SYLACE

INTRODUCTION TO ELASTIC STACK

Mehdi LAMRANI

May 2020

elastic stack

SYLACE

GOALS :

- Provide a broad and comprehensive introduction to Elastic Search Stack
- Get Hands-On experience with queries
- Build a complete pipeline

“ Boxing is a science. You don't just walk into a gym and start punching.”

Eddie Futch

A BIT OF HISTORY

SYLACE

It started with a recipe app

In a London apartment, Shay Banon was looking for a job while his wife attended cooking school at Le Cordon Bleu. In his spare time, he started building a search engine for her growing list of recipes.

His first iteration was called Compass. The second was Elasticsearch (with Apache Lucene under the hood). He open sourced Elasticsearch, created the #elasticsearch IRC channel, and waited for users to appear.

The response was impressive. Users took to it naturally and easily. Adoption went through the roof, a community started to form, and people noticed — namely Steven Schuurman, Uri Boness, and Simon Willnauer. Together, they founded a search company.

SYLACE

- La recherche, on sait faire depuis des décades.
- Moteur de recherche le plus connu ?
- Quoi de nouveau selon vous ?
- Pourquoi maintenant ? Pourquoi un tel succès ?
- Nous allons le voir plus tard… (dans pas trop longtemps… patience ;-))

INVERTED INDEXES

SYLACE

- Forward Index
- A traditional database would typically store the information about each document based on the unique ID of each document :

```
id          keywords
-----------------------------------------------------
Doc1    | quick brown fox jump over lazy dog
Doc2    | quick brown fox leap over lazy dog summer
Doc3    | fox quick jump over bridge
-----------------------------------------------------
```

SYLACE

- Inverted Index

Term	Doc1	Doc2	Doc3
quick		X	X
brown	X	X	
dog	X	X	
fox	X	X	X
jump	X		X
lazy	X	X	
leap		X	
over	X	X	X
quick	X		
summer		X	
bridge			X

SYLACE

- A votre avis quel intérêt ?

MAIN CONCEPTS

SYLACE

documents

Documents are the things you're searching for. They can be more than text – any structured JSON data works. Every document has a unique ID, and a type.

types

A type defines the schema and mapping shared by documents that represent the same sort of thing. (A log entry, an encyclopedia article, etc.)

indices

An index powers search into all documents within a collection of types. They contain inverted indices that let you search across everything within them at once.

SYLACE

- Document Oriented
- Quizz : What is a Document ?
- What are the benefits ?

```
{
  "email":      "john@smith.com",
  "first_name": "John",
  "last_name":  "Smith",
  "info": {
      "bio":          "Eco-warrior and defender of the weak",
      "age":          25,
      "interests": [ "dolphins", "whales" ]
},
  "join_date": "2014/05/01"
}
```

SYLACE

- Documents have no fixed structure
- No predefined Schema
- Nestable
- Easy Notation

SYLACE

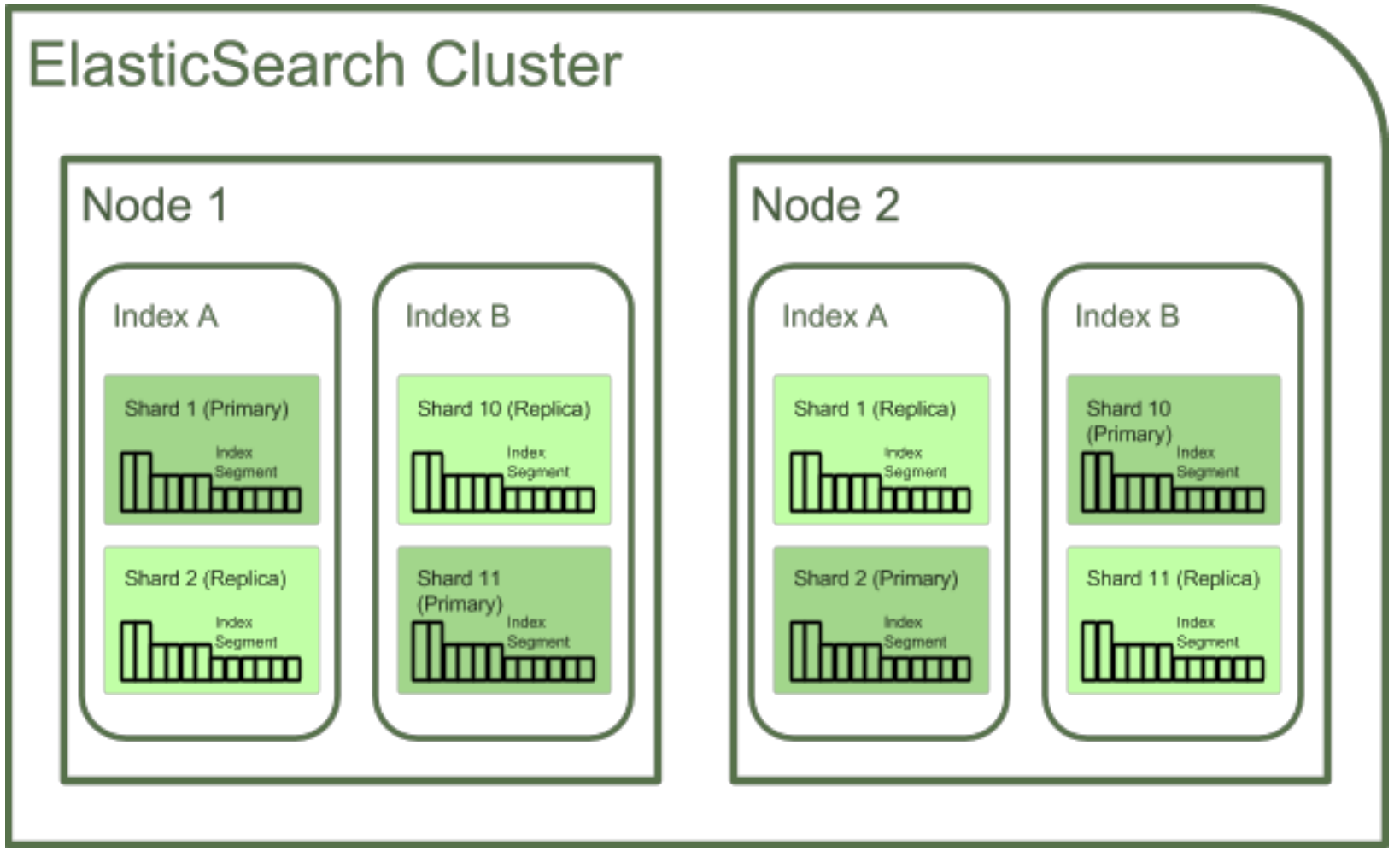
- Relâchement de contraintes
  - => SchemaLess (No structure)
  - => No Transactions
  - => No Relation

- Inversion de Paradigme
  - Structure Defines Content
  - Content Defines Structure

