Elasticsearch
For BEGINNERS

Neil Baker
Software Engineer (Zapelin S.L.)

What is elasticsearch?

ElasticSearch is a free and open source distributed inverted index created by Shay Banon.

Build on top of Apache Lucene
- Lucene is a most popular java-based full text search index implementation.

First public release version v0.4 in February 2010.

Developed in Java, so inherently cross-plateform.

Which companies use elasticsearch?

LinkedIn

accenture
High performance. Delivered.

stackoverflow

GitHub

NETFLIX

f

Symantec.

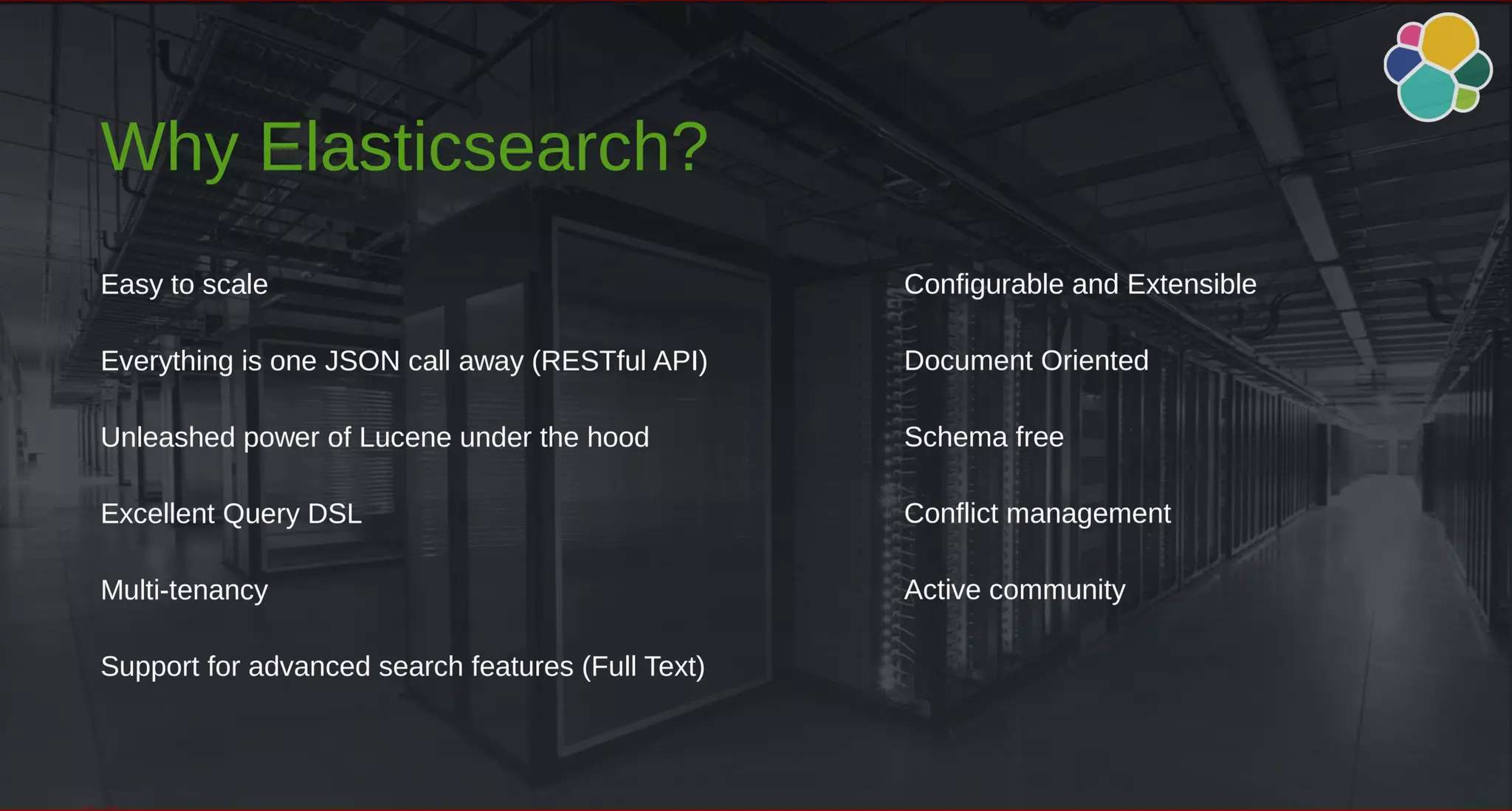
orange™

The New York Times

ebay

CISCO™

Microsoft



Why Elasticsearch?

Easy to scale

Everything is one JSON call away (RESTful API)

Unleashed power of Lucene under the hood

Excellent Query DSL

Multi-tenancy

Support for advanced search features (Full Text)

