You know, for search

11. května 2011, EurOpen - Pavlov

Outline

- Talking about ElasticSearch and giving some demos
- What you should take away from this talk?

About me

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

What is ElasticSearch ?

- Open Source (ASL2)
- Distributed (cloud friendly)
- Highly-available
- RESTful search engine (on top of Lucene)
- Designed to speak JSON (JSON in, JSON out)
- Author: Shay Banon (@kimchy)

Where ?

https://github.com/elasticsearch/elasticsearch

http://www.elasticsearch.org/

Demo #1

Searching in emails

REST API: Faceted search, Highlighting

RESTful

- Network interface for data indexing, searching and administration.

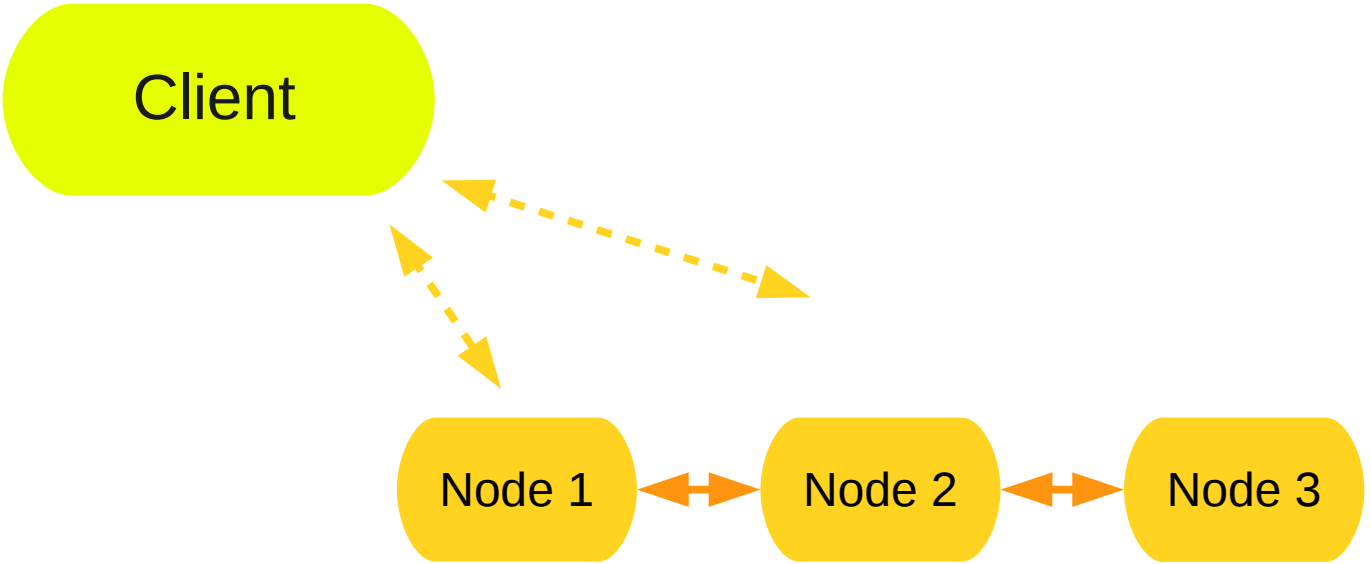
```
curl -XGET 'http://localhost:9200/index1,index2/typeA,typeB/_search' -d '{
  "query" : { "match_all" : {} }
}'
```