- Le Mapping est inféré du document

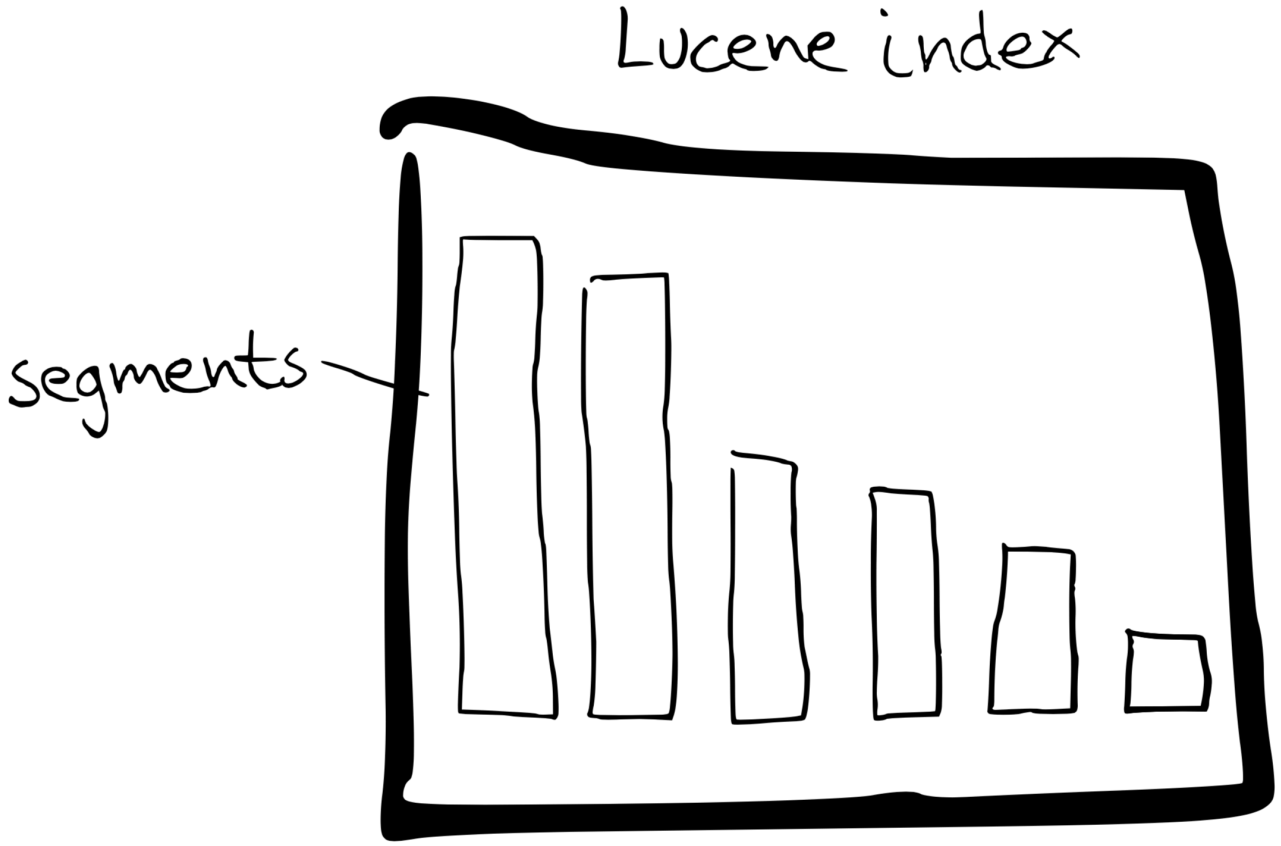
FROM LUCENE TO ELASTIC

SYLACE


ElasticSearch Cluster
Node 1
Index A
Shard 1 (Primary)
Index Segment
Shard 2 (Replica)
Index Segment
Index B
Shard 10 (Replica)
Index Segment
Shard 11 (Primary)
Index Segment
Node 2
Index A
Shard 1 (Replica)
Index Segment
Shard 2 (Primary)
Index Segment
Index B
Shard 10 (Primary)
Index Segment
Shard 11 (Replica)
Index Segment

SYLACE

- Rien de nouveau sous le soleil…



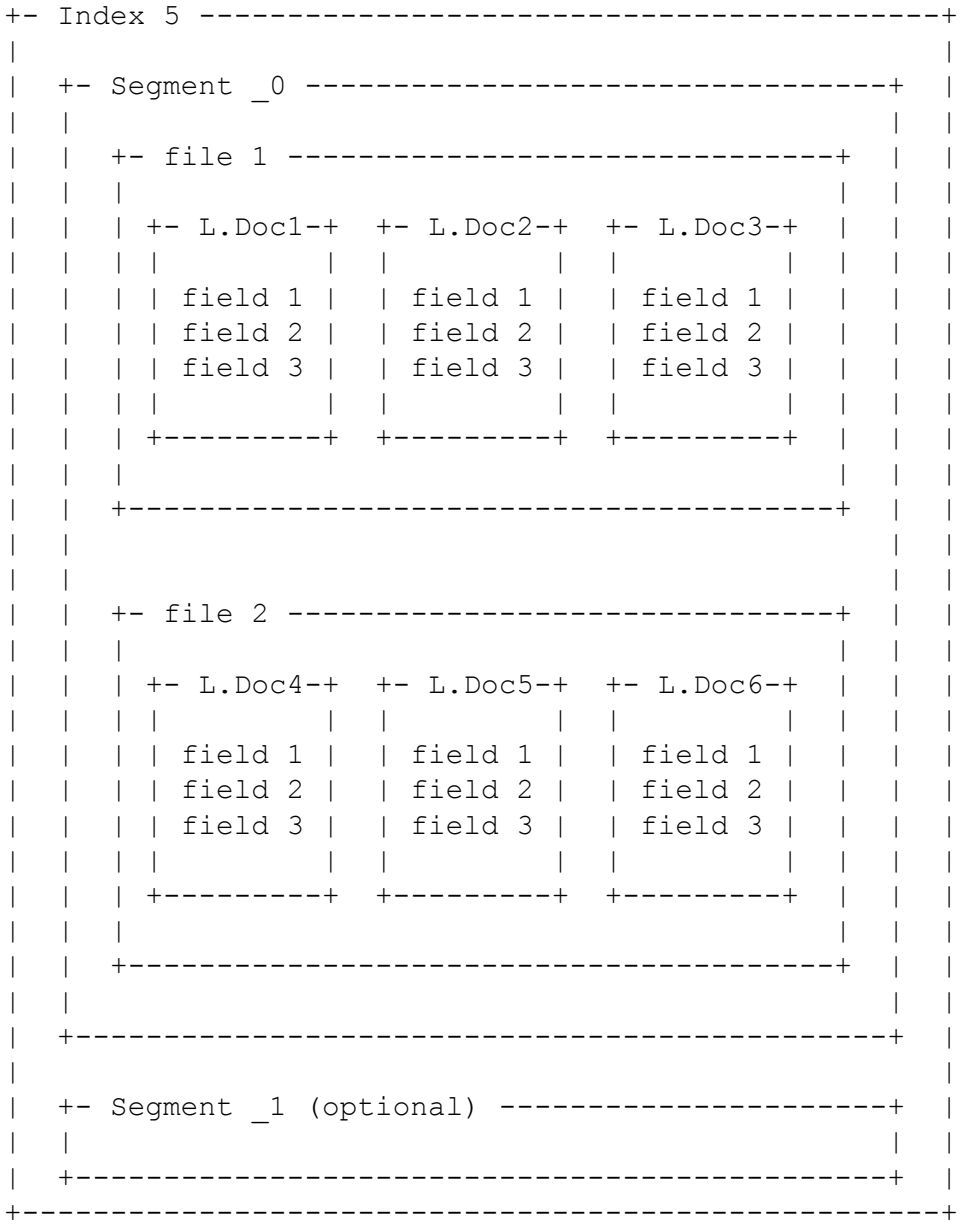
SYLACE

- Lucene Segments

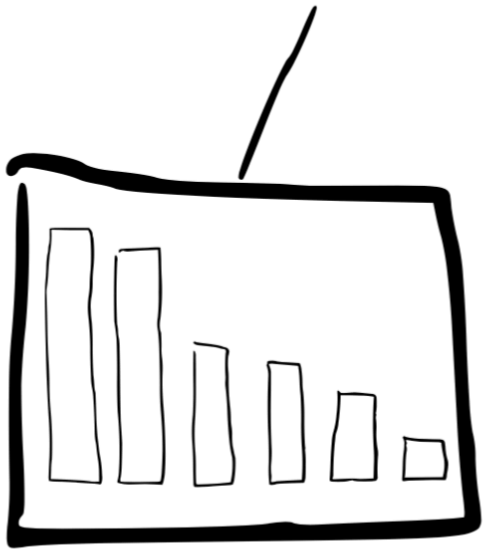
A Lucene segment is part of an Index.

Each segment is composed of several index files.

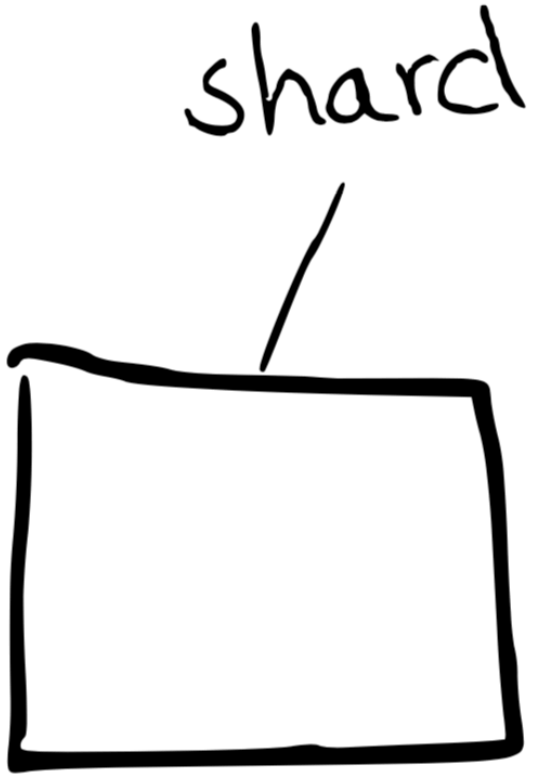
Each index file holds one or more Lucene documents.

