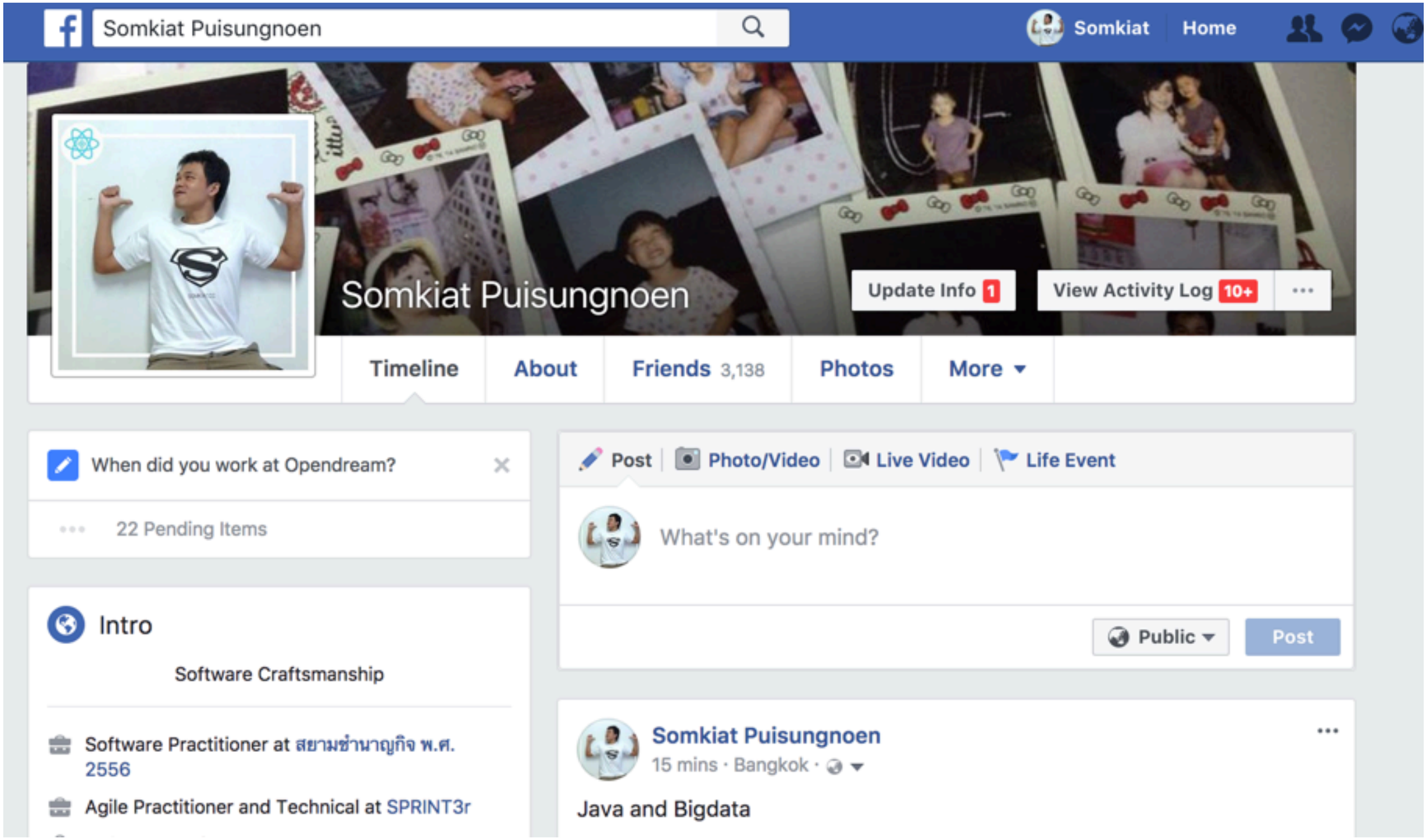
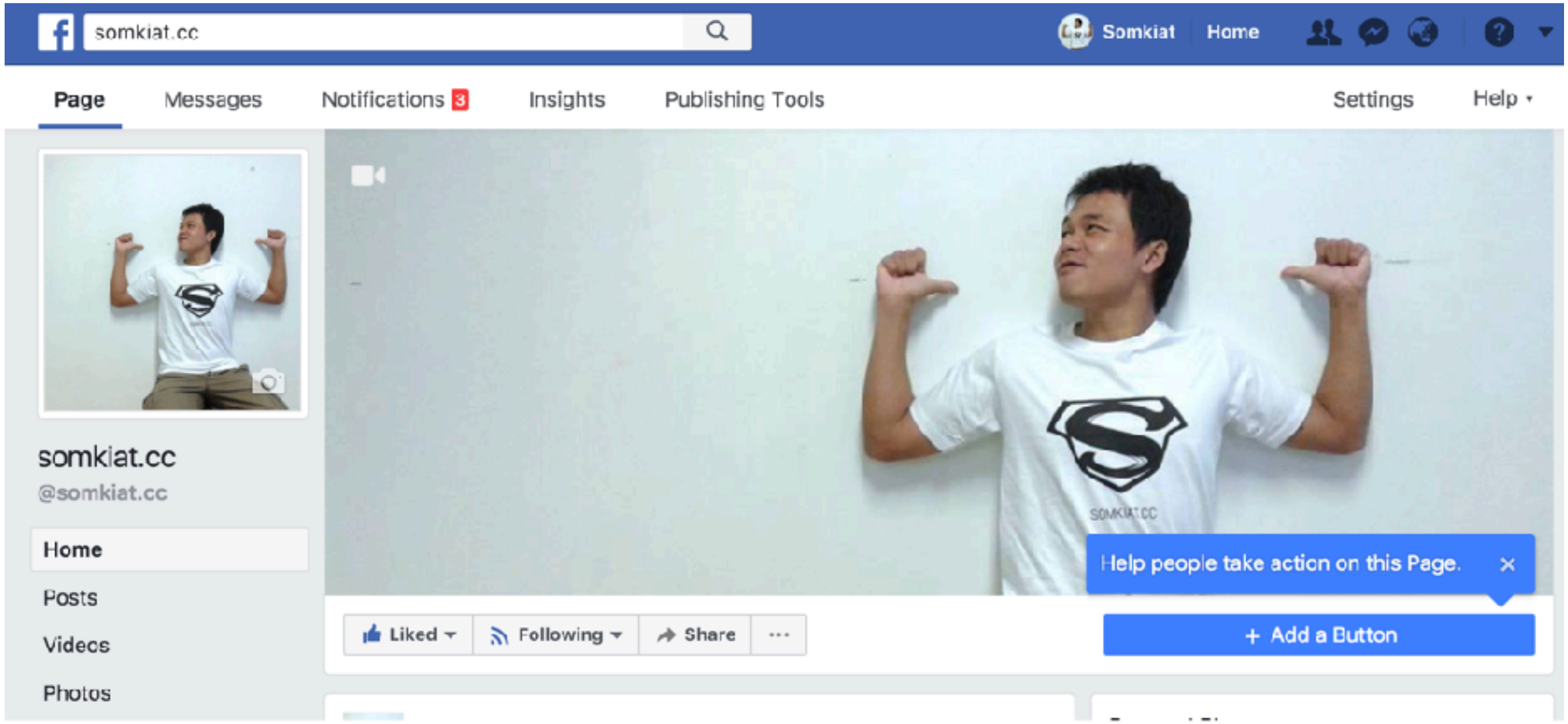
Elasticsearch