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.*

Talking to the cluster

- REST client



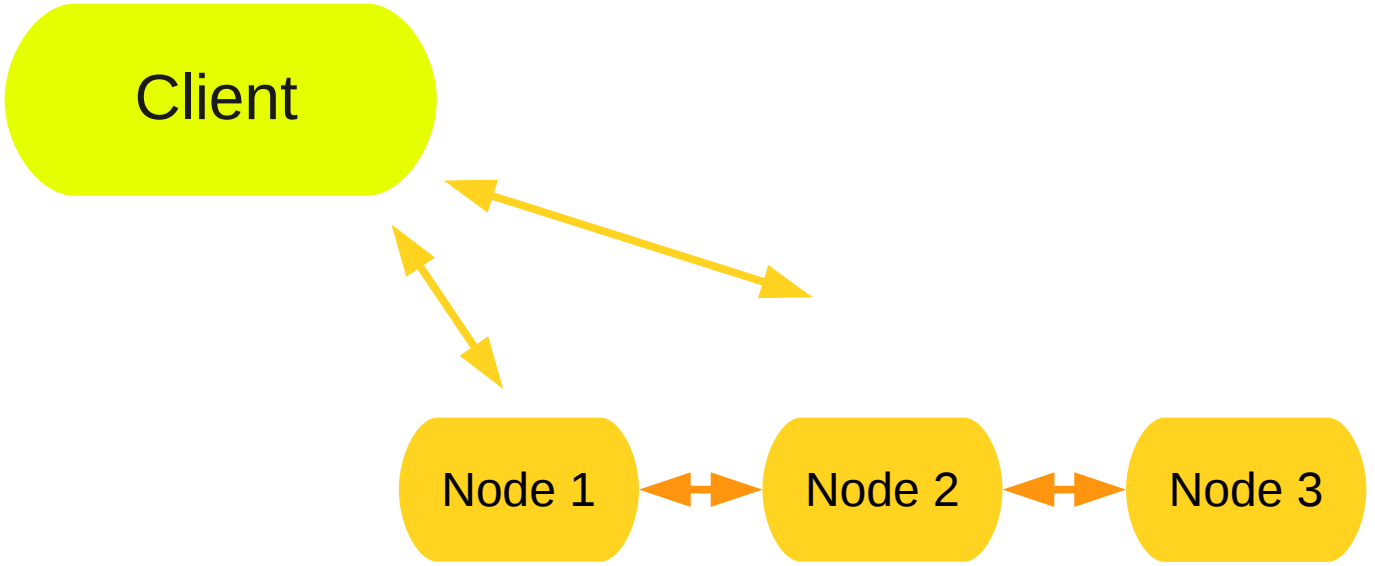
- Many clients built on top of REST API
  - Perl, PHP, Python, Ruby, Erlang, ... etc