Configurable and Extensible

Document Oriented

Schema free

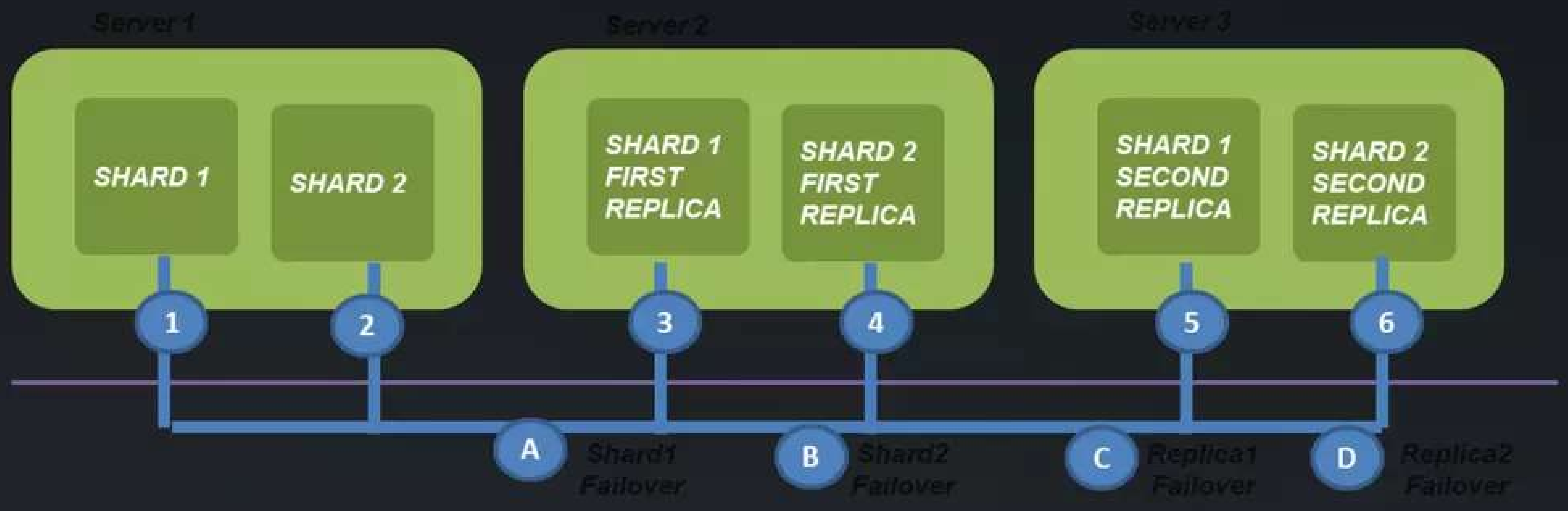
Conflict management

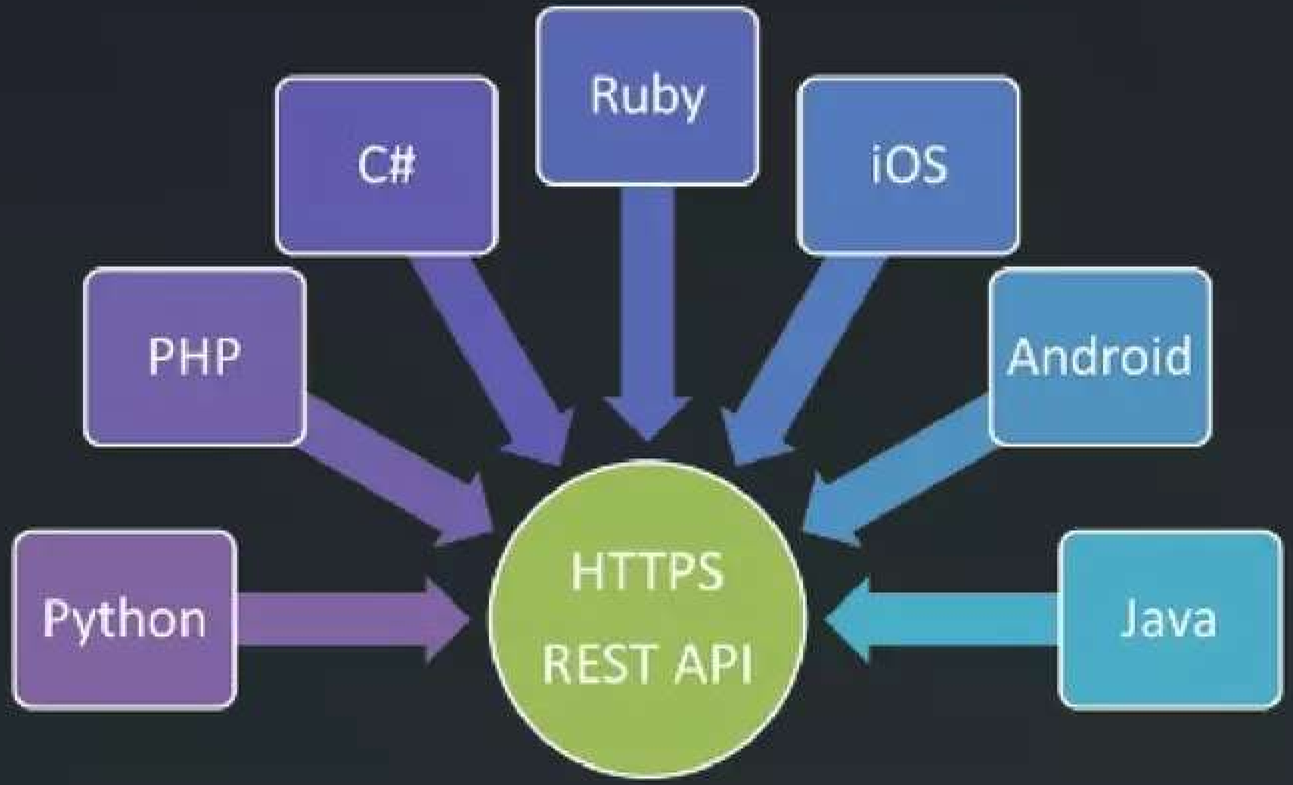
Active community

Easy to Scale

Elasticsearch allows you to start small, but will grow with your business. It is built to scale horizontally out of the box.

As you need more capacity, just add more nodes, and let the cluster reorganize itself to take advantage of the extra hardware.





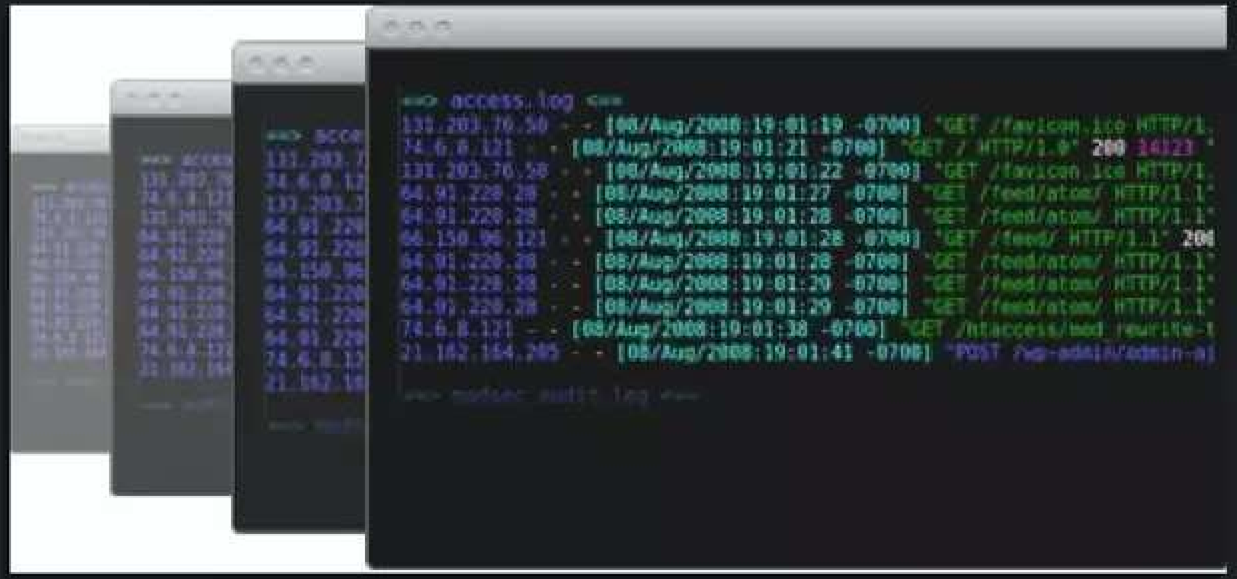
RESTful API

Elasticsearch is API driven. Almost any action can be performed using a simple RESTful API using JSON over HTTP. An API already exists in the language of your choice.

Responses are always in JSON, which is both machine and human readable.

Per-operation Persistence

Elasticsearch puts your data safety first. Document changes are recorded in transaction logs on multiple nodes in the cluster to minimize the chance of any data loss.



```
==> access.log <==
131.203.76.50 - - [08/Aug/2008:19:01:19 -0700] "GET /favicon.ico HTTP/1.
74.6.8.121 - - [08/Aug/2008:19:01:21 -0700] "GET / HTTP/1.0" 200 14123 "
131.203.76.50 - - [08/Aug/2008:19:01:22 -0700] "GET /favicon.ico HTTP/1.
64.91.228.28 - - [08/Aug/2008:19:01:27 -0700] "GET /feed/atom/ HTTP/1.1"
64.91.228.28 - - [08/Aug/2008:19:01:28 -0700] "GET /feed/atom/ HTTP/1.1"
66.150.96.121 - - [08/Aug/2008:19:01:28 -0700] "GET /feed/ HTTP/1.1" 200
64.91.228.28 - - [08/Aug/2008:19:01:28 -0700] "GET /feed/atom/ HTTP/1.1"
64.91.228.28 - - [08/Aug/2008:19:01:29 -0700] "GET /feed/atom/ HTTP/1.1"
64.91.228.28 - - [08/Aug/2008:19:01:29 -0700] "GET /feed/atom/ HTTP/1.1"
74.6.8.121 - - [08/Aug/2008:19:01:38 -0700] "GET /htaccess/mod_rewrite-1
21.182.164.205 - - [08/Aug/2008:19:01:41 -0700] "POST /wp-admin/admin-aj

==> modsec_audit.log <==
```

Excellent Query DSL

The REST API exposes a very complex and capable query DSL, that is very easy to use. Every query is just a JSON object that can practically contain any type of query, or even several of them combined.