Data modeling
Reduce size of data storage
Search improvement

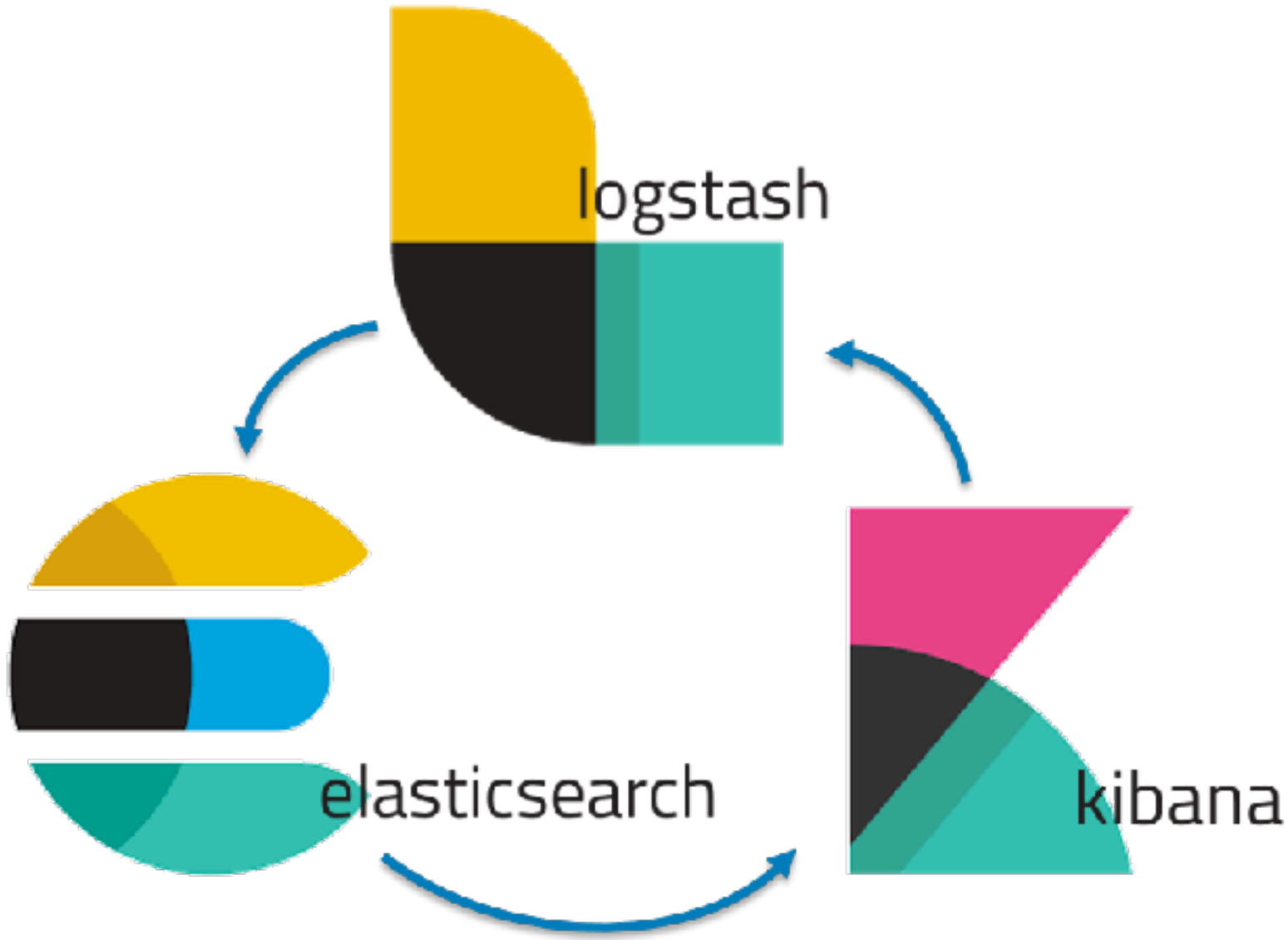




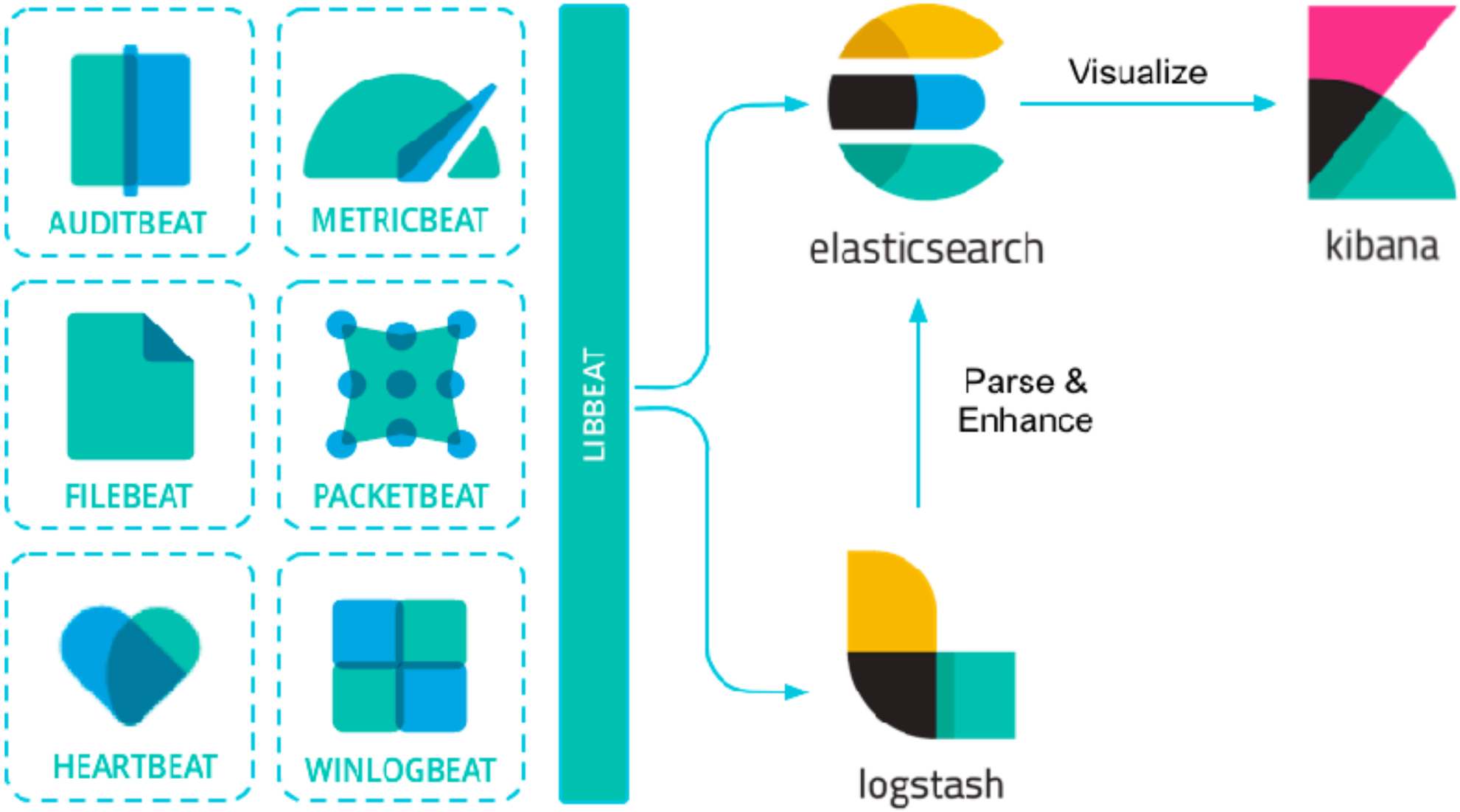
My Journey with ES

Data Modeling

Elasticsearch

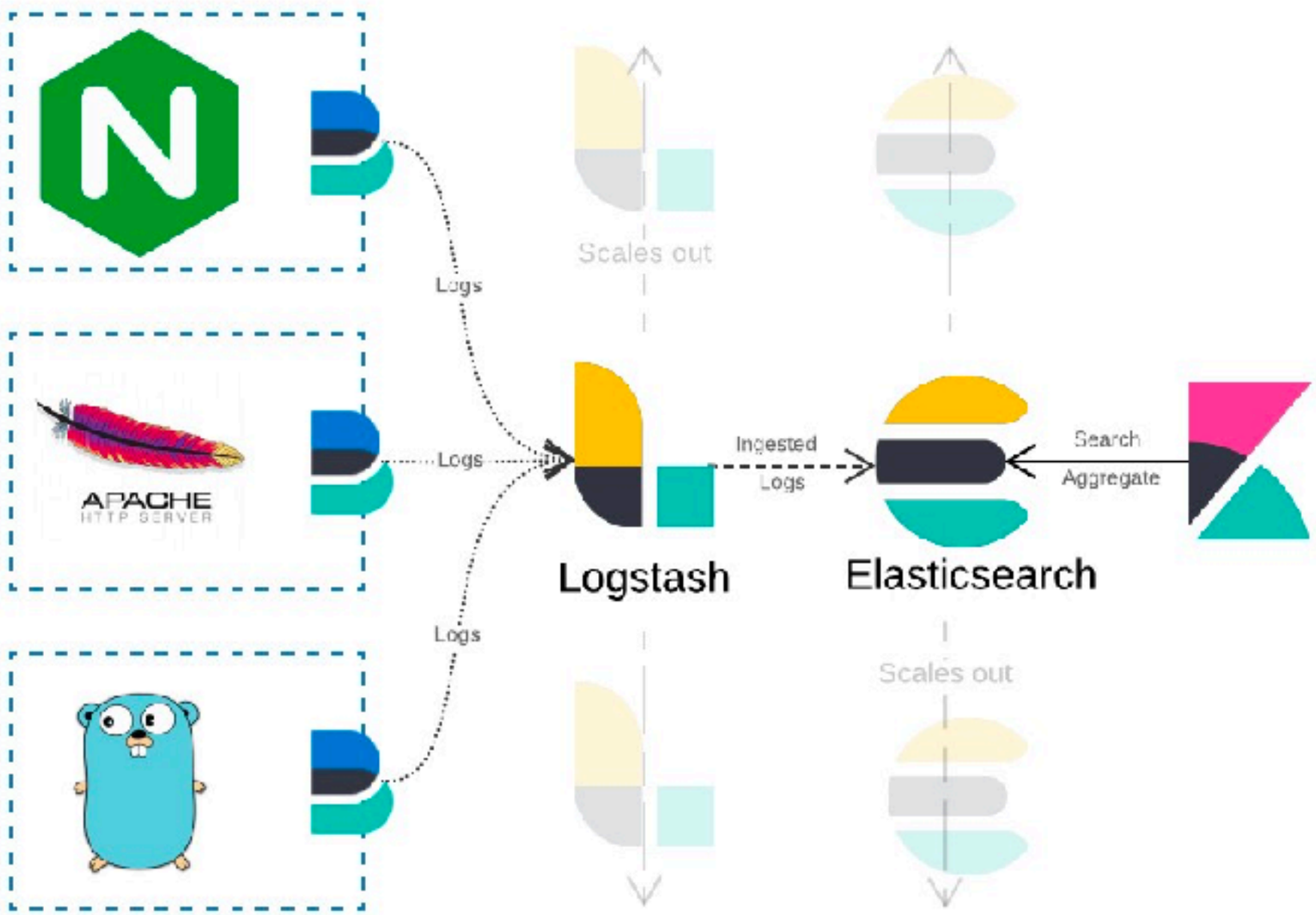


Beat



https://www.elastic.co/guide/en/beats/libbeat/current/index.html

ELK + Beats



Document based

JSON (JavaScript Object Notation)
Dynamic Schema (Schema-less)
Some relationship (nested, parent/child)

StackOverflow Question


```
{
  "items": [
    {
      "owner": {
        "reputation": 13,
        "user_id": 9796344,
        "user_type": "registered",
        "profile_image": "",