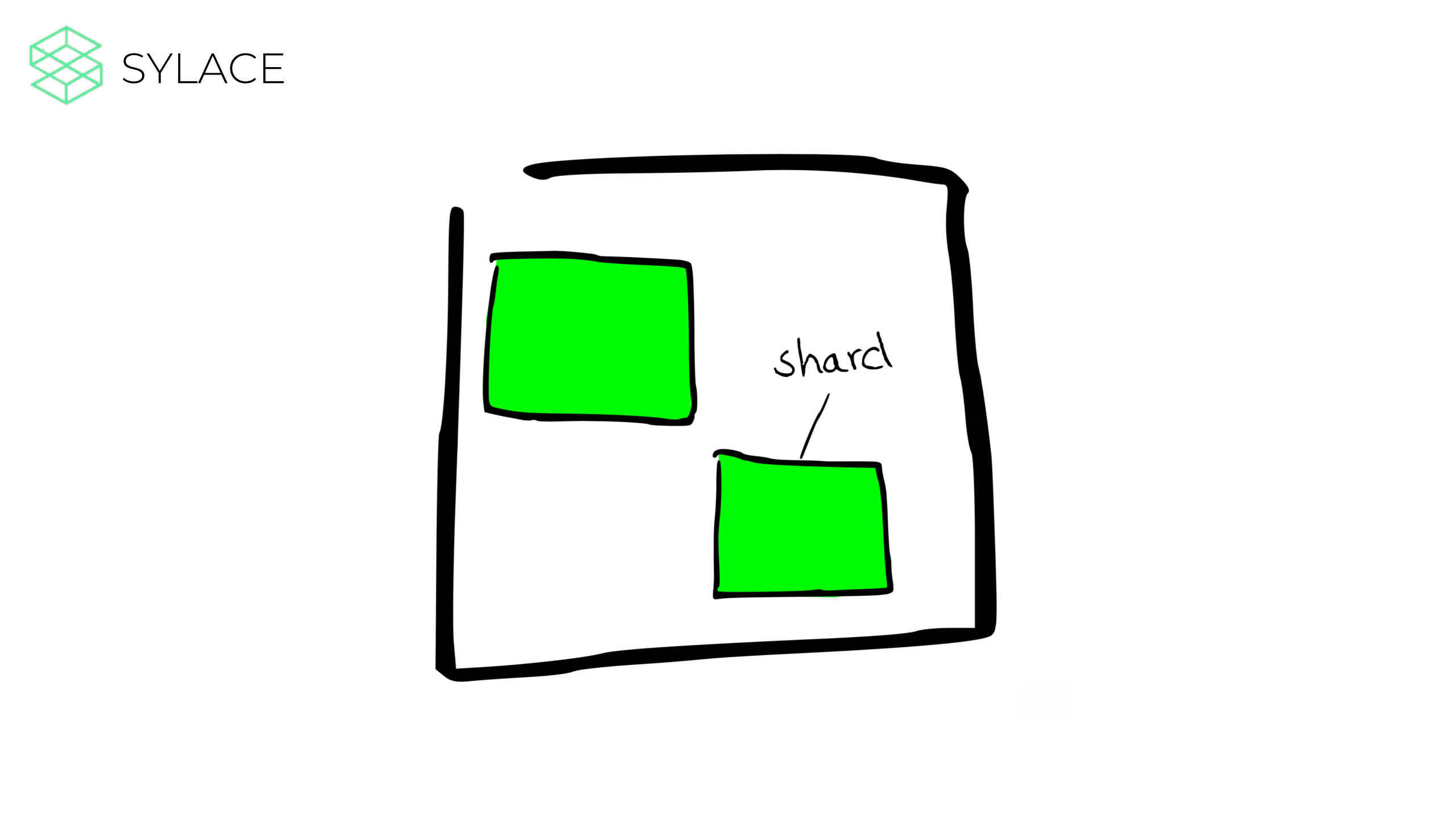


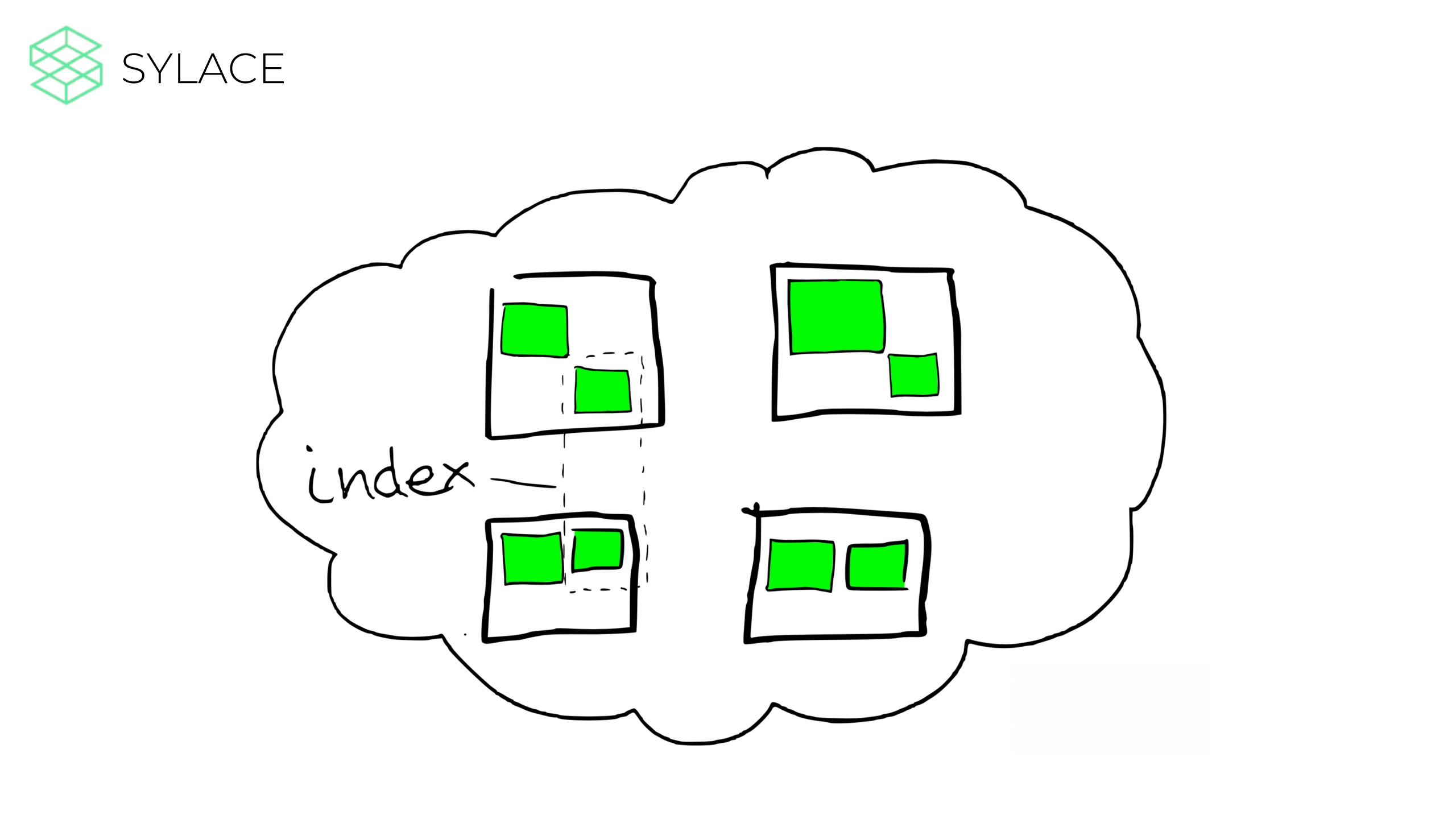
Lucene index



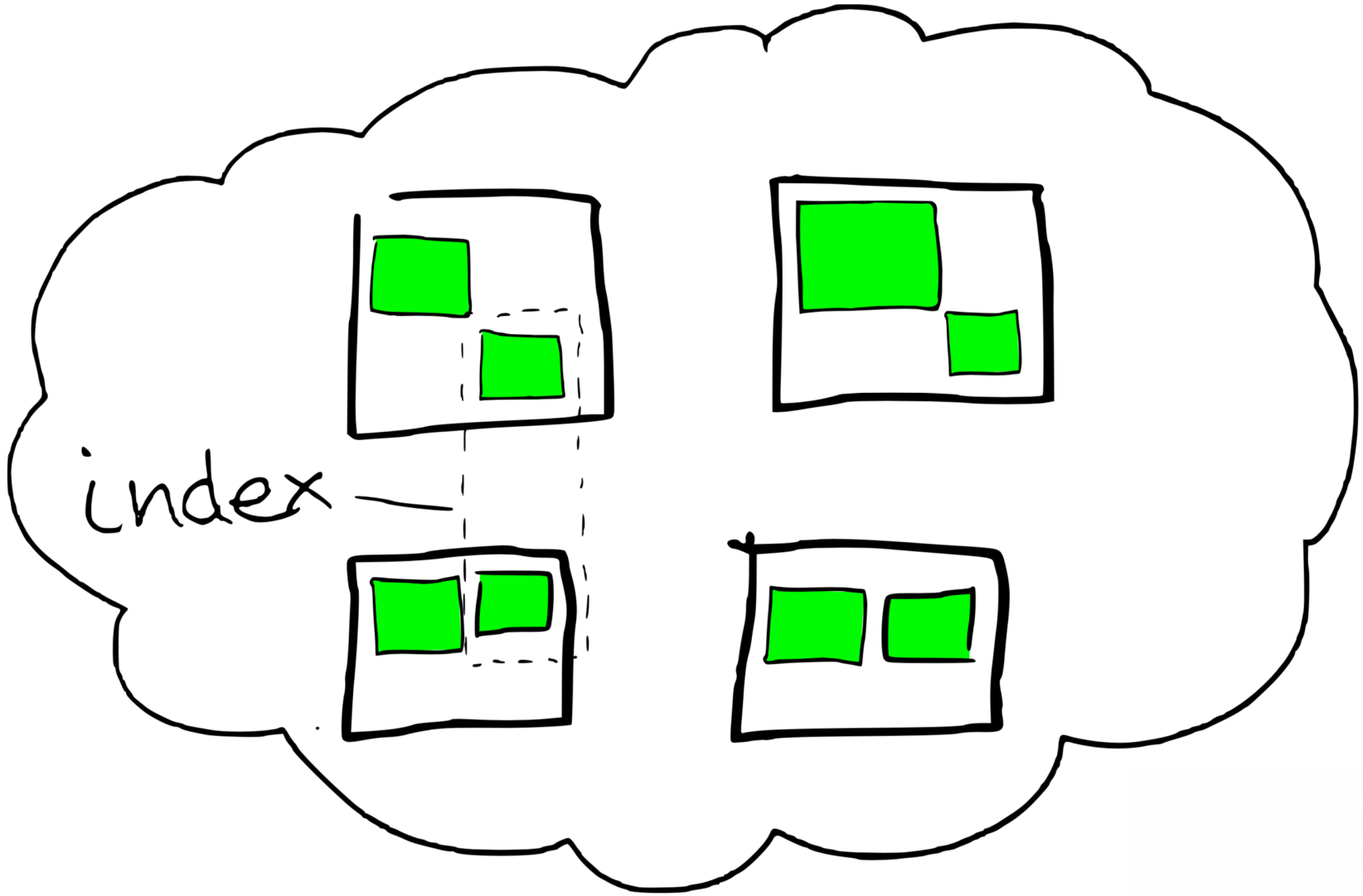
- C’est là que ça commence à devenir intéressant