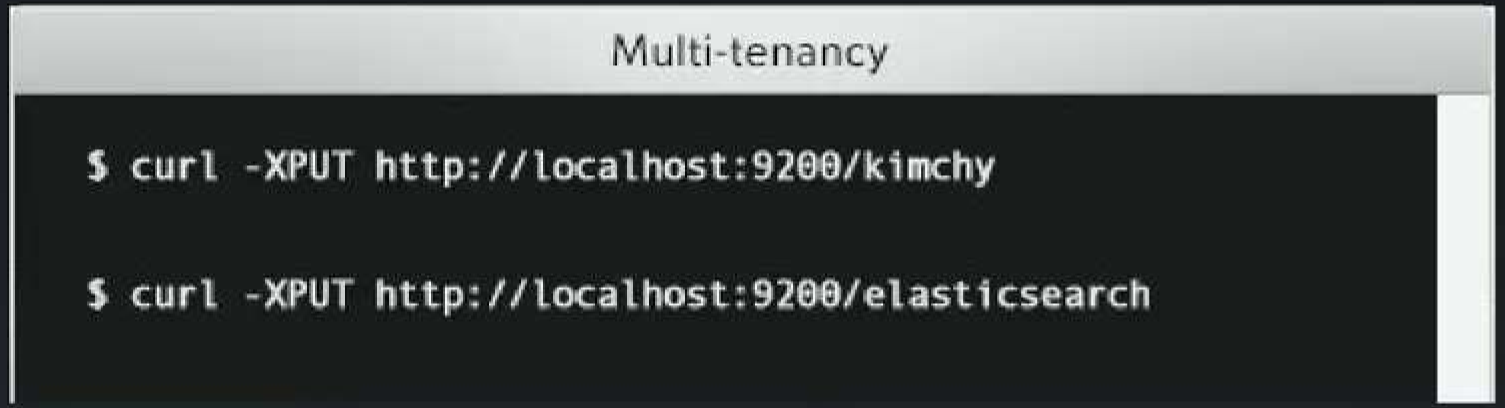
Using filtered queries, with some queries expressed as Lucene filters, helps leverage caching and thus speed up common queries, or complex queries with parts that can be reused.

Faceting, another very common search feature, is just something that upon-request is accompanied to search results, and then is ready for you to use.

Multi-tenancy

You can host multiple indexes on one Elasticsearch installation - node or cluster. Each index can have multiple "types", which are essentially completely different indexes.

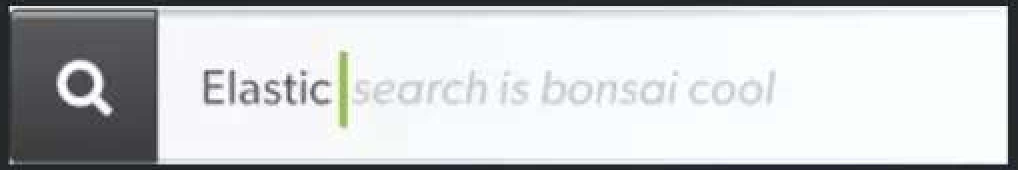
The nice thing is you can query multiple types and multiple indexes with one simple query. This opens quite a lot of options.


Multi-tenancy

```
$ curl -XPUT http://localhost:9200/kimchy

$ curl -XPUT http://localhost:9200/elasticsearch
```

Support for advanced search features (Full Text)



Elasticsearch uses Lucene under the covers to provide the most powerful full text search capabilities available in any open source product.

Search comes with multi-language support, a powerful query language, support for geolocation, context aware did-you-mean suggestions, autocomplete and search snippets.

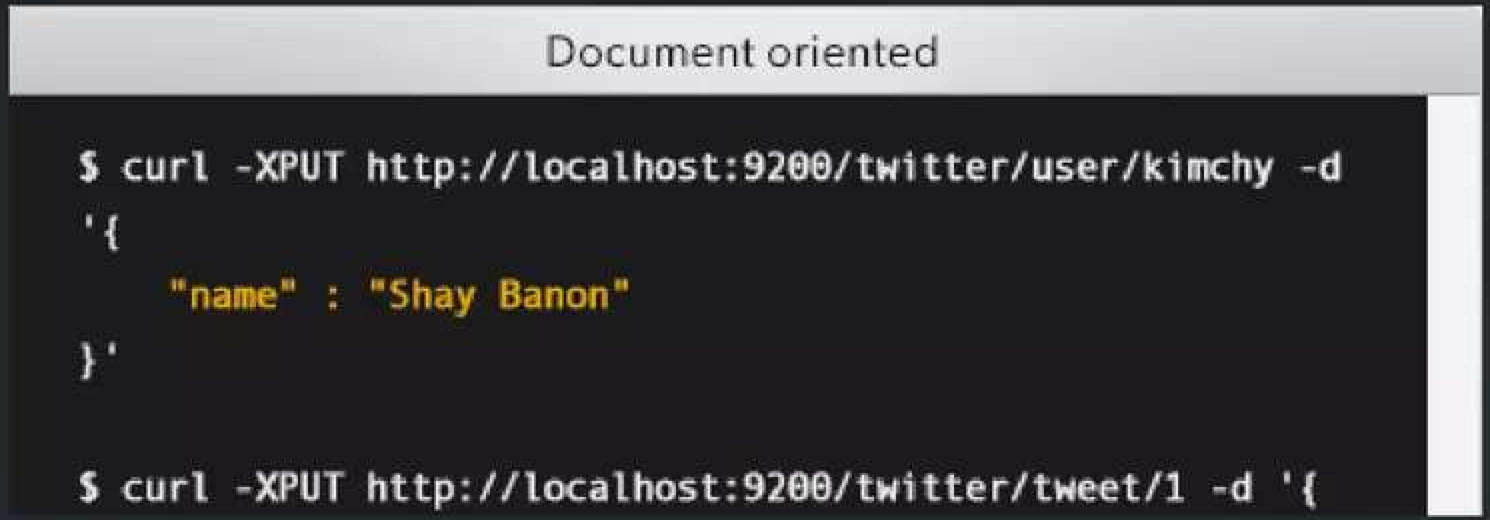
script support in filters and scorers

Configurable and Extensible

Many of Elasticsearch configurations can be changed while Elasticsearch is running, but some will require a restart (and in some cases reindexing). Most configurations can be changed using the REST API too.

Elasticsearch has several extension points - namely site plugins (let you serve static content from ES - like monitoring javascript apps), rivers (for feeding data into Elasticsearch), and plugins that let you add modules or components within Elasticsearch itself. This allows you to switch almost every part of Elasticsearch if so you choose, fairly easily.

If you need to create additional REST endpoints to your Elasticsearch cluster, that is easily done as well.


Document oriented

```
$ curl -XPUT http://localhost:9200/twitter/user/kimchy -d
'{
    "name" : "Shay Banon"
}'

$ curl -XPUT http://localhost:9200/twitter/tweet/1 -d '{
```

Document Oriented

Store complex real world entities in Elasticsearch as structured JSON documents. All fields are indexed by default, and all the indices can be used in a single query, to return results at breath taking speed.

Schema free

Elasticsearch allows you to get started easily. Toss it a JSON document and it will try to detect the data structure, index the data and make it searchable. Later, apply your domain specific knowledge of your data to customize how your data is indexed.

Conflict management

Optimistic version control can be used where needed to ensure that data is never lost due to conflicting changes from multiple processes.

Active community

The community, other than creating nice tools and plugins, is very helpful and supporting. The overall vibe is really great, and this is an important metric of any OSS project.

There are also some books currently being written by community members, and many blog posts around the net sharing experiences and knowledge

BASIC CONCEPTS

Cluster :

A cluster consists of one or more nodes which share the same cluster name. Each cluster has a single master node which is chosen automatically by the cluster and which can be replaced if the current master node fails.

Node :

A node is a running instance of elasticsearch which belongs to a cluster. Multiple nodes can be started on a single server for testing purposes, but usually you should have one node per server.

At startup, a node will use unicast (or multicast, if specified) to discover an existing cluster with the same cluster name and will try to join that cluster.

Index :

An index is like a 'database' in a relational database. It has a mapping which defines multiple types.
An index is a logical namespace which maps to one or more primary shards and can have zero or more replica shards.

Type :

A type is like a 'table' in a relational database. Each type has a list of fields that can be specified for documents of that type. The mapping defines how each field in the document is analyzed.

Document :

A document is a JSON document which is stored in elasticsearch. It is like a row in a table in a relational database. Each document is stored in an index and has a type and an id.

A document is a JSON object (also known in other languages as a hash / hashmap / associative array) which contains zero or more fields, or key-value pairs. The original JSON document that is indexed will be stored in the _source field, which is returned by default when getting or searching for a document.