        "display_name": "Cherry",
        "link": "https://stackoverflow.com/users/9796344/cherry"
      },
      "score": 0,
      "last_activity_date": 1528986761,
      "creation_date": 1528986761,
      "post_type": "question",
      "post_id": 50859951,
      "link": "https://stackoverflow.com/q/50859951"
    }
  ],
  "has_more": false,
  "quota_max": 10000,
  "quota_remaining": 9986
}
```

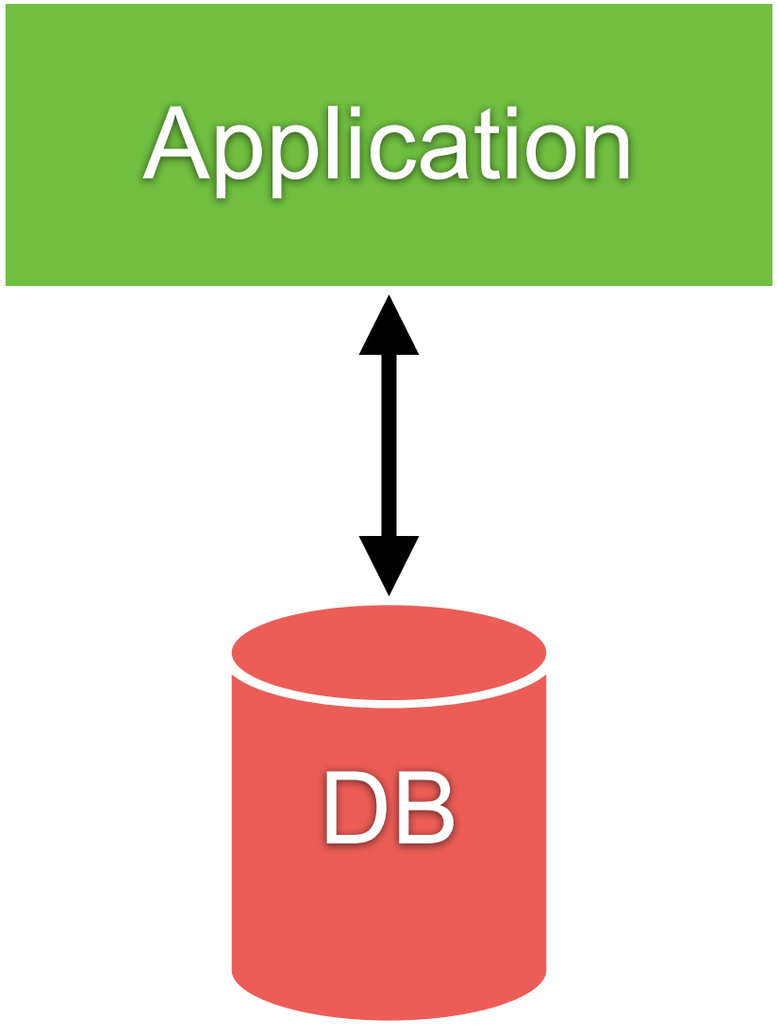

https://api.stackexchange.com/docs/posts-by-ids

Use cases

Security/log analytics
Marketing
Operations
Searching data

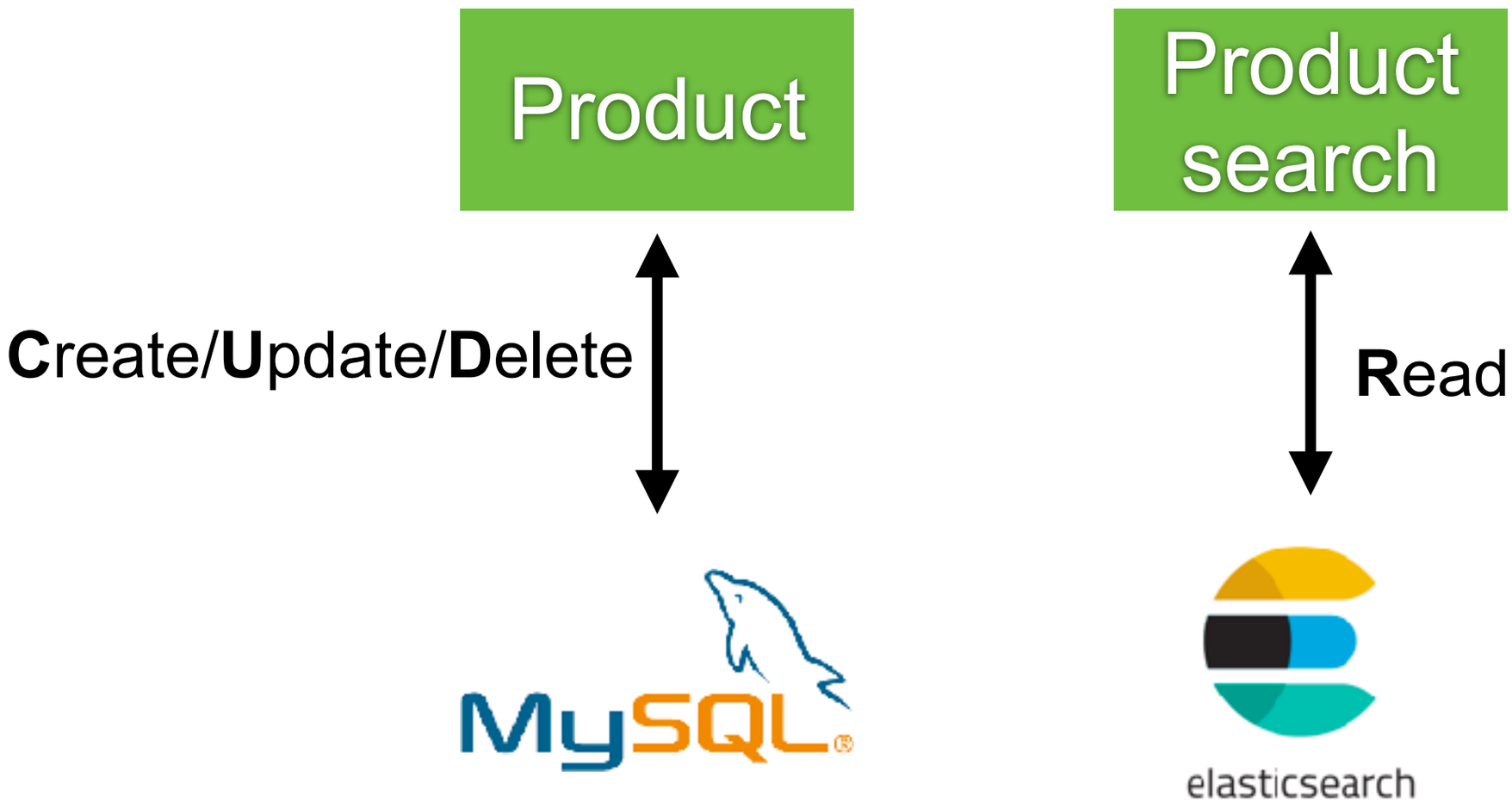
Problem ?

