/elasticsearch

http://www.elasticsearch.org
What is ElasticSearch?

- Distributed
- Highly-available
- RESTful search engine (on top of Lucene)
- Designed to speak JSON (JSON in, JSON out)
- and more...

... but first, let's check simple examples.
Demo #1

RESTful JSON teaser
**RESTful**

- Network interface for data indexing, searching and administration.

```
curl -XGET 'http://localhost:9200/index1,index2/typeA,typeB/_search' -d '{
   "query" : { "match_all" : {} }
}'
```

You can query one or more indices. Indices can have **aliases**, you can also use **_all** for all indices.

Each index have one or more types, *something like columns in DB table.*
Highly available

• For each index you can specify:
  • **Number of shards**
    – Each index has fixed number of shards
  • **Number of replicas**
    – Each shard can have 0-many replicas, can be changed dynamically
Distributed

- Check next slides...
ZEN Discovery

Node 1

Node 2

Node 3

Node 4

A: { shards: 3, replicas: 2 }

B: { shards: 2, replicas: 3 }

C: { shards: 1, replicas: 0 }

Gateway (longterm persistency of cluster data & metadata)
ZEN Discovery

Node 1

Node 2

Node 3

Node 4

A: { shards: 3, replicas: 2 }

B: { shards: 2, replicas: 3 }

C: { shards: 1, replicas: 4 }

Can not allocate all replicas!
Check Health API

Gateway (longterm persistency of cluster data & metadata)
Talking to the cluster

- Native client in Java and Groovy

Client

- Client type:
  - Node client
  - Transport Client
Talking to the cluster

- REST client

- Many clients built on top of REST API
  - Perl, PHP, Python, Ruby, Erlang, ... etc
Nodes do not have to be equal

- Can be a master
- Can be a data node
- Can allow for REST transport interface
  - Http, memcached, thrift
- Index store (file, memory)
- JMX enabled
- Thread pool type
- ...

Diagram:

- Node 1
- Node 2
- Node 3
Gateway

- Long time persistency allows for whole (and partial) cluster backup and recovery.

Types:
- Local (default)
- NFS
- HDFS
- AWS: S3
Distributed queries

• You can control the type of the search query per search request:
  • Query and Fetch
  • Query then Fetch
  • Dfs, Query and Fetch
  • Dfs, Query then Fetch
Demo #2

Dynamic allocation of indices, shards, replicas and Health API
Admin API

- **Indices**
  - Status
  - CRUD operation
  - Mapping, Open/Close, Update settings
  - Flush, Refresh, Snapshot, Optimize

- **Cluster**
  - Health
  - State
  - Node Info and stats
  - Shutdown
Demo #3

Admin API: getting JVM and OS stats
Rich query API

• There is rich Query DSL for search, includes:
  • Queries
    - Boolean, Fuzzy, MLT, Prefix, DisMax, ...
  • Filters
    - And/Or/Not, Boolean, Geo, Missing, Exists, ...
  • Highlighting
  • Sort
  • Facets
    - on a next slide...
Facets

- Facets allows to provide aggregated data on the search request.
  - query
  - filter
  - terms
  - range
  - (date) histogram
  - statistical
  - geo distance
Scripting support

- There is a support for using scripting languages in many places (for example for custom scoring, script fields, script key in facets ...)
  - mvel (default)
  - JS
  - Groovy
  - Python
Demo #4

Java API: Indexing data
REST API: Faceted search, Highlighting
Parent / Child

- The parent/child support allows to define a parent relationship from a child to a parent type.
  - `has_child` (query, filter)
  - `top_children` (filter)
River

- Let's listen on stream of changes and index the data...
  - CouchDB
  - RabbitMQ
  - Twitter
  - Wikipedia
Versioning (new in 0.15)

- “update if current” functionality
- ie: I can get a document, change it and then put it back in (referencing the version ID I fetched) and it will either index or fail (if the document has been modified in the interim)
- Completely real-time
Percolator (new in 0.15)

- The percolator API allows to register queries against an index, and then send a *percolate request* which includes a document, and getting back the queries that match on that document out of set of registered queries.
Q&A
Thank you!