Field :

A document contains a list of fields, or key-value pairs. The value can be a simple (scalar) value (eg a string, integer, date), or a nested structure like an array or an object. A field is similar to a column in a table in a relational database.

The mapping for each field has a field 'type' (not to be confused with document type) which indicates the type of data that can be stored in that field, eg integer, string, object. The mapping also allows you to define (amongst other things) how the value for a field should be analyzed.

Mapping :

A mapping is like a 'schema definition' in a relational database. Each index has a mapping, which defines each type within the index, plus a number of index-wide settings. A mapping can either be defined explicitly, or it will be generated automatically when a document is indexed

Shard :

A shard is a single Lucene instance. It is a low-level "worker" unit which is managed automatically by elasticsearch. An index is a logical namespace which points to primary and replica shards.

Elasticsearch distributes shards amongst all nodes in the cluster, and can move shards automatically from one node to another in the case of node failure, or the addition of new nodes.

Primary Shard :

Each document is stored in a single primary shard. When you index a document, it is indexed first on the primary shard, then on all replicas of the primary shard. By default, an index has 5 primary shards. You can specify fewer or more primary shards to scale the number of documents that your index can handle.

Replica Shard :

Each primary shard can have zero or more replicas. A replica is a copy of the primary shard, and has two purposes:

1) increase failover: a replica shard can be promoted to a primary shard if the primary fails.
2) increase performance: get and search requests can be handled by primary or replica shards.

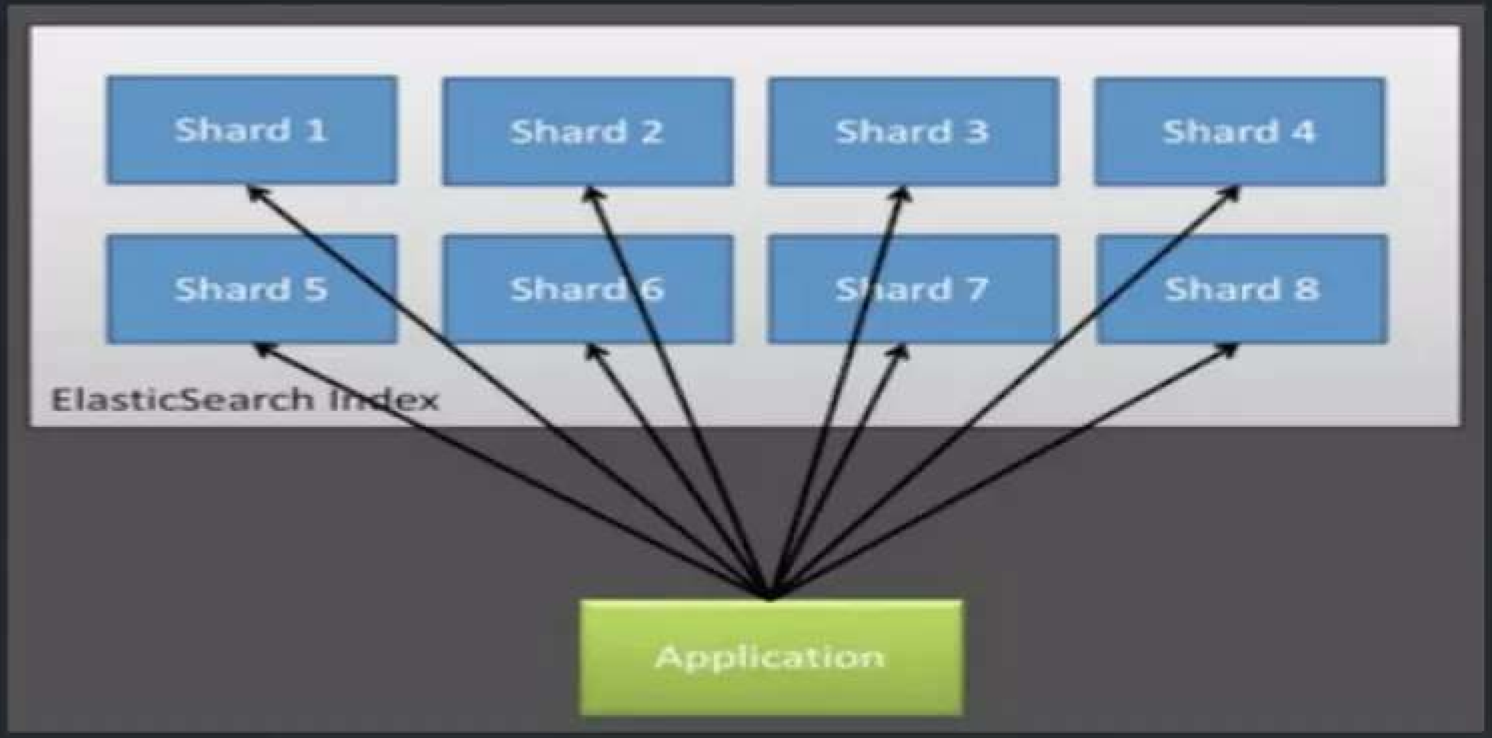
ElasticSearch Routing

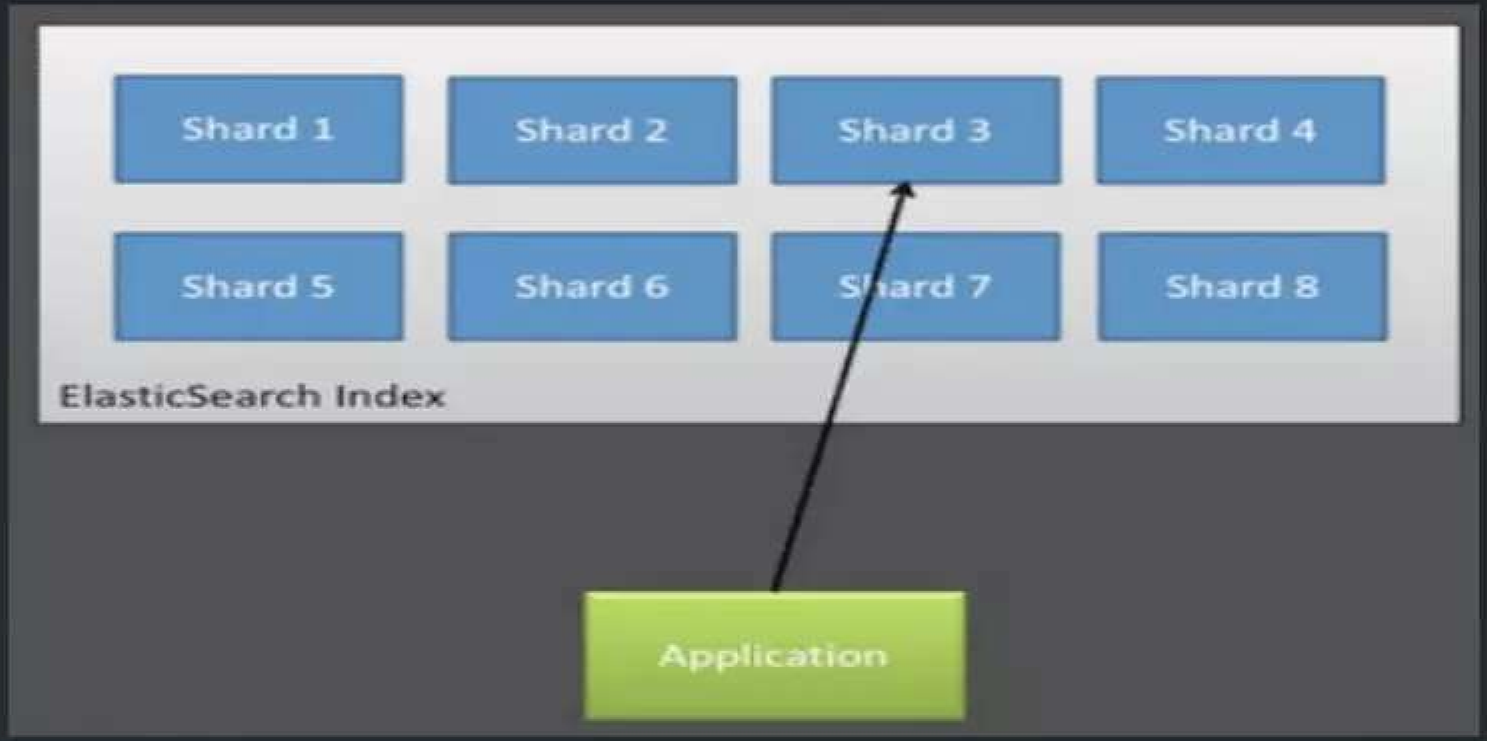
All of your data lives in a primary shard, somewhere in the cluster. You may have five shards or five hundred, but any particular document is only located in one of them. Routing is the process of determining *which* shard that document will reside in.