Single/Centralize database



Separate data for read and write

For example MySQL to write, Elasticsearch to search



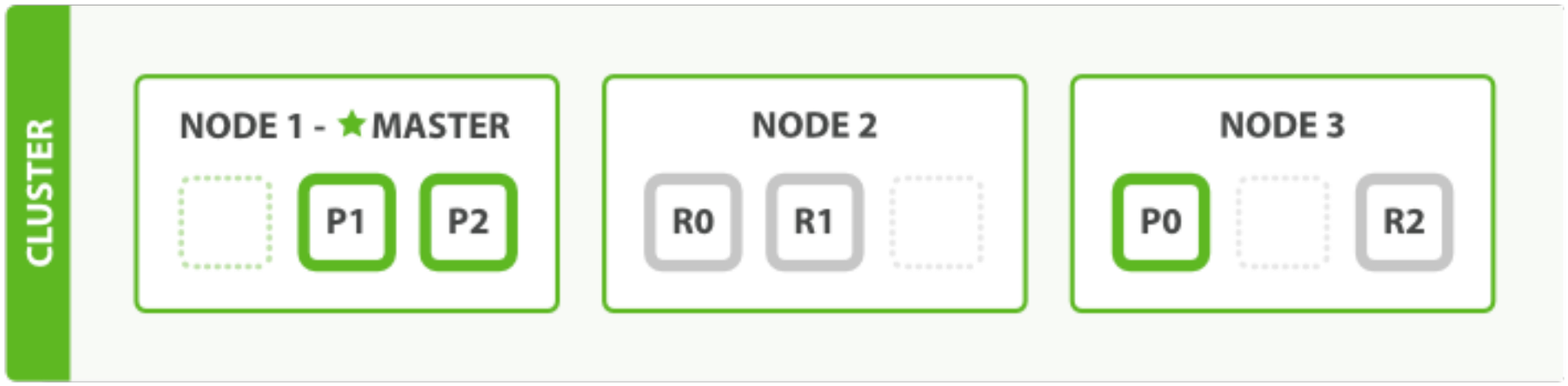
Basic of Elasticsearch

Cluster
Indices
Mapping
Analysis

Cluster

Cluster

Collection of nodes



Node Roles

CAT Nodes		FEATURE USAGE
m	master-eligible	Master
v	voting_only	Tiebreaker Master
d	data	-
s	data_content	-
h	data_hot	ILM
w	data_warm	...
c	data_cold	...
f	data_frozen	..., Searchable Snapshots
i	ingest	Stack Monitoring, Ingest Pipelines
l	ml	Machine Learning
r	remote_cluster_client	Cross-cluster search/replication
t	transform	Transform, Fleet, SIEM
-	coordinating	(implicit)

data tiers

https://www.elastic.co/guide/en/elasticsearch/reference/current/modules-node.html

Node Roles

Master

Data

Coordinate

https://www.elastic.co/guide/en/elasticsearch/reference/current/modules-node.html

Node Roles

Master

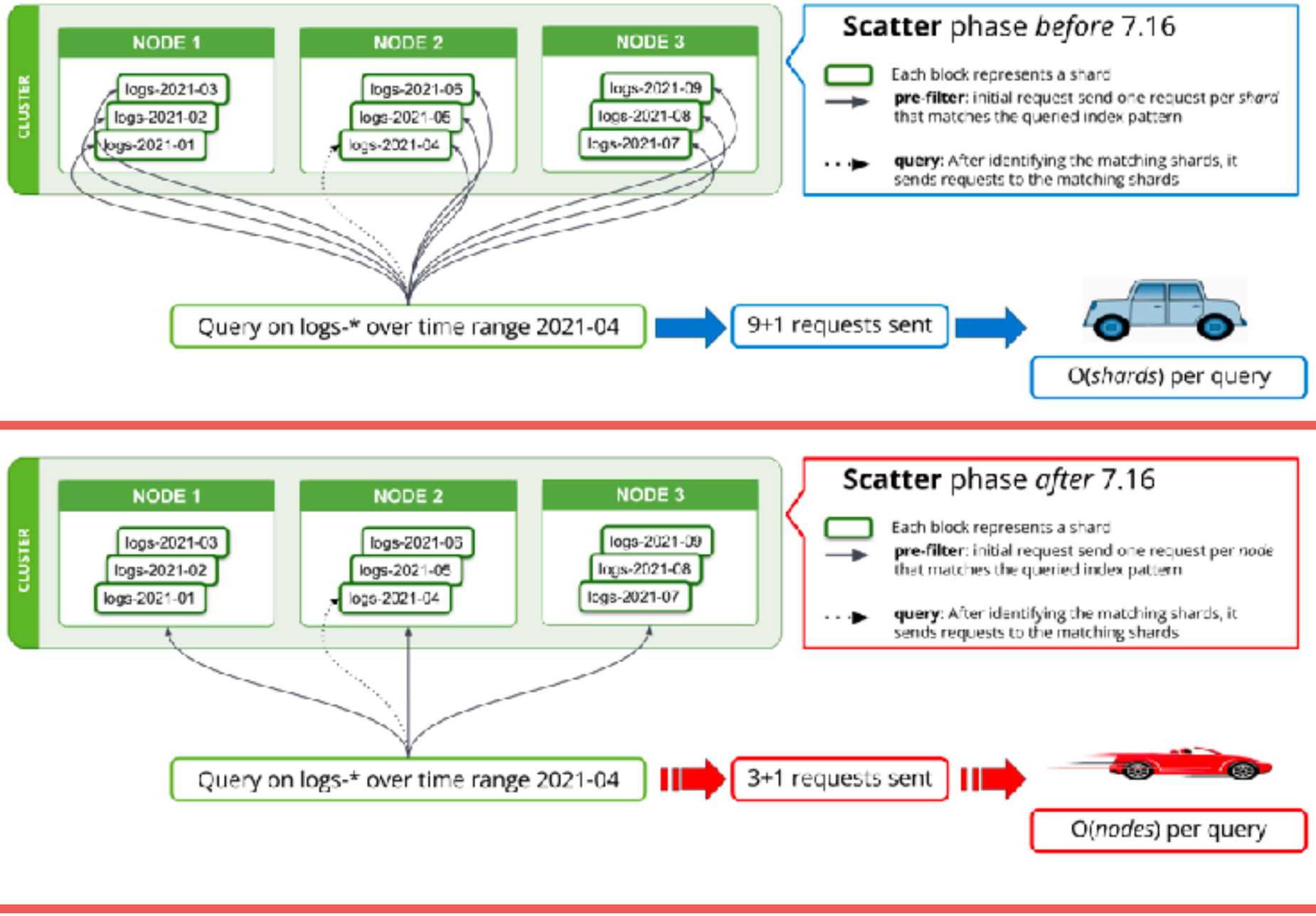
Data

Coordinate

Content, hot, warm
cold, frozen ..