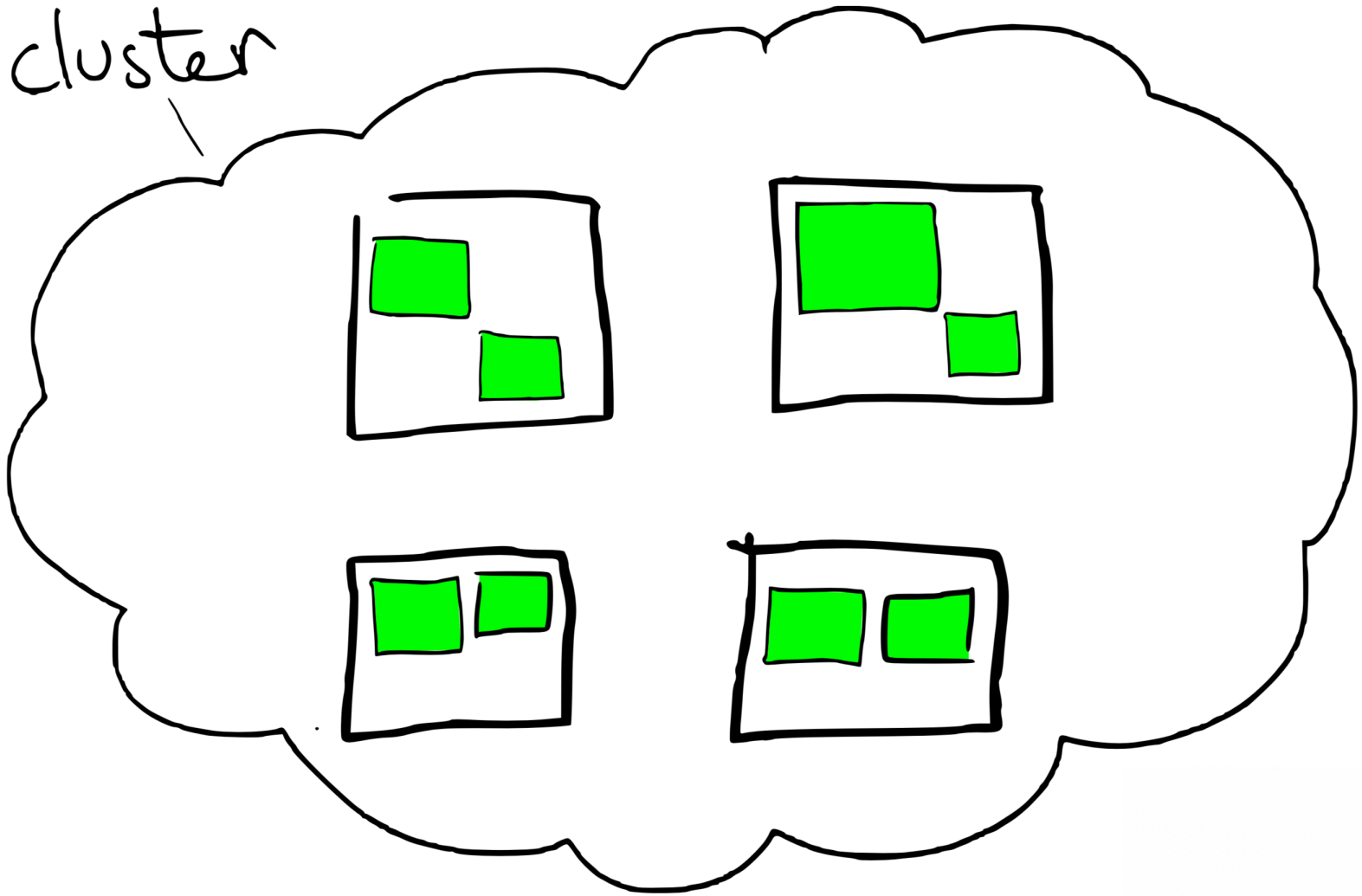



SYLACE
shard

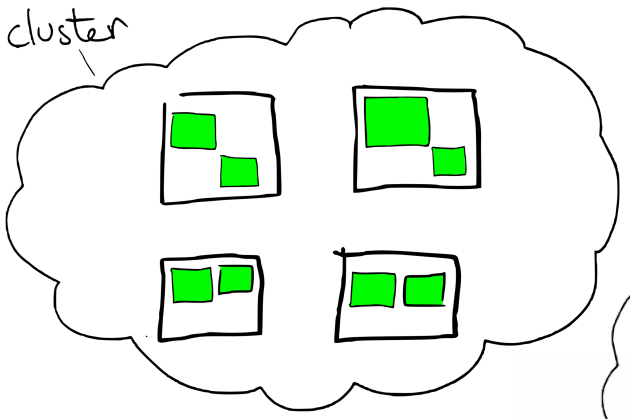

SYLACE
index

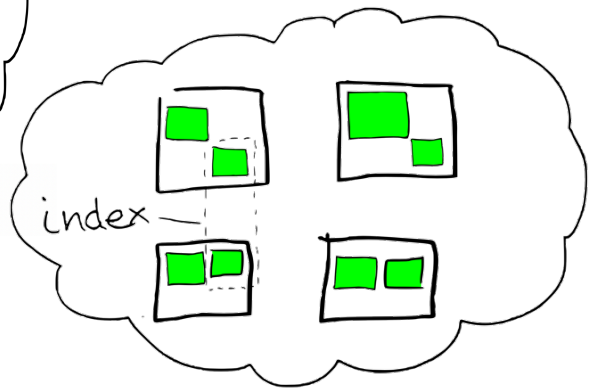
SYLACE

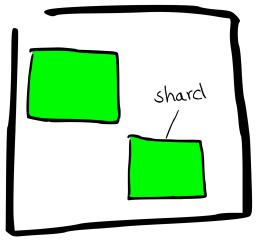


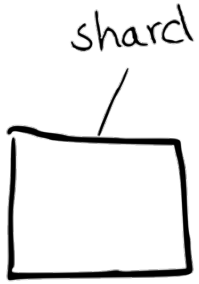

cluster

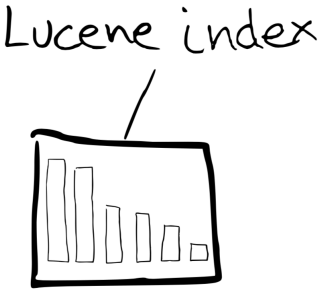
SYLACE

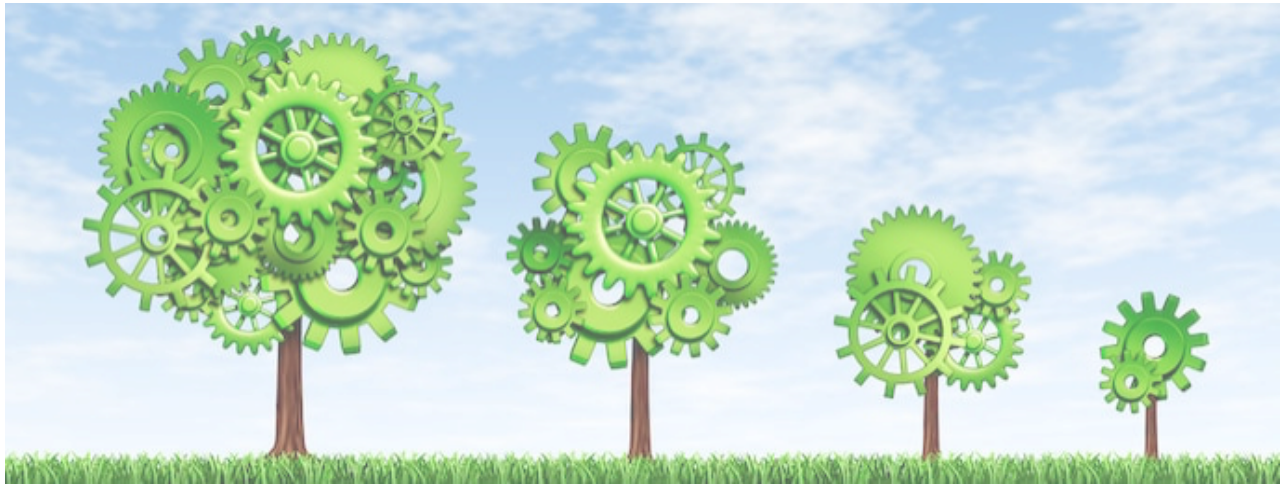

cluster

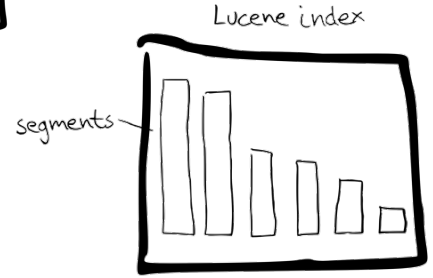

index


shard


shard


Lucene index