Elasticsearch has no idea where to look for your document. All the docs were randomly distributed around your cluster. so Elasticsearch has no choice but to broadcasts the request to all shards. This is a non-negligible overhead and can easily impact performance.

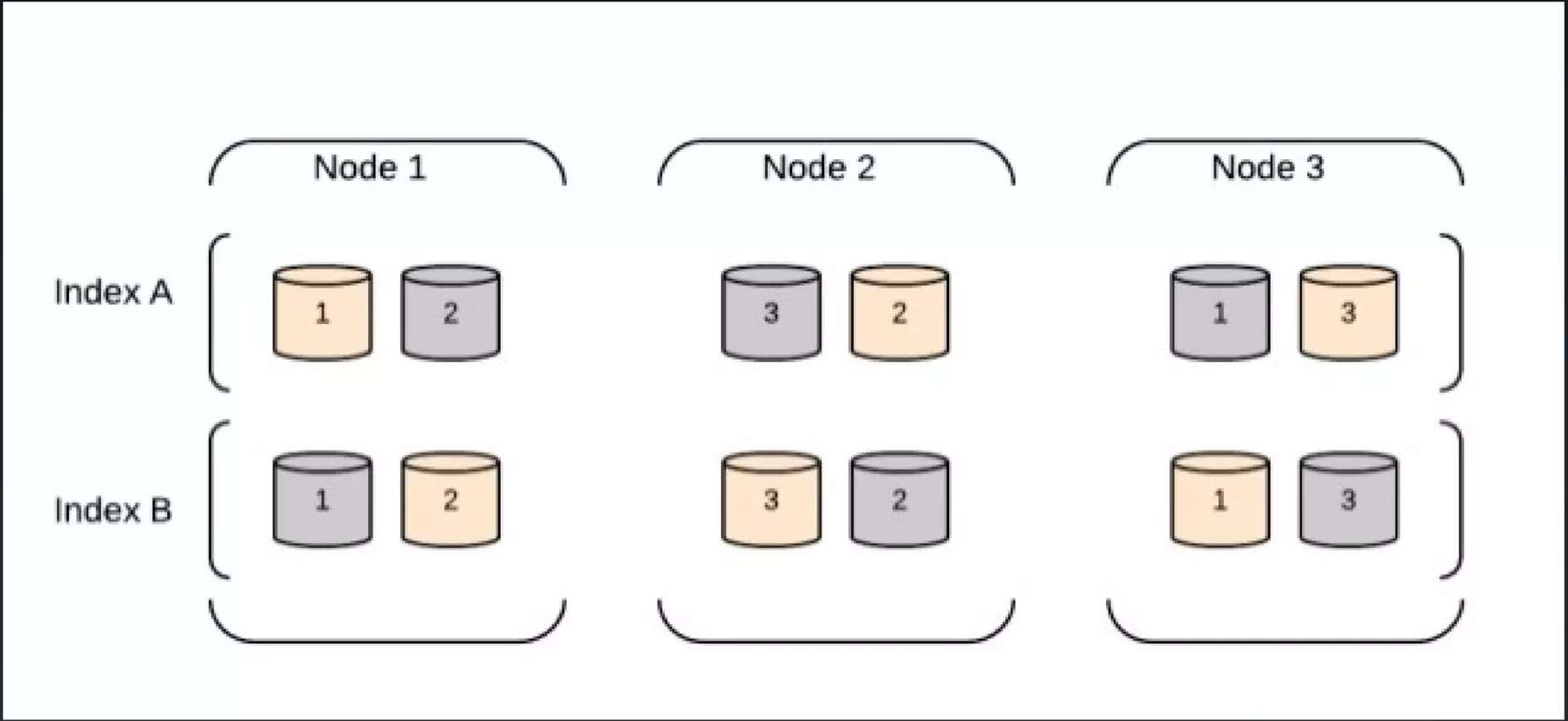
Wouldn't it be nice if we could tell Elasticsearch which shard the document lived in? Then you would only have to search one shard to find the document(s) that you need.

Routing ensures that all documents with the same routing value will locate to the same shard, eliminating the need to broadcast searches.