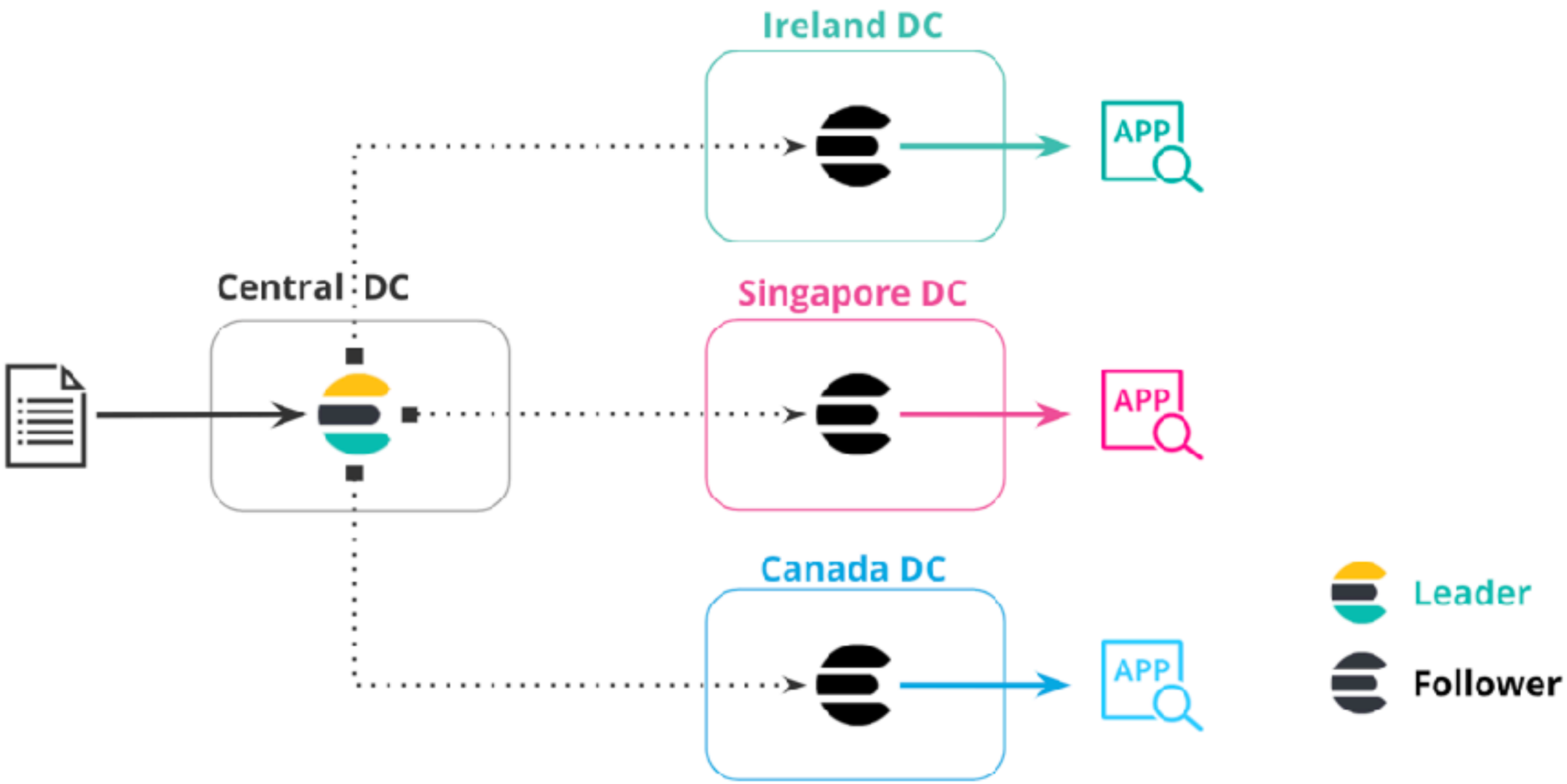
https://www.elastic.co/guide/en/elasticsearch/reference/current/modules-node.html

Search Improvement



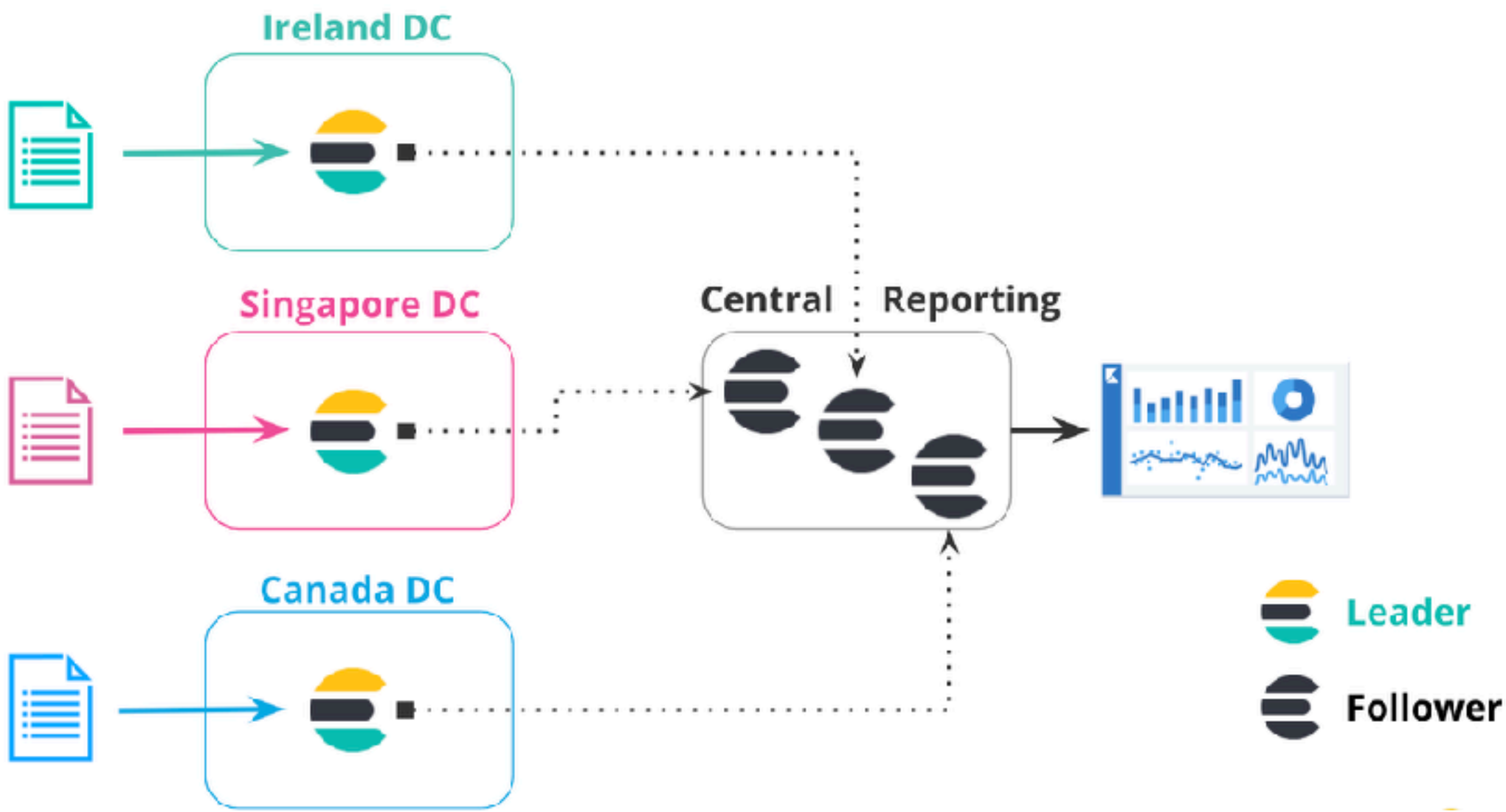
https://www.elastic.co/blog/three-ways-improved-elasticsearch-scalability

Cross Cluster Replication



https://www.elastic.co/guide/en/elasticsearch/reference/current/xpack-ccr.html

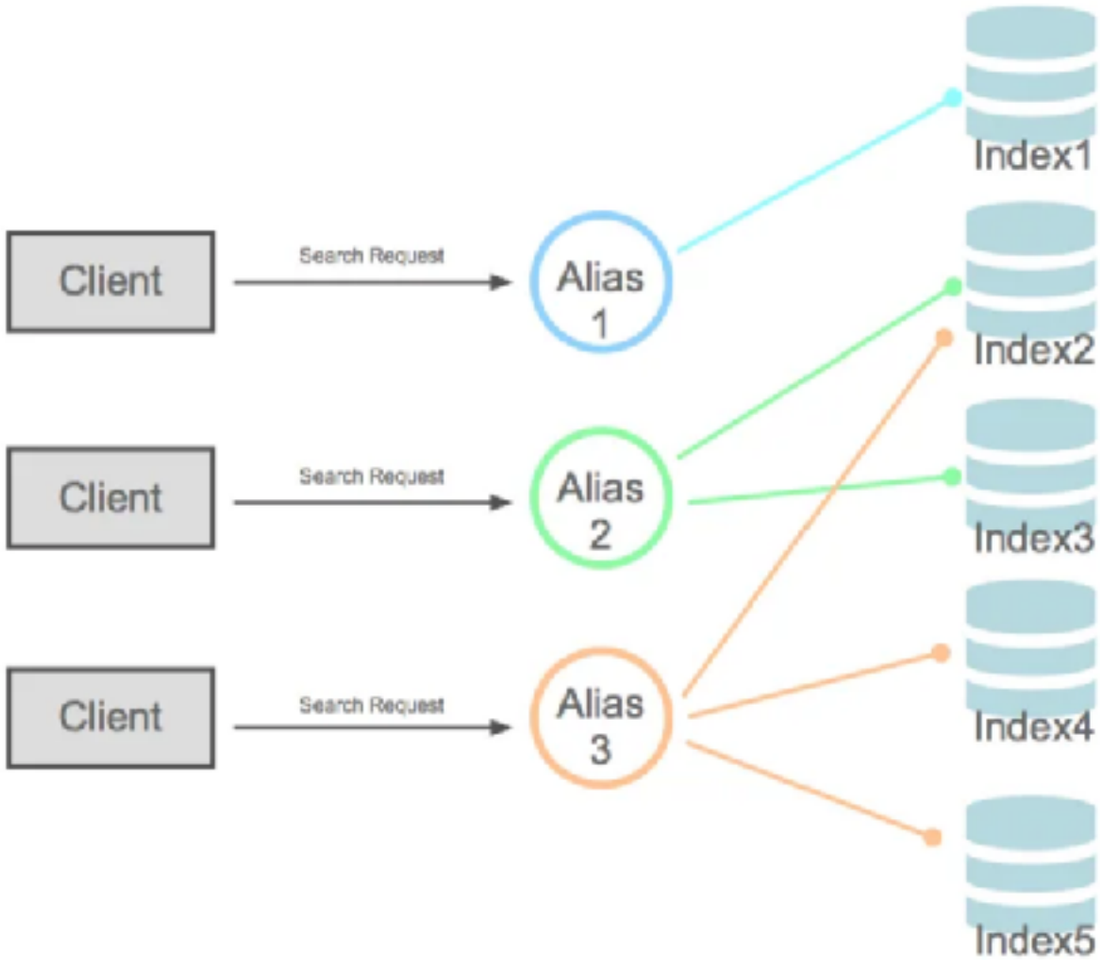
Cross Cluster Replication



https://www.elastic.co/guide/en/elasticsearch/reference/current/xpack-ccr.html

Indices (Database)

Alias index



https://www.elastic.co/guide/en/elasticsearch/reference/current/aliases.html

Delete data in index !!