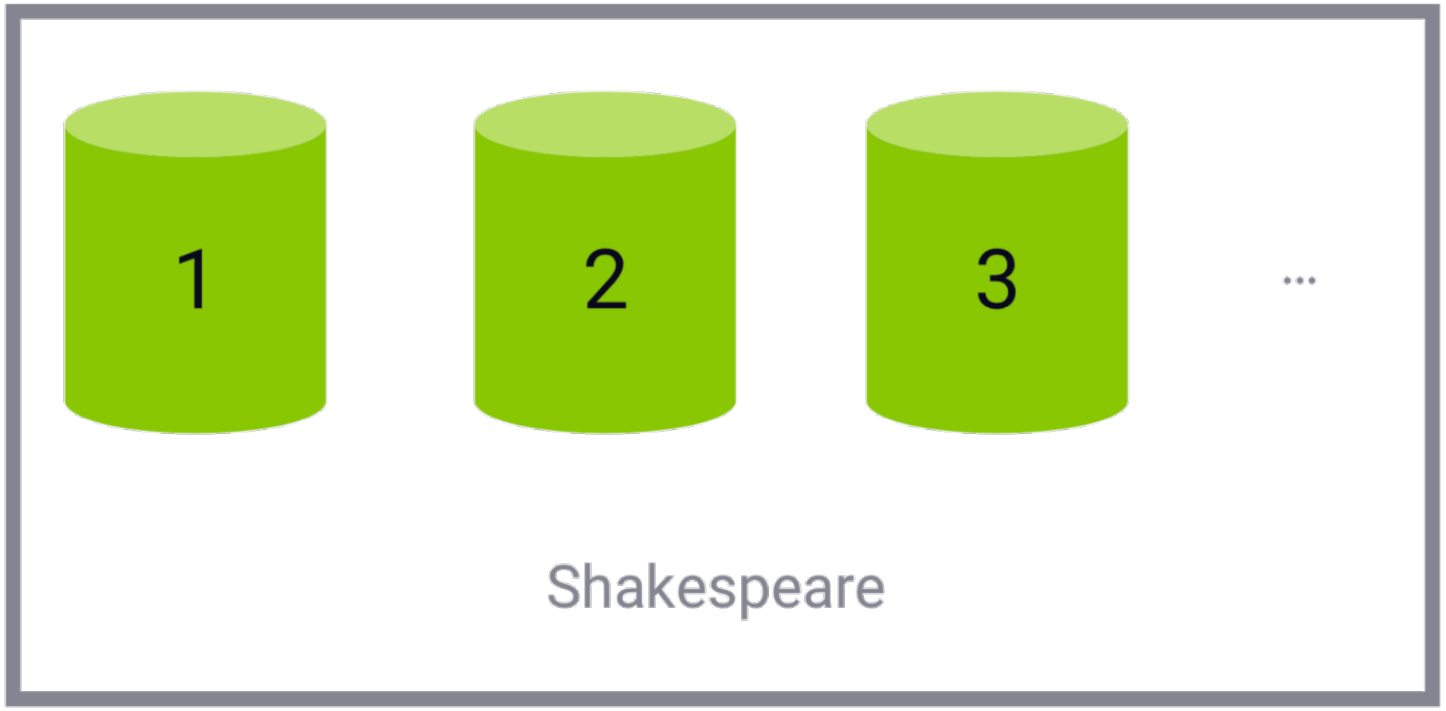

Lucene index
segments

CLUSTER DISTRIBUTION

SYLACE

Distribution

Documents are hashed to a particular shard.

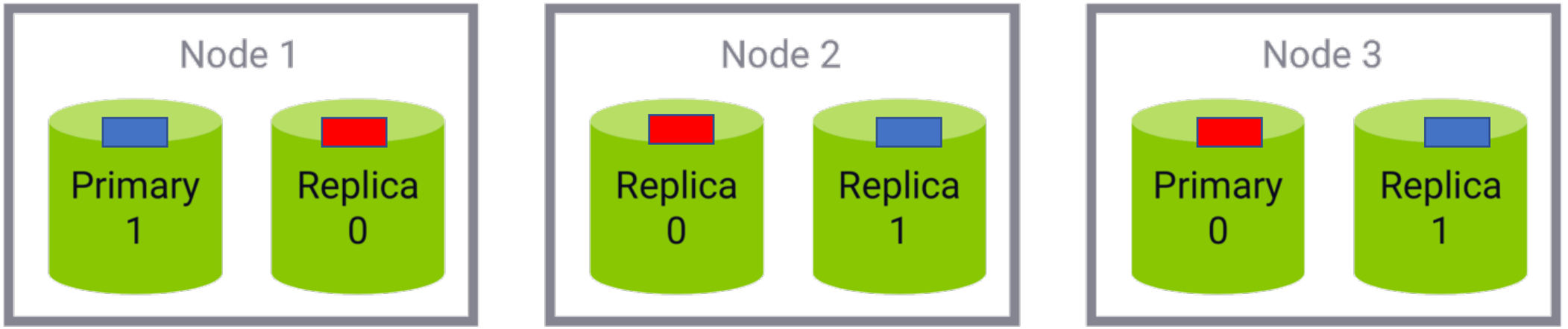


Each shard may be on a different node in a cluster.
Every shard is a self-contained Lucene index of its own.

SYLACE

Réplication

This index has two primary shards and two replicas.
Your application should round-robin requests amongst nodes.