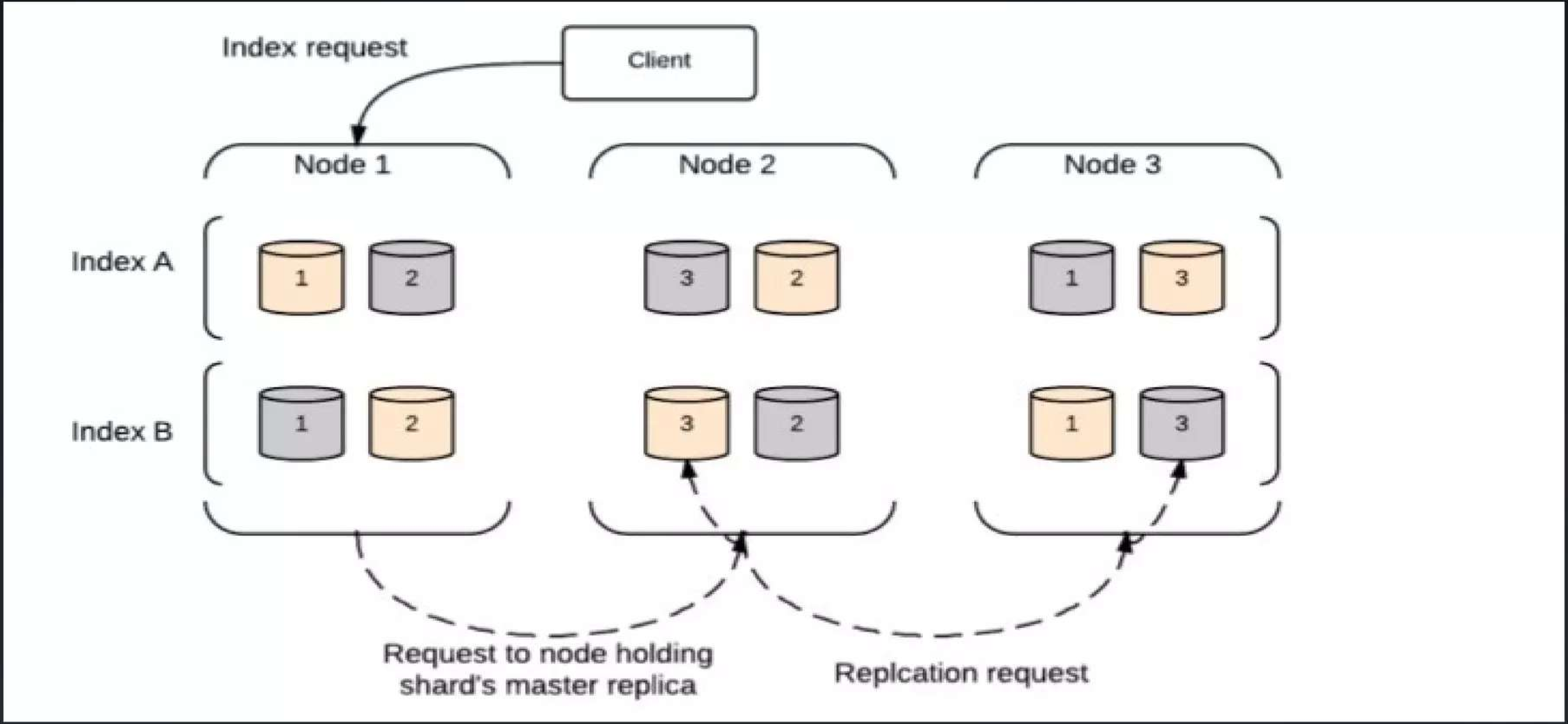




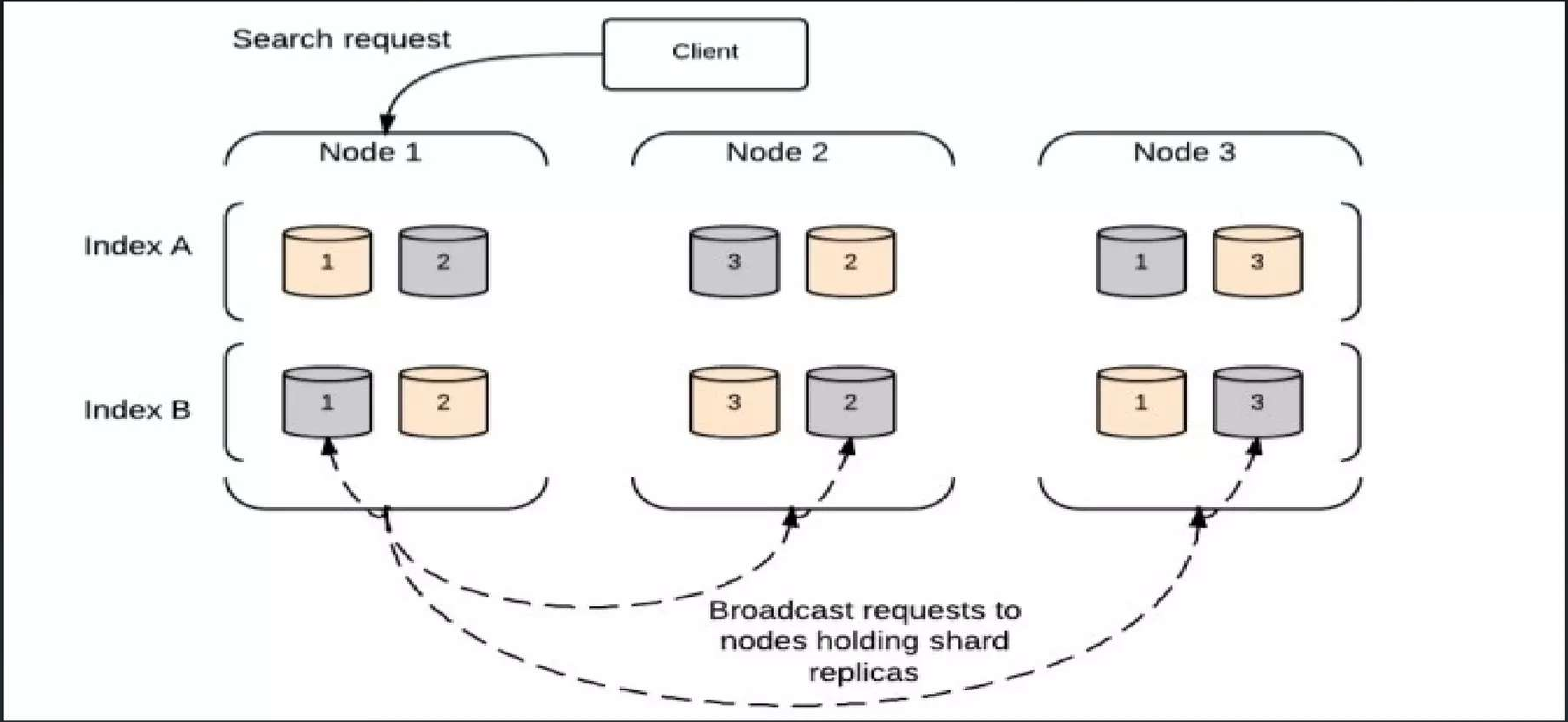
Cluster Architecture



Index Request



Search Request



ELASTIC VS. MySQL

ElasticSearch	MySQL
Indices	Database
Types	Tables
Documents	Rows
Keys	Columns

Performance

Core i7, a 2Ghz, 8GB RAM, 128GB SSD)

Insert of 10 Mio. Datasets:

Elasticsearch:	23 Minutes
MySQL without index:	56 Minutes
MySQL with Index:	228 Minutes

Select of 100 full Entrys:

Elasticsearch:	5 ms
MySQL:	9 ms

Select of the next 100 full Entrys:

Elasticsearch:	4 ms
MySQL:	18 ms

Select name and firstname of 100 Entrys:

Elasticsearch:	5 ms
MySQL:	9 ms

INSTALL ELASTIC

```
wget https://artifacts.elastic.co/downloads/elasticsearch/elasticsearch-5.1.1.tar.gz

tar -xzf elasticsearch-5.1.1.tar.gz

cd elasticsearch-5.1.1/


./bin/elasticsearch
```

Is it running?

GET http://localhost:9200/?pretty

Response :

```
{
    "name": "Vivisector",
    "cluster_name": "elasticsearch",
    "version": {
        "number": "2.3.3",
        "build_hash": "218bdf10790eef486ff2c41a3df5cfa32dadcfde",
        "build_timestamp": "2016-05-17T15:40:04Z",
        "build_snapshot": false,
        "lucene_version": "5.5.0"
    },
    "tagline": "You Know, for Search"
}
```

Let´s PLAY WITH ELASTICSEARCH

Indexing a document

Request :

```
$ curl -XPUT "http://localhost:9200/test-data/cities/21" -d '{
    "rank": 21,
    "city": "Boston",
    "state": "Massachusetts",
    "population2010": 617594,
    "land_area": 48.277,
    "location": {
    "lat": 42.332,
    "lon": 71.0202 },
    "abbreviation": "MA"
}'
```

Response : `{"ok":true,"_index":"test-data","_type":"cities","_id":"21","_version":1}`

Getting a document

Request :

```
    $ curl -XGET "http://localhost:9200/test-data/cities/21?pretty"
```

Response :

```
{
     "_index" : "test-data",
     "_type" : "cities",
     "_id" : "21",
     "_version" : 1,
     "exists" : true, "_source" : {
        "rank": 21,
        "city": "Boston",
        "state": "Massachusetts",
        "population2010": 617594,
        "land_area": 48.277,
        "location": {
        "lat": 42.332,
        "lon": 71.0202  },
        "abbreviation": "MA"
    }
}
```

Updating a document

Request :

```
$ curl -XPUT "http://localhost:9200/test-data/cities/21" -d '{
    "rank": 21,
    "city": "Boston",
    "state": "Massachusetts",
    "population2010": 617594,
    "population2012": 636479,
    "land_area": 48.277,
    "location": {
    "lat": 42.332,
    "lon": 71.0202 },
    "abbreviation": "MA"
}'
```

Response : `{"ok":true,"_index":"test-data","_type":"cities","_id":"21","_version":2}`

Searching

Searching and querying takes the format of: http://localhost:9200/[index]/[type]/[operation]

Search across all indexes and all types
    http://localhost:9200/_search

Search across all types in the test-data index.
    http://localhost:9200/test-data/_search

Search explicitly for documents of type cities within the test-data index.
    http://localhost:9200/test-data/cities/_search

Search explicitly for documents of type cities within the test-data index using paging.
    http://localhost:9200/test-data/cities/_search?size=5&from=10

There's 3 different types of search queries

- Full Text Search (query string)
- Structured Search (filter)
- Analytics (facets)

Full Text Search (query string)

In this case you will be searching in bits of natural language for (partially) matching query strings. The Query DSL alternative for searching for "Boston" in all documents, would look like:

Request :

```
     $ curl -XGET "http://localhost:9200/test-data/cities/_search?pretty=true" -d '{
             "query": { "query_string": {  "query": "boston" }}}'
```

Response : {

```
     "took" : 5,
     "timed_out" : false,
     "_shards" : {
             "total" : 1,
             "successful" : 1,
             "failed" : 0 },
     "hits" : {
              "total" : 1,
              "max_score" : 6.1357985,
              "hits" : [ {
                  "_index" : "test-data",
                  "_type" : "cities",
                  "_id" : "21",
                  "_score" : 6.1357985, "_source" : {"rank":"21","city":"Boston",...}
                   } ]
              }
      }...
```

Structured Search (filter)

Structured search is about interrogating data that has inherent structure. Dates, times and numbers are all structured—they have a precise format that you can perform logical operations on. Common operations include comparing ranges of numbers or dates, or determining which of two values is larger.