Mark deleted only, not remove from disk !!

```
DELETE user/_doc/123

POST user/_delete_by_query
{
  "query": {
    "match": {
      "id": 123
    }
  }
}
```

Index Lifecycle Management

ILM Policy

Hot phase Required

Store your most recent, most frequently-searched data in the hot tier. The hot tier provides the best indexing and search performance by using the most powerful, expensive hardware.

> Advanced settings

Warm phase

Move data to the warm tier when you are still likely to search it, but infrequently need to update it. The warm tier is optimized for search performance over indexing performance.

Cold phase

Move data to the cold tier when you are searching it less often and don't need to update it. The cold tier is optimized for cost savings over search performance.

Delete phase Remove

Delete data you no longer need.

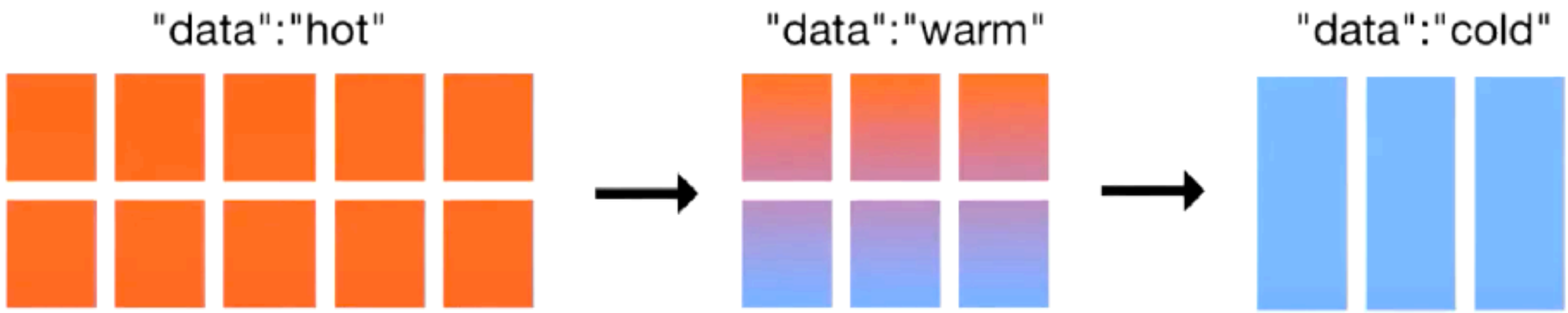
Wait for snapshot policy

Specify a snapshot policy to be executed before the deletion of the index. This ensures that a snapshot of the deleted index is available. Learn more

Policy name (optional)

Type and then hit "ENTER"

Index Lifecycle



Index Lifecycle

Phase	Description
Hot	Active with update and query
Warm	No longer to update, but still query
Cold	No longer to update and query infrequently
Frozen	No longer to update and query rarely
Delete	No longer to use

https://www.elastic.co/guide/en/elasticsearch/reference/current/ilm-index-lifecycle.html

Mapping

Schema of data

Dynamic Mapping !!

Try to indexing data

Learn from dynamic mapping

```
PUT user/_doc/123
{
  "id": "123",
  "name": “somkiat pui",
  "email": "xxx@xxx.com",
  "mobile": "0888888888"
}
```

Result from dynamic mapping

```
GET user/_mapping
{
  "user": {
    "mappings": {
      "properties": {
        "email": {
          "type": "text",
          "fields": {
            "keyword": {
              "type": "keyword",
              "ignore_above": 256
            }
          }
        }
```

Result from dynamic mapping

```
GET user/_mapping
{
  "user": {
    "mappings": {
      "properties": {
        "email": {
          "type": "text",
          "fields": {
            "keyword": {
              "type": "keyword",
              "ignore_above": 256
            }
          }
        }
      }
    }
```

What ?

Do you need it ?

Data Types

Text data type

Text

match_only_text

keyword

https://www.elastic.co/guide/en/elasticsearch/reference/current/text.html

Text vs Keyword

Hello Elasticsearch

Keyword Data Type		
Term	Count	Document
Hello, How to install ElasticSearch	1	Test
Hello, this is elastisearch	1	Test
Text Data Type*		
Term	Count	Document
Hello	2	Test
this	1	Test
is	1	Test
elastisearch	2	Test
How	1	Test
to	1	Test
install	1	Test
* the most common words are usually uninteresting, e.g. "the", "a", "in", "for" and so on.		

Text

For search data
Partial search

Keyword

Exactly query/search
Aggregation or group by

Text vs Match_only_text

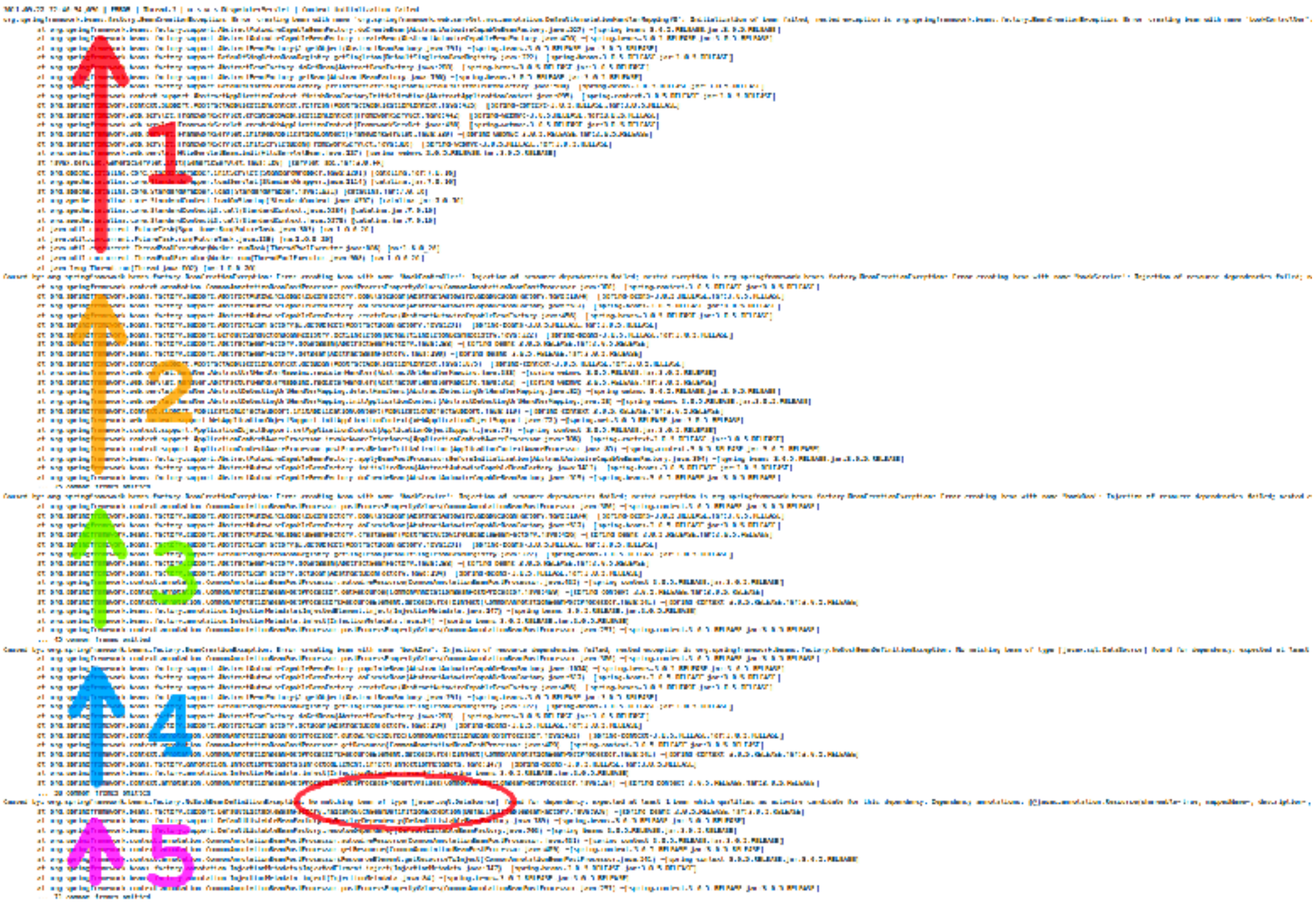
Text type family

The text family includes the following field types:

- `text`, the traditional field type for full-text content such as the body of an email or the description of a product.
- `match_only_text`, a space-optimized variant of `text` that disables scoring and performs slower on queries that need positions. It is best suited for indexing log messages.