Write requests are routed to the primary shard, then replicated
Read requests are routed to the primary or any replica

- INTEGRATION
  - REST API
- SCALABILITE
    - Distribution
      - Distributed Storage
      - Distributed Computing
- TOLERANCE AUX PANNES
  - Réplication
- POLYMORHISME Linguistique
  - Très grand nombre d’APIs
- TOOLSET & ECOSYSTEM (STACK)

QUERY DSL

SYLACE

- Different from “pure” http requests
- RESTful API with JSON over HTTP
- communicate over port 9200 using a RESTful API
- Accessible
  - from web client or Kibana
  - from the command line by using the curl command.
  - from a language REST API (Java, python, js, etc.)

```
curl -XGET 'http://localhost:9200/_count?pretty' -d '
    {
        "query": {
            "match_all": {}
      }
}
```

- Domains Specific Language is More syntaxically practical than plain URI

SYLACE

- Elasticsearch returns an HTTP status code (like 200 OK)
- a JSON-encoded response body.

```
{
"count" : 10,
        "_shards" : {
            "total" : 5,
            "successful" : 5,
"failed" : 0 }
}
```

SYLACE

- Elasticsearch provides official clients for several language
- Groovy, JavaScript, .NET, PHP, Perl, Python, and Ruby, Java
- and there are numerous community-provided clients

QUICK TOUR OF THE ELK STACK

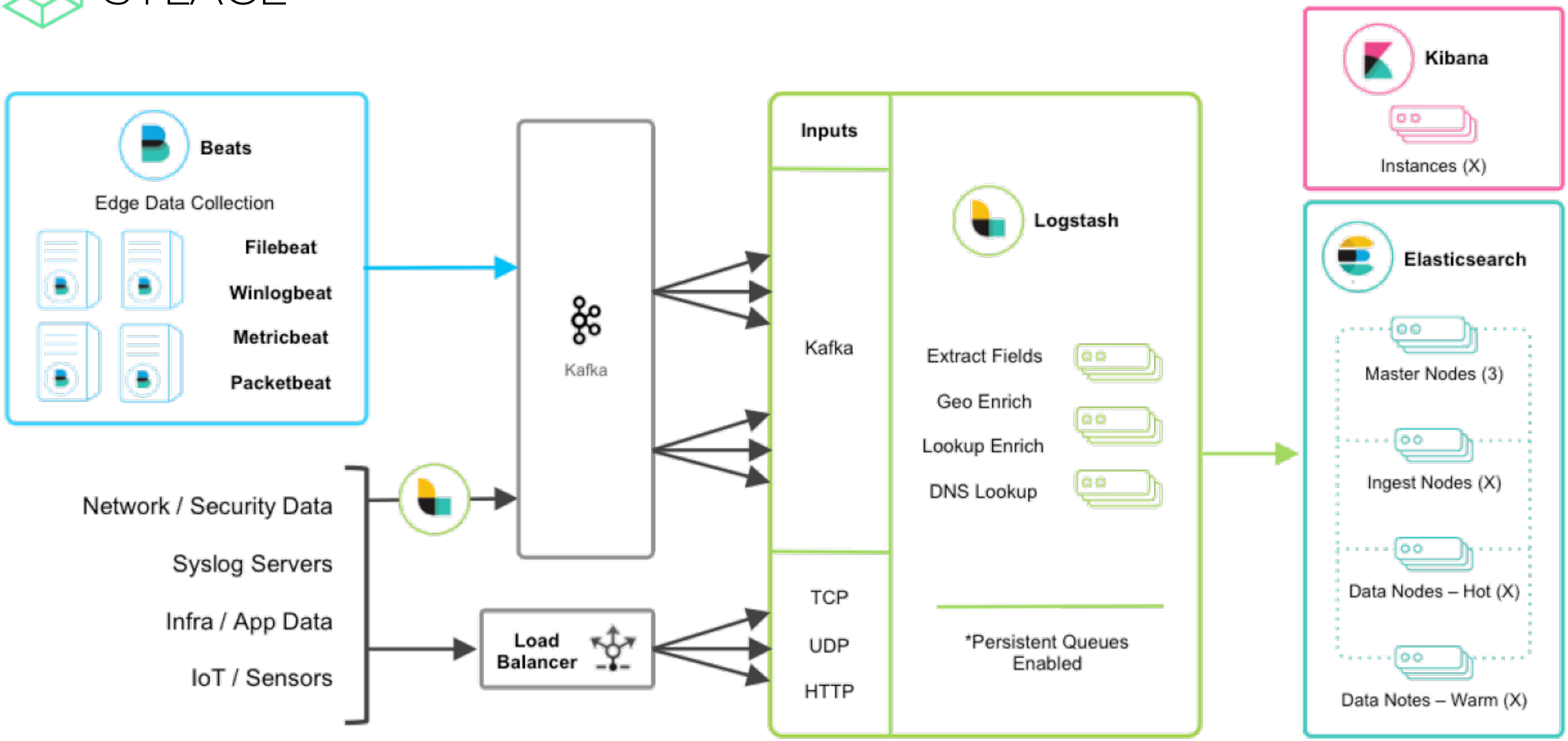
Data Collection

Data Aggregation & Processing

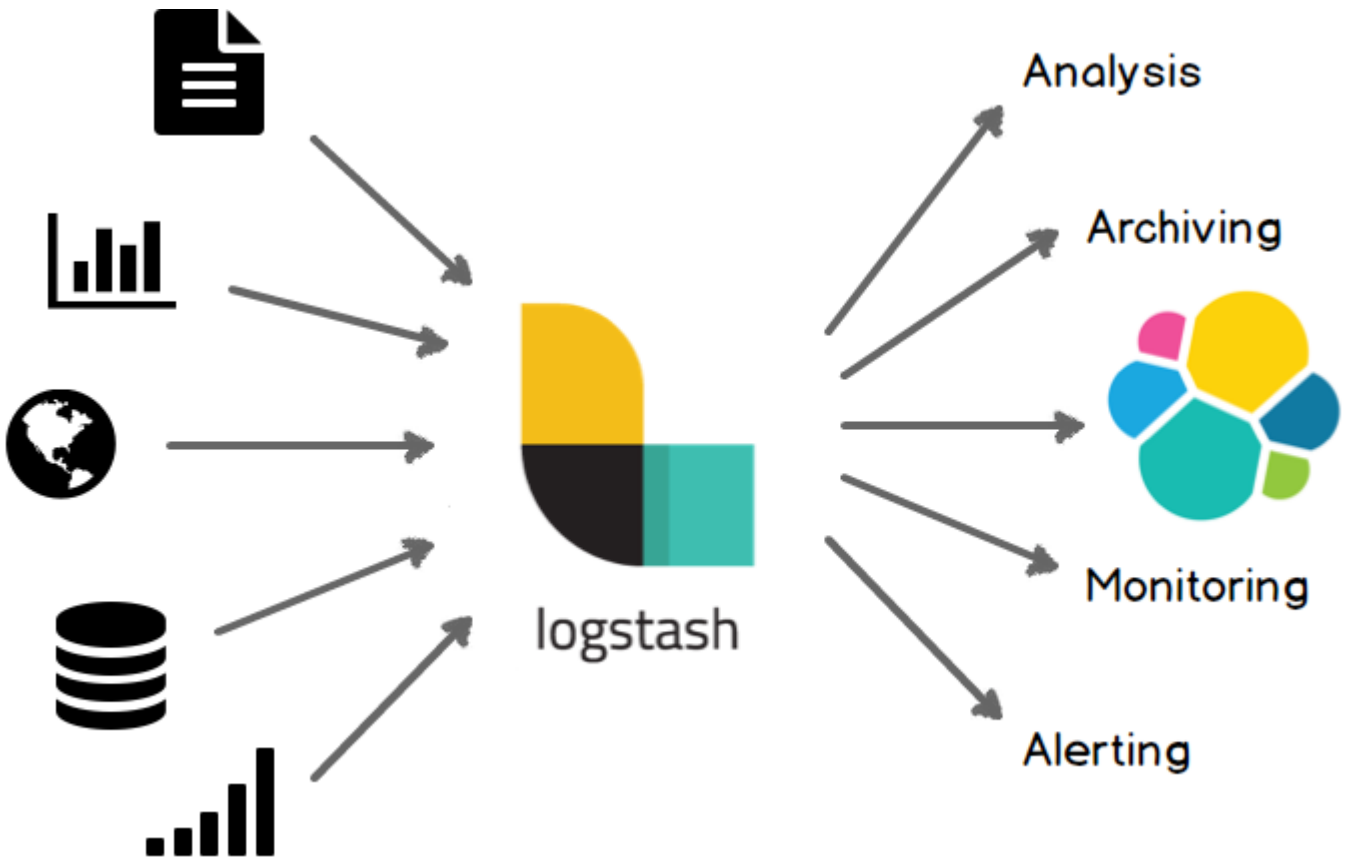
Indexing & storage

Analysis & visualization

SYLACE


Beats
Edge Data Collection
Filebeat
Winlogbeat
Metricbeat
Packetbeat
Network / Security Data
Syslog Servers
Infra / App Data
IoT / Sensors
Kafka
Load Balancer
Inputs
Kafka
TCP
UDP
HTTP
Logstash
Extract Fields
Geo Enrich
Lookup Enrich
DNS Lookup
*Persistent Queues Enabled
Kibana
Instances (X)
Elasticsearch
Master Nodes (3)
Ingest Nodes (X)
Data Nodes – Hot (X)
Data Notes – Warm (X)