With structured search, the answer to your question is always a yes or no; something either belongs in the set or it does not. Structured search does not worry about document relevance or scoring—it simply includes or excludes documents.

Request :

```
$ curl -XGET "http://localhost:9200/test-data/cities/_search?pretty=true" -d '{
        “query": { “filtered":  { “filter”: {  “term":  { “city” : “boston“ }}}}}’

$ curl -XGET "http://localhost:9200/test-data/cities/_search?pretty" -d '{
     "query": {
        "range": {
        "population2012": {
             "from": 500000,
             "to": 1000000
        }}}}‘

$ curl -XGET "http://localhost:9200/test-data/cities/_search?pretty" -d '{
"query": { "bool": { "should": [{  "match": { "state": "Texas"}  }, {"match": { "state":
"California"} }],
"must": {  "range": {  "population2012": {  "from": 500000,  "to": 1000000  }  }  },
"minimum_should_match": 1}}}'
```

Analytics (facets)

Requests of this type will not return a list of matching documents, but a statistical breakdown of the documents.

Elasticsearch has functionality called aggregations, which allows you to generate sophisticated analytics over your data. It is similar to GROUP BY in SQL.

Request :

```
$ curl -XGET "http://localhost:9200/test-data/cities/_search?pretty=true" -d '{
           “aggs": { “all_states":  { “terms“:  { “field” : “state“ }}}}’
```

Response :

```
        { ...
       "hits": { ... },
       "aggregations": {
                  "all_states": {
                        "buckets": [
                            {"key":  "massachusetts ", "doc_count": 2},
                            {"key":  "danbury", "doc_count": 1}
                        ]
        }}}
```

ElasticSearch Monitoring

ElasticSearch-Head - https://github.com/mobz/elasticsearch-head

Marvel - http://www.elasticsearch.org/guide/en/marvel/current/#_marvel_8217_s_dashboards

Paramedic - https://github.com/karmi/elasticsearch-paramedic

Bigdesk - https://github.com/lukas-vlcek/bigdesk/

ElasticSearch Limitations

Security : ElasticSearch does not provide any build-in authentication or access control functionality.

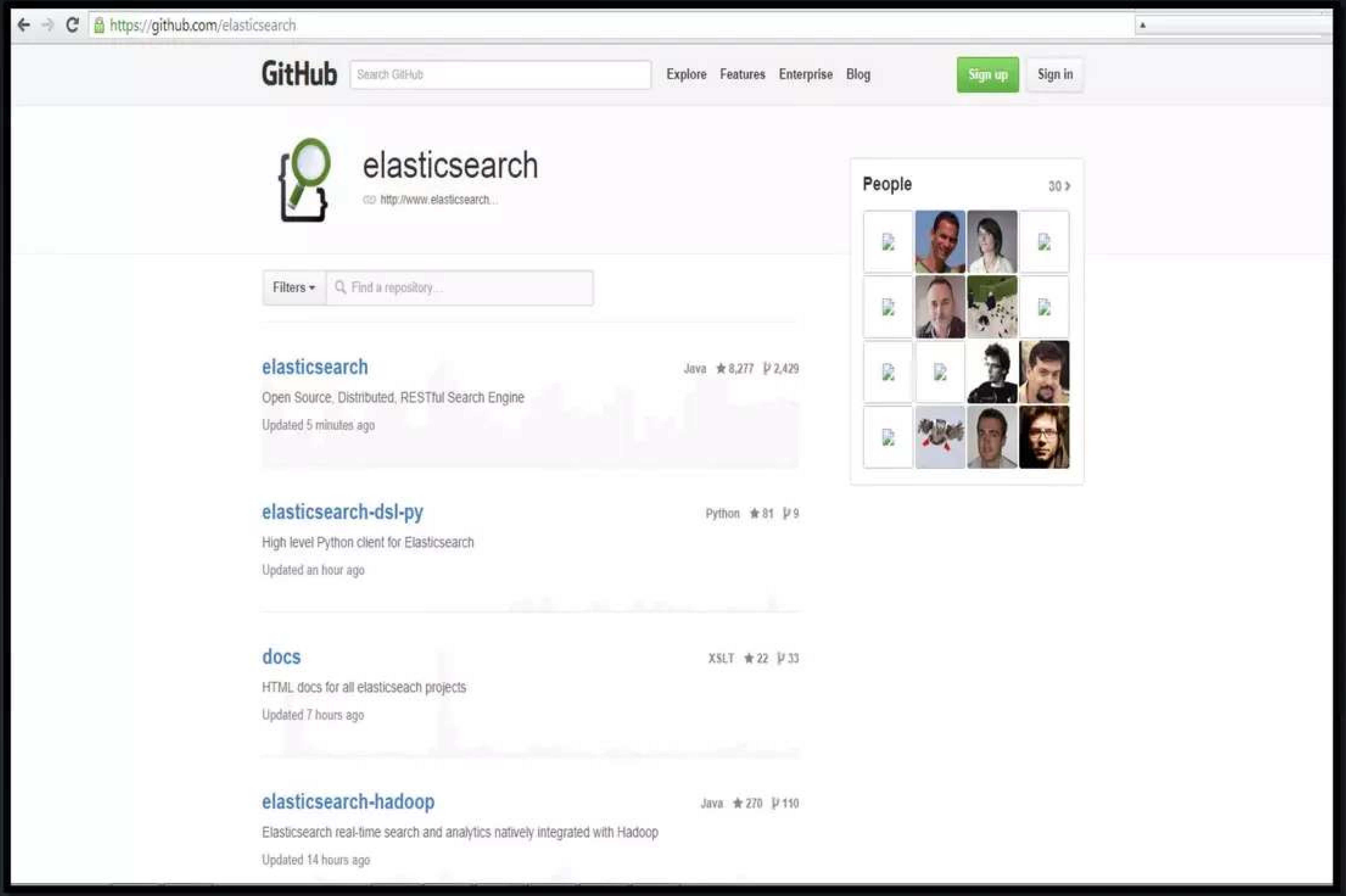
Transactions : There is no much more support for transactions or processing on data manipulation.

Durability : ES is distributed and fairly stable but backups and durability are not as high priority as in other data stores

Large Computations: Commands for searching data are not suited to "large" scans of data and advanced computation on the db side.

Data Availability : ES makes data available in "near real-time" which may require additional considerations in your application (ie: comments page where a user adds new comment, refreshing the page might not actually show the new post because the index is still updating).