Improve search performance for logging

Log message !!



Back to default mapping

```
GET user/_mapping
{
  "user": {
    "mappings": {
      "properties": {
        "email": {
          "type": "text",
          "fields": {
            "keyword": {
              "type": "keyword",
              "ignore_above": 256
            }
          }
        }
      }
```

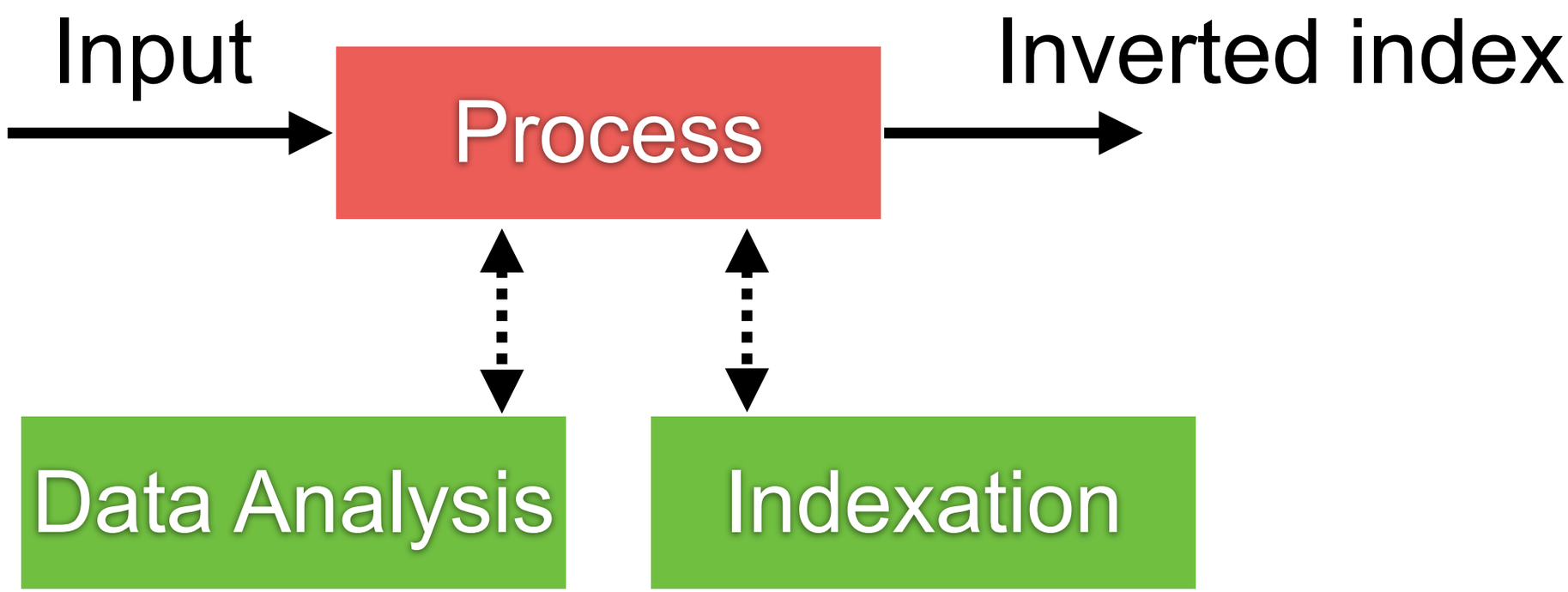
What ?

Do you need it ?

Analysis

Input data analysis

Write-once and read-many-times structure



Analyzer

Indexing or Re-indexing process

Consume expensive resources !!

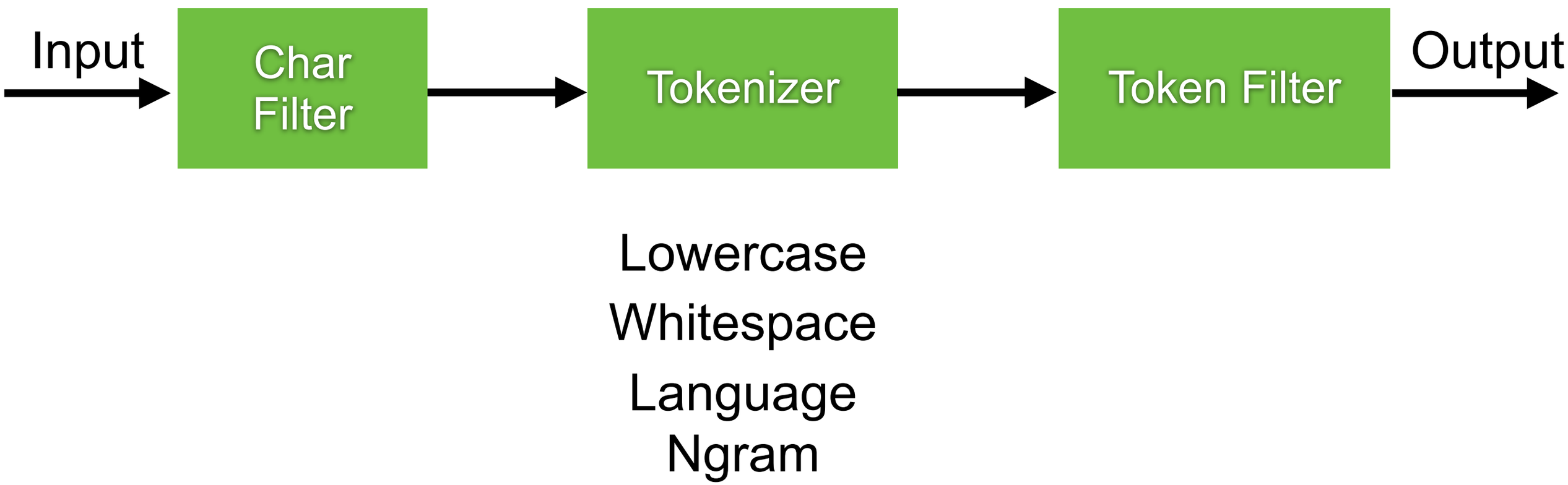
Analyzer

HTML strip

Mapping characters

Pattern replace

Analyzer



Analyzer

Lowercase

Remove stopword

Synonym

Example

Document no.	Data
1	Elasticsearch Server
2	Mastering Elasticsearch Second Edition
3	Apache Solr Cookbook Third Edition

Token

Token	Document no.
Elasticsearch	1
Elasticsearch	2
Server	1
Mastering	2
Second	2
Edition	2
Edition	3
Apache	3
Solr	3
Cookbook	3
Third	3

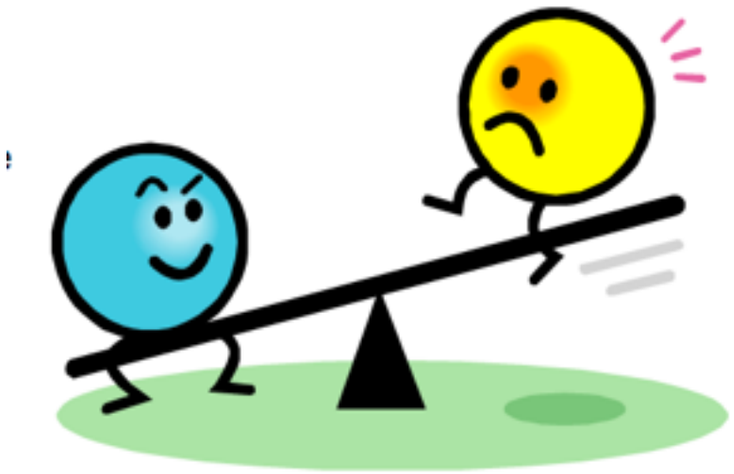
Term

Token	Count	Document no.
elasticsearch	2	1,2
server	1	1
mastering	1	2
second	1	2
edition	2	2,3
apache	1	3
solr	1	3
cookbook	1	3
third	1	3

Lucene inverted index

Token	Count	Document no.
elasticsearch	2	1,2
server	1	1
mastering	1	2
second	1	2
edition	2	2,3
apache	1	3
solr	1	3
cookbook	1	3
third	1	3

More token, more disk
Better token, search better



Build-in Analyzer ?

https://www.elastic.co/guide/en/elasticsearch/reference/current/analysis-analyzers.html

Default Analyzer ?