Demo #2

RESTful JSON teaser

Talking to the cluster

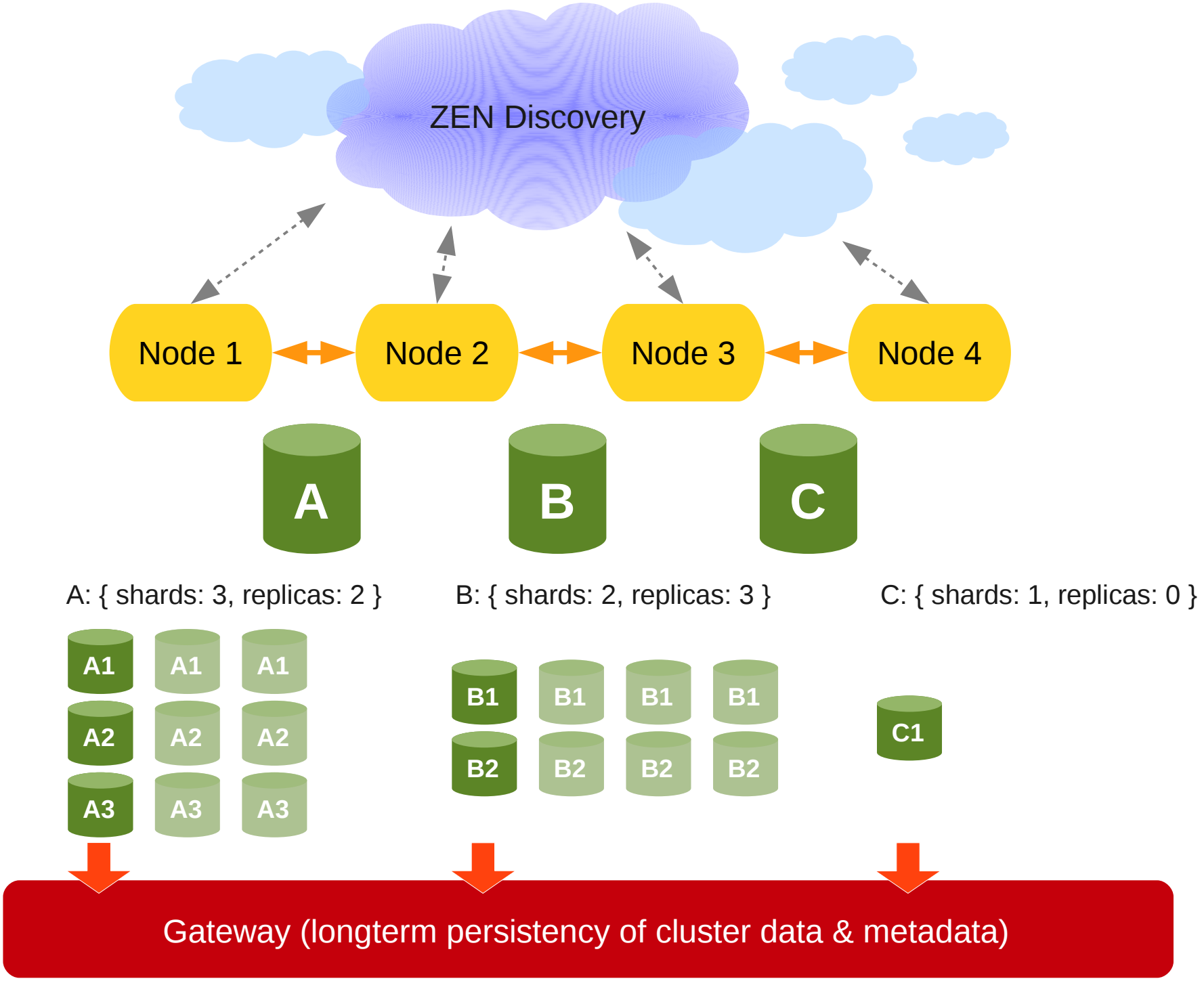
- Native client in Java and Groovy

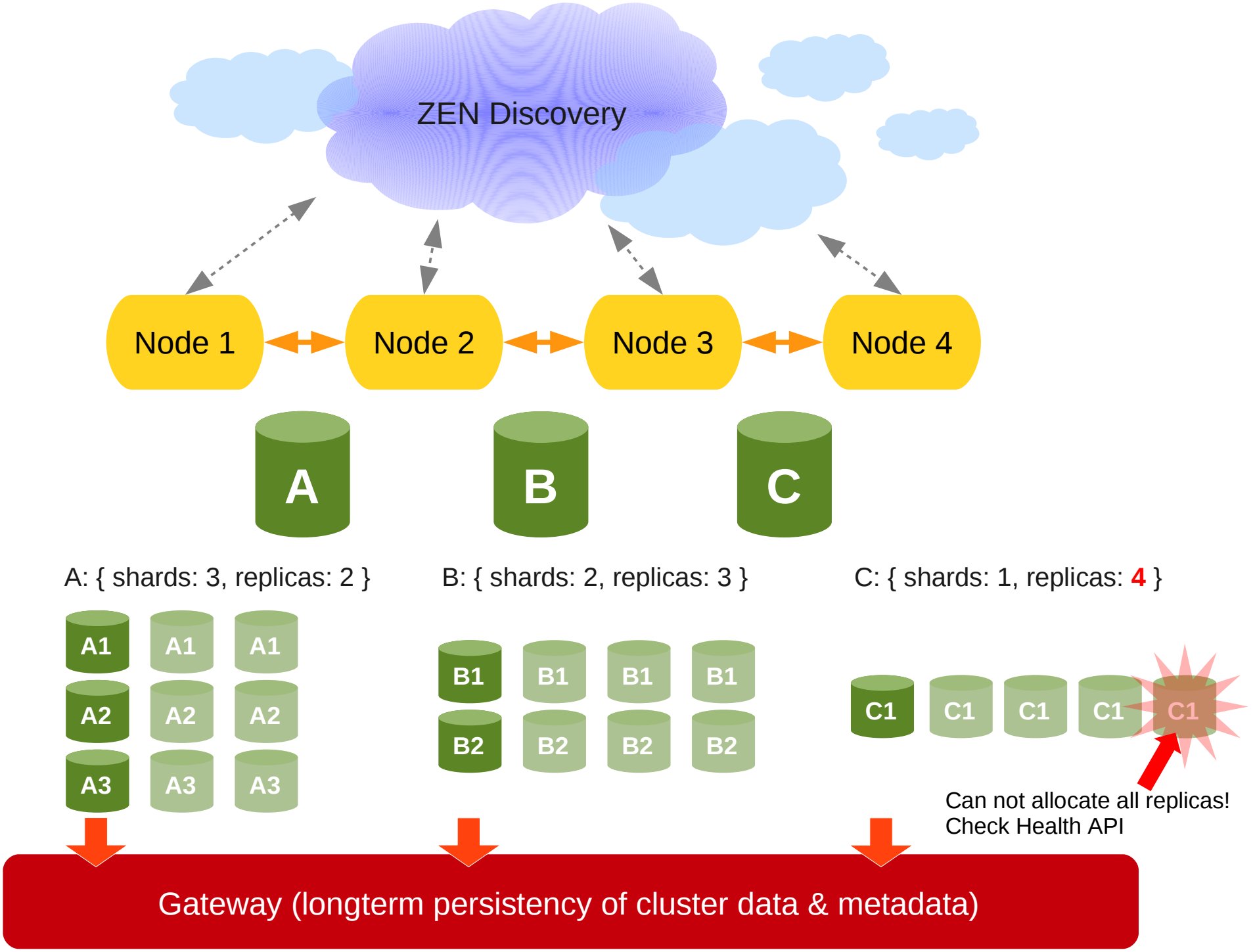


- Client type:
  - Node client
  - Transport Client

Highly available

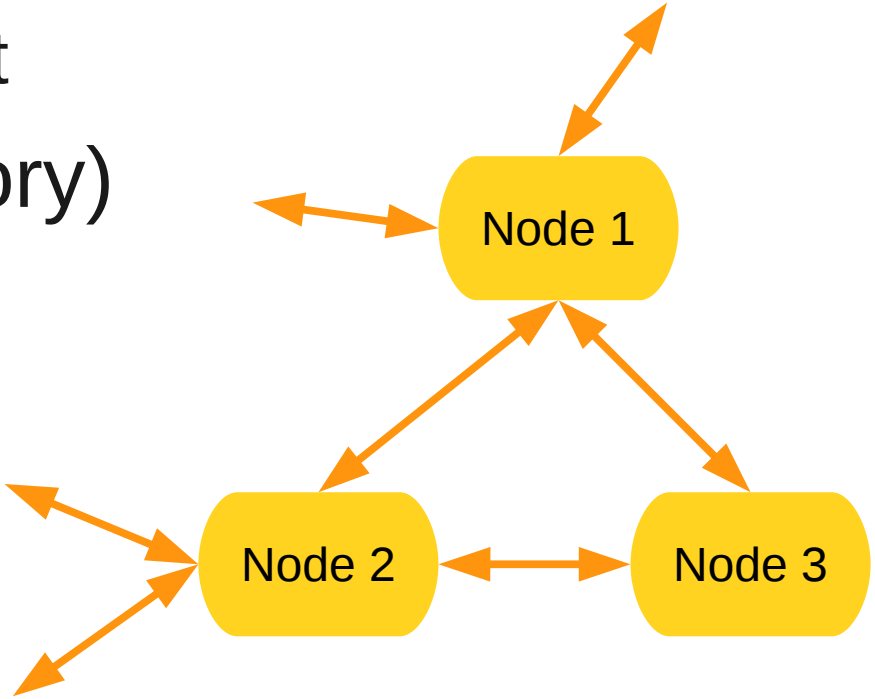
- For each index you can specify:
  - **Number of shards**
    - Each index has fixed number of shards
  - **Number of replicas**
    - Each shard can have 0-many replicas, can be changed dynamically


ZEN Discovery
Node 1
Node 2
Node 3
Node 4
A
B
C
A: { shards: 3, replicas: 2 }
B: { shards: 2, replicas: 3 }
C: { shards: 1, replicas: 0 }
A1
A1
A1
A2
A2
A2
A3
A3
A3
B1
B1
B1
B1
B2
B2
B2
B2
C1
Gateway (longterm persistency of cluster data & metadata)


ZEN Discovery
Node 1
Node 2
Node 3
Node 4
A
B
C
A: { shards: 3, replicas: 2 }
B: { shards: 2, replicas: 3 }
C: { shards: 1, replicas: 4 }
A1
A2
A3
B1
B2
C1
Can not allocate all replicas!
Check Health API
Gateway (longterm persistency of cluster data & metadata)

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)
- ...



Gateway

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

  Types:

  - Local (default)
  - NFS
  - HDFS
  - AWS: S3

Demo #3

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 #4

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 for the search request.
  - query
  - filter
  - terms
  - range
  - 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

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

- “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

- 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!