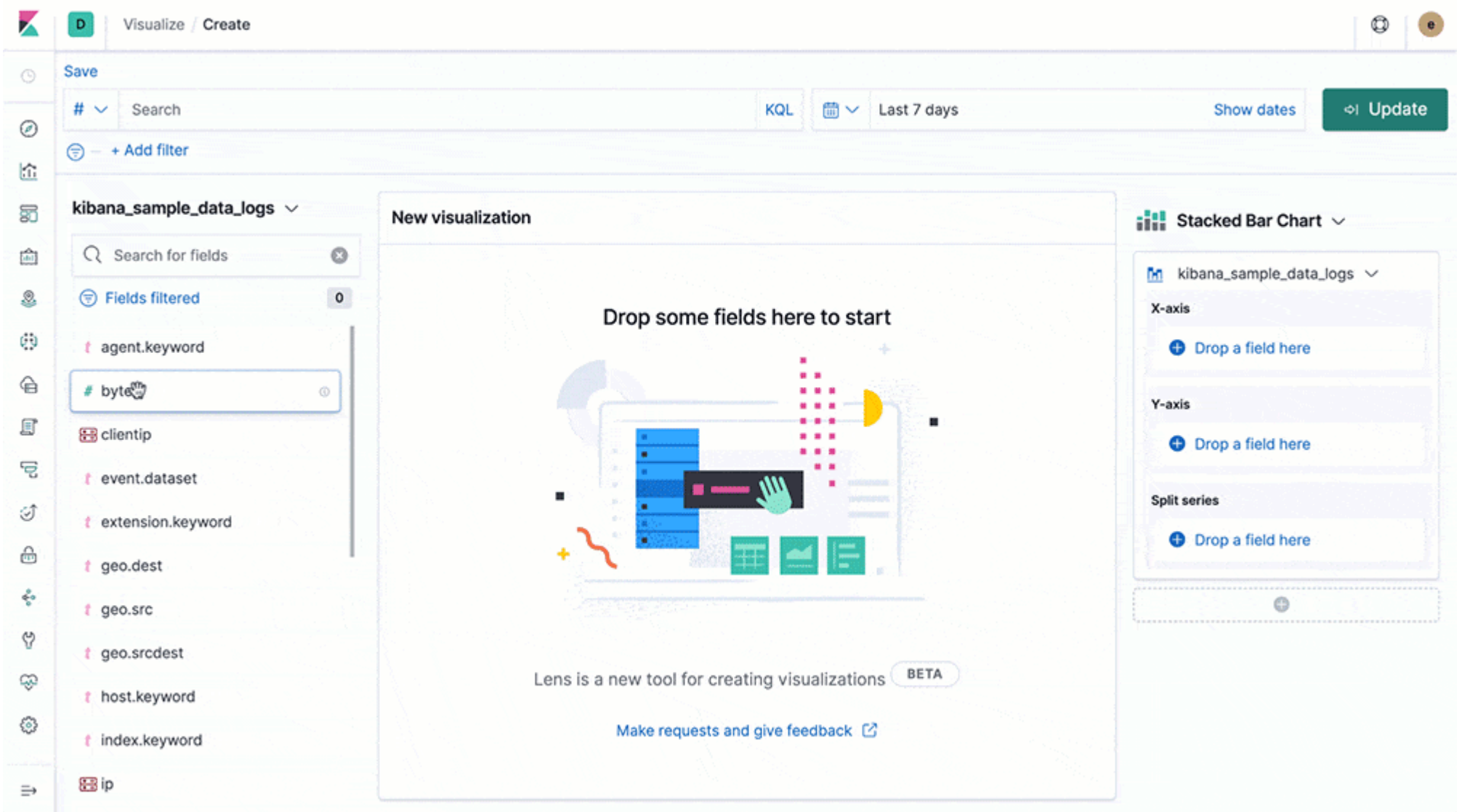

logstash


Analysis
Archiving
Monitoring
Alerting
logstash

SYLACE

kibana

SYLACE


Visualize / Create
Save
Search
KQL
Last 7 days
Show dates
Update
+ Add filter
kibana_sample_data_logs
Search for fields
Fields filtered
0
agent.keyword
byte
clientip
event.dataset
extension.keyword
geo.dest
geo.src
geo.srcdest
host.keyword
index.keyword
ip
New visualization
Drop some fields here to start
Lens is a new tool for creating visualizations
BETA
Make requests and give feedback
Stacked Bar Chart
kibana_sample_data_logs
X-axis
Drop a field here
Y-axis
Drop a field here
Split series
Drop a field here

INSTALLATION TIME !

SYLACE

- https://github.com/mehdi-lamrani/elasticsearch-workshop

Exercices 01

Getting Started

Document Management

PLAYING AROUND WITH THE API

Document Management

PLAYING AROUND WITH THE API

CRUD OPERATIONS

SYLACE

- INSERT A DOCUMENT

```
curl -XPUT 127.0.0.1:9200/index/doc/1234 -d '


curl -XPUT 127.0.0.1:9200/movies/movie/1234 -d '
{
    "genre" : ["IMAX","Sci-Fi"],
    "title" : "Interstellar",
    "year" : 2014
}
```

SYLACE

- BULK IMPORT

```
curl -XPUT 127.0.0.1:9200/_bulk -d '
{ "create" : { "_index" : "movies", "_type" : "movie", "_id" : "135569" } }
{ "id": "135569", "title" : "Star Trek Beyond", "year":2016 , "genre":["Action", "Adventure", "Sci-Fi"] }
{ "create" : { "_index" : "movies", "_type" : "movie", "_id" : "122886" } }
{ "id": "122886", "title" : "Star Wars: Episode VII ", "year":2015 , "genre":["Adventure", "Sci-Fi",} {
"create" : { "_index" : "movies", "_type" : "movie", "_id" :   "1286" } }
{ "id": "109487", "title" : "Interstellar", "year":2014 , "genre":["Sci-Fi", "IMAX"] }
{ "create" : { "_index" : "movies", "_type" : "movie", "_id" : "58559" } }
{ "id": "58559", "title" : "Dark Knight, The", "year":2008 , "genre":["Action", "Crime", "Drama", "IMAX"] }
{ "create" : { "_index" : "movies", "_type" : "movie", "_id" : "1924" } }
{ "id": "1924", "title" : "Plan 9 from Outer Space", "year":1959 , "genre":["Horror", "Sci-Fi"] } '
```

SYLACE

- PARTIAL UPDATE

```
curl -XPOST 127.0.0.1:9200/movies/movie/109487/_update -d '
{
    "doc": {
            "title": "Interstellar"
    }
}
```