Built-in analyzer reference

Elasticsearch ships with a wide range of built-in analyzers, which can be used in any index without further configuration:

Standard Analyzer

The `standard` analyzer divides text into terms on word boundaries, as defined by the Unicode Text Segmentation algorithm. It removes most punctuation, lowercases terms, and supports removing stop words.

Simple Analyzer

The `simple` analyzer divides text into terms whenever it encounters a character which is not a letter. It lowercases all terms.

Standard Analyzer

Analyze

```
GET _analyze
{
  "analyzer": "standard",
  "text": "Hello, Elasticsearch..."
}
```

Result

```
{
  "tokens": [
    {
      "token": "hello",
      "start_offset": 0,
      "end_offset": 5,
      "type": "<ALPHANUM>",
      "position": 0
    },
    {
      "token": "elasticsearch",
      "start_offset": 7,
      "end_offset": 20,
      "type": "<ALPHANUM>",
      "position": 1
    }
  ]
}
```

Language Analyzer ?

Language analyzers

A set of analyzers aimed at analyzing specific language text. The following types are supported: `arabic`, `armenian`, `basque`, `bengali`, `brazilian`, `bulgarian`, `catalan`, `cjk`, `czech`, `danish`, `dutch`, `english`, `estonian`, `finnish`, `french`, `galician`, `german`, `greek`, `hindi`, `hungarian`, `indonesian`, `irish`, `italian`, `latvian`, `lithuanian`, `norwegian`, `persian`, `portuguese`, `romanian`, `russian`, `sorani`, `spanish`, `swedish`, `turkish`, `thai`.

https://www.elastic.co/guide/en/elasticsearch/reference/current/analysis-lang-analyzer.html

Thai Analyzer

Analyze

```
GET _analyze
{
  "analyzer": "thai",
  "text": "สวัสดีประเทศไทยนะจ๊ะ"
}
```

Result

```
{
  "tokens": [
    {
      "token": "สวัสดี",
      "start_offset": 0,
      "end_offset": 6,
      "type": "word",
      "position": 0
    },
    {
      "token": "ประเทศ",
      "start_offset": 6,
      "end_offset": 12,
      "type": "word",
      "position": 1
    },
    {
      "token": "ไทย",
      "start_offset": 12,
      "end_offset": 15,
      "type": "word",
      "position": 2
    },
…
}
```

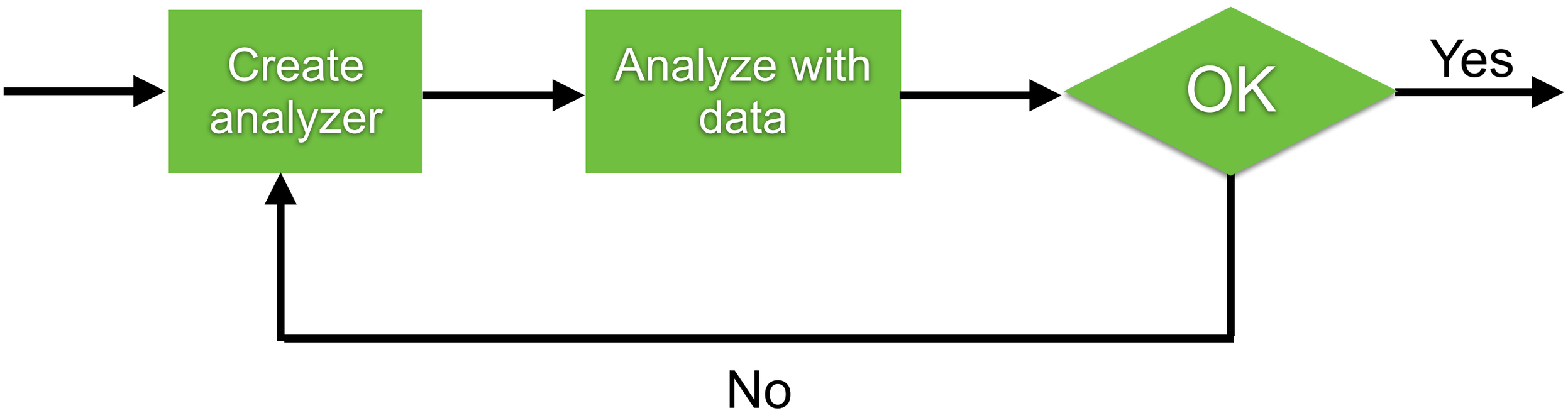
Customized Analyzer ?

Customized Analyzer ?

Application

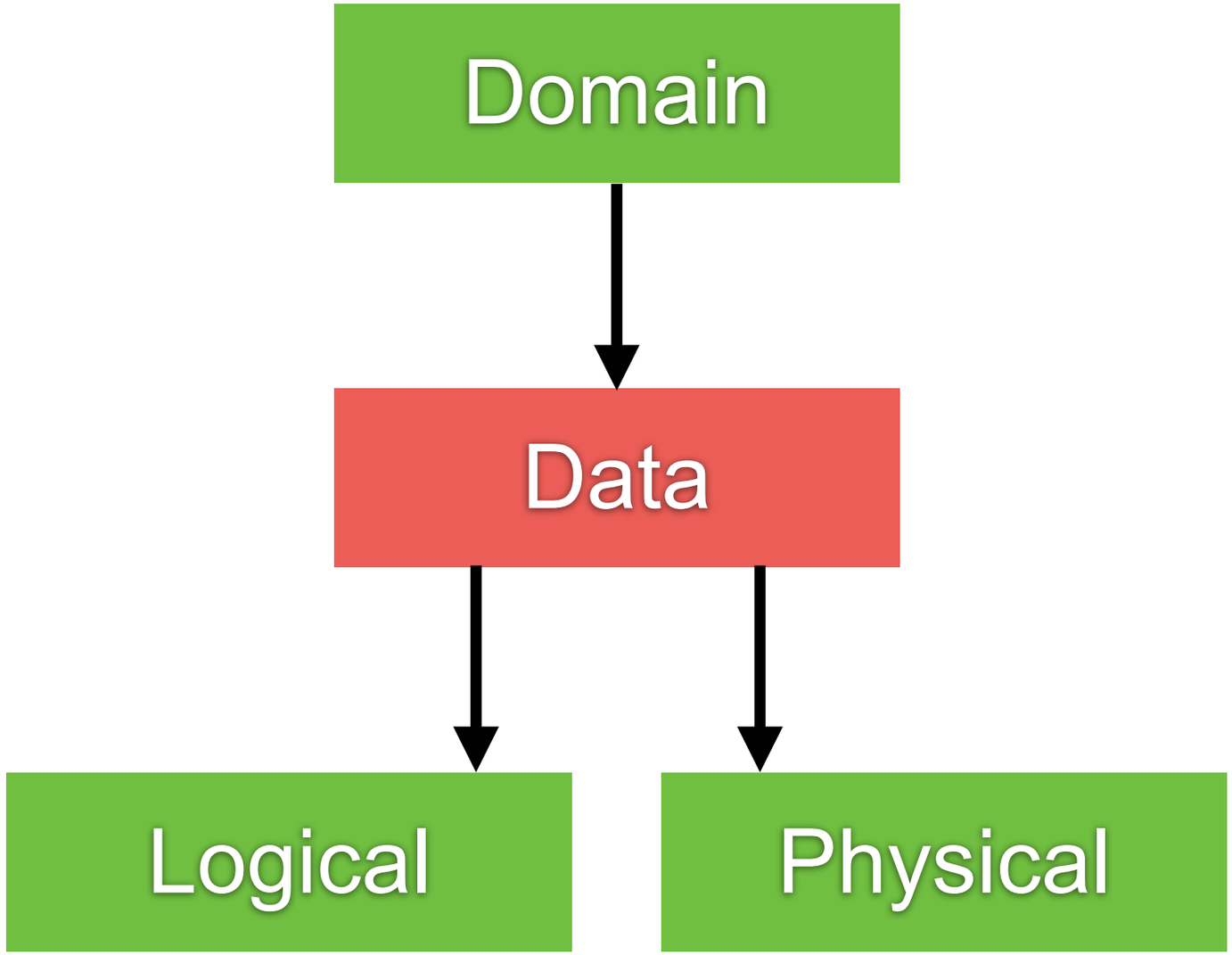
Elasticsearch

Alway, Test Analyzer



Data modeling in Elasticsearch

Modeling ?