Open-Source Libraries



https://github.com/elasticsearch

GitHub Explore Features Enterprise Blog Sign up Sign in

elasticsearch
http://www.elasticsearch...

elasticsearch — Java ★ 8,277 2,429
Open Source, Distributed, RESTful Search Engine
Updated 5 minutes ago

elasticsearch-dsl-py — Python ★ 81 9
High level Python client for Elasticsearch
Updated an hour ago

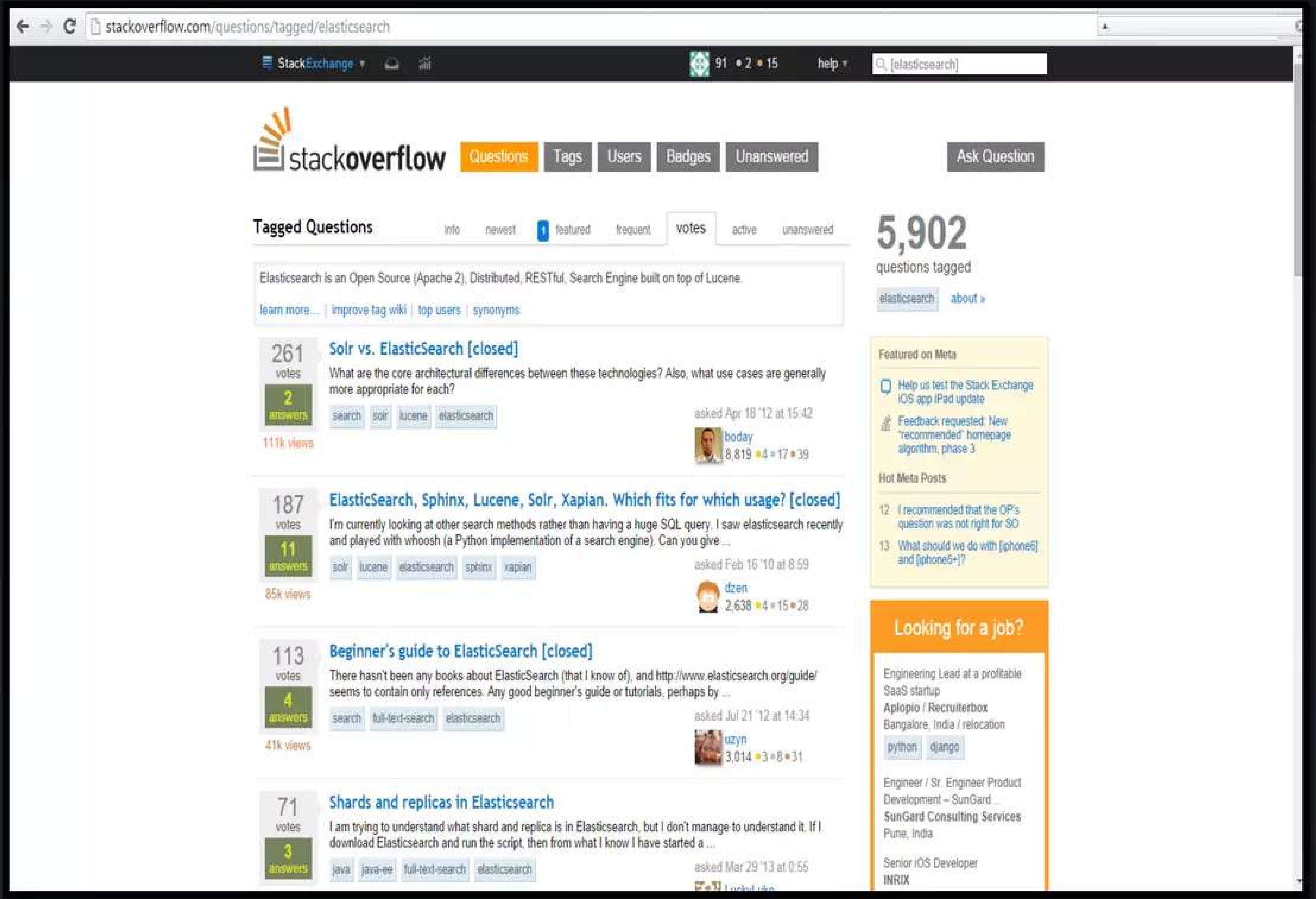
docs — XSLT ★ 22 33
HTML docs for all elasticseach projects
Updated 7 hours ago

elasticsearch-hadoop — Java ★ 270 110
Elasticsearch real-time search and analytics natively integrated with Hadoop
Updated 14 hours ago

People 30

https://github.com/elasticsearch

stackoverflow



stackoverflow.com/questions/tagged/elasticsearch

StackExchange 91 • 2 • 15 help [elasticsearch]

stackoverflow Questions Tags Users Badges Unanswered Ask Question

Tagged Questions info newest 1 featured frequent votes active unanswered

Elasticsearch is an Open Source (Apache 2), Distributed, RESTful, Search Engine built on top of Lucene.

learn more... | improve tag wiki | top users | synonyms

261 votes, 2 answers, 111k views
Solr vs. ElasticSearch [closed]
What are the core architectural differences between these technologies? Also, what use cases are generally more appropriate for each?
search solr lucene elasticsearch
asked Apr 18 '12 at 15:42 boday 8,819 • 4 • 17 • 39

187 votes, 11 answers, 85k views
ElasticSearch, Sphinx, Lucene, Solr, Xapian. Which fits for which usage? [closed]
I'm currently looking at other search methods rather than having a huge SQL query. I saw elasticsearch recently and played with whoosh (a Python implementation of a search engine). Can you give ...
solr lucene elasticsearch sphinx xapian
asked Feb 16 '10 at 8:59 dzen 2,638 • 4 • 15 • 28

113 votes, 4 answers, 41k views
Beginner's guide to ElasticSearch [closed]
There hasn't been any books about ElasticSearch (that I know of), and http://www.elasticsearch.org/guide/ seems to contain only references. Any good beginner's guide or tutorials, perhaps by ...
search full-text-search elasticsearch
asked Jul 21 '12 at 14:34 uzyn 3,014 • 3 • 8 • 31

71 votes, 3 answers
Shards and replicas in Elasticsearch
I am trying to understand what shard and replica is in Elasticsearch, but I don't manage to understand it. If I download Elasticsearch and run the script, then from what I know I have started a ...
java java-ee full-text-search elasticsearch
asked Mar 29 '13 at 0:55

5,902 questions tagged elasticsearch about »

Featured on Meta
- Help us test the Stack Exchange iOS app iPad update
- Feedback requested: New "recommended" homepage algorithm, phase 3

Hot Meta Posts
- 12 I recommended that the OP's question was not right for SO
- 13 What should we do with [iphone6] and [iphone6+]?

Looking for a job?
- Engineering Lead at a profitable SaaS startup
  Aplopio / Recruiterbox
  Bangalore, India / relocation
  python django
- Engineer / Sr. Engineer Product Development – SunGard...
  SunGard Consulting Services
  Pune, India
- Senior iOS Developer
  INRIX

http://stackoverflow.com/questions/tagged/elasticsearch

Get started.

www.elasticsearch.org

About me

Founder and Technical Director of Terions Communication LTD (London/Berlin, 1996-2006)
- Datacenter Operator for Press and Image Agencys and Part of the DPA (German Press Agency)

Software Developer Zapelin S.L (Adeje)
- Development of IPTV Solutions for Hotels e.g.

RHCE - Red Hat Certified Engineer

CCDP - Cisco Certified Design Professional

DBA - Oracle Certified Professional, MySQL 5 Database Administrator

Contact for Questions: neil@cconnect.es

THANK YOU

GRACIAS
ARIGATO
SHUKURIA
DANKSCHEEN
JUSPAXAR
BÏYAN
SHUKRIA
TINGKI
YAQHANYELAY
CHALTU
NUHUN
SPASSIBO
SNACHALHUYA
TASHAKKUR ATU
YUSPAGARATAM
WABEEJA
MAITEKA
HUI
SUKSAMA
DHANYABAAD
ANIHA
EKHMET
UNALCHEESH
GUI
HATUR
SPASIBO
MERSI
DENKAUJA
ATTO
NENACHALHYA
EKOJU
SIKOMO
SAINCO
MAAKE
GRAZIE
MEHRBANI
MERASTAWHY
KOMAPSUMNIDA
GAEJTHO
MAKETAI
TAVTAPUCH
LAH
GOZAIMASHITA
MEDAWAGSE
EFCHARISTO
AGUYJE
PALDIES
BAIIKA
FAKAAUE
MINMONCHAR
BOLZÏN
MERCI