SYLACE

- DELETE

```
curl -XDELETE 127.0.0.1:9200/movies/movie/58559
```

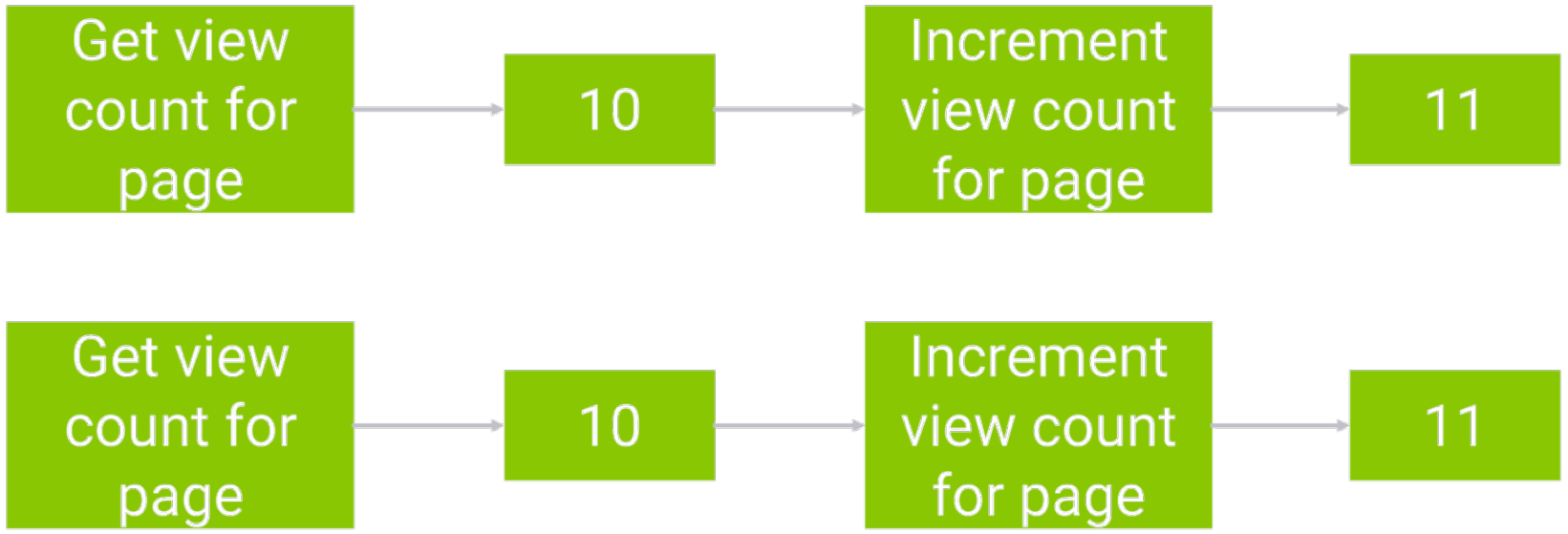
Exercices 02.01>08

Managing Documents

CONCURRENCY MANAGEMENT

SYLACE

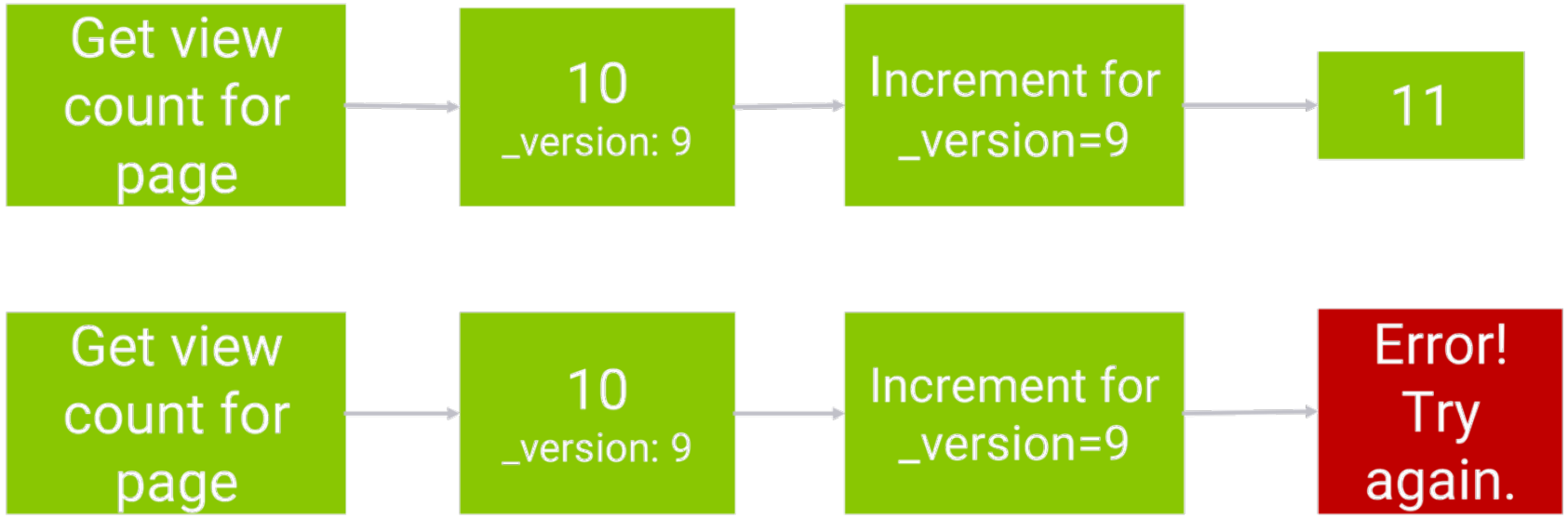
- Problem :



But it should be 12!

SYLACE

- Solution : Versioning



Use retry_on_conflicts=N to automatically retry.

Exercice 02.09

Optimistic Concurrency Control

MAPPING

SYLACE

- MAPPING

```
{
    "twitter":{
      "mappings":{
         "tweet":{
            "properties":{
               "date":{
                  "type":"date",
                  "format":"dateOptionalTime"
               },
               "name":{
                  "type":"string"
               },
               "tweet":{
                  "type":"string"
               },
               "user_id":{
                  "type":"long"
               }
            }
         }
      }
   }
}
```

SYLACE

- Core Simple Field Types
  - String
  - Whole number:
    - byte, short, integer, long
  - Floating-point:
    - float,double
  - Boolean
  - Date

- Mapping is DYNAMIC
- Elasticsearch will use dynamic mapping to try to guess the field type from the basic datatypes available in JSON

SYLACE

- Common Mapping Types

field types

text, keyword, byte, short, integer, long, float, double, boolean, date

```
"properties": {
   "user_id" : {
       "type": "long"
    }
}
```

field index

do you want this field to be queryable? true / false

```
"properties": {
   "genre" : {
      "index": "false"
    }
}
```

field analyzer

define your tokenizer and token filter. standard / whitespace / simple / english etc.

```
"properties": {
   "description" : {
      "analyzer": "english"
    }
}
```

SYLACE

Updating a Mapping

- You can specify the mapping for a type when you first create an index.
- Later, you can :
  - add the mapping for a new type
  - update the mapping for an existing type,
  - using the /_mapping endpoint.
- Although you can add to an existing mapping, you can’t change it.
- If a field already exists in the mapping, the data from that field probably has already been indexed.
- If you were to change the field mapping, the already indexed data would be wrong and would not be properly searchable.
- We can update a mapping to add a new field, but we can’t change an existing field from analyzed to not_analyzed.

Exercices 03

Mapping Management

ANALYZERS

SYLACE

Analysis is a process that consists of the following:

- First, tokenizing a block of text into individual terms suitable for use in an inverted index,
- Then normalizing these terms into a standard form to improve their “searchability,” or recall

This job is performed by analyzers.
An analyzer is really just a wrapper that combines three functions into a single package

SYLACE

1. Character filters
   - the string is passed through any character filters in turn.
   - Their job is to tidy up the string before tokenization.
   - A character filter could be used to strip out HTML, or to convert ‘&’ characters to the word ‘and’.

2. Tokenizer
   - the string is tokenized into individual terms by a tokenizer. `
   - A simple tokenizer might split the text into terms whenever it encounters whitespace or punctuation.

3. Token filters
   - each term is passed through any token filters in turn,
   - which can
     - change terms (for example, lowercasing Quick),
     - remove terms (for example, stopwords such as a, and, the)
     - or add terms (for example, synonyms like jump and leap).

SYLACE

- Built-in Analyzers

  `"Set the shape to semi-transparent by calling set_trans(5)"`

  - Standard analyzer

    `set, the, shape, to, semi, transparent, by, calling, set_trans, 5`

  - Simple analyzer

    `set, the, shape, to, semi, transparent, by, calling, set, trans`

  - Whitespace analyzer

    `Set, the, shape, to, semi-transparent, by, calling, set_trans(5)`

  - Language analyzer

    `set, shape, semi, transparent, call, set_trans, 5`

SYLACE

Algorithmic stemmers

- Algorithmic stemmers apply a series of rules to each word to reduce it to its root form.
- For example, an algorithmic stemmer for English may remove the -s and -es prefixes from the end of plural words.

Lemmatization

A lemma is the canonical, or dictionary, form of a set of related words—the lemma of paying, paid, and pays is pay. Usually the lemma resembles the words it is related to but sometimes it doesn't
— the lemma of is, was, am, and being is be.

Exercices 04

Analyzers

Exercices 04

Analyzers

TEXT & TERM QUERIES

SYLACE

Term-level queries

- term-level queries are used to find documents based on precise values in structured data. Examples of structured data include date ranges, IP addresses, prices, or product IDs.

- term-level queries do NOT analyze search terms.
- Instead, term-level queries match the exact terms stored in a field.

SYLACE

term query
Returns documents that contain an exact term in a provided field.

terms query
Returns documents that contain one or more exact terms in a provided field.

terms_set query
Returns documents that contain a minimum number of exact terms in a provided field. You can define the minimum number of matching terms using a field or script.

type query

Returns documents of the specified type.

wildcard query
Returns documents that contain terms matching a wildcard pattern.

SYLACE

exists query
Returns documents that contain any indexed value for a field.

fuzzy query
Returns documents that contain terms similar to the search term.
Elasticsearch measures similarity, or fuzziness

distance.ids query
Returns documents based on their document IDs.
prefix query
Returns documents that contain a specific prefix in a provided field.
range query
Returns documents that contain terms within a provided range.
regexp query
Returns documents that contain terms matching a regular expression.

SYLACE

Full Text queries

- The full text queries enable you to search analyzed text fields such as the body of an email.
- The query string is processed using the same analyzer that was applied to the field during indexing.

SYLACE

[intervals query]

A full text query that allows fine-grained control of the ordering and proximity of matching terms.

[match query]

The standard query for performing full text queries, including fuzzy matching and phrase or proximity queries.

[match_bool_prefix query]

Creates a bool query that matches each term as a term query, except for the last term, which is matched as a prefix query

[match_phrase query]

Like the match query but used for matching exact phrases or word proximity matches.

[match_phrase_prefix query]

Like the match_phrase query, but does a wildcard search on the final word.

SYLACE

multi_match query

The multi-field version of the match query.

common terms query

A more specialized query which gives more preference to uncommon words.

query_string query

Supports the compact Lucene query string syntax, allowing you to specify AND|OR|NOT conditions and multi-field search within a single query string. For expert users only.

simple_query_string query

A simpler, more robust version of the query_string syntax suitable for exposing directly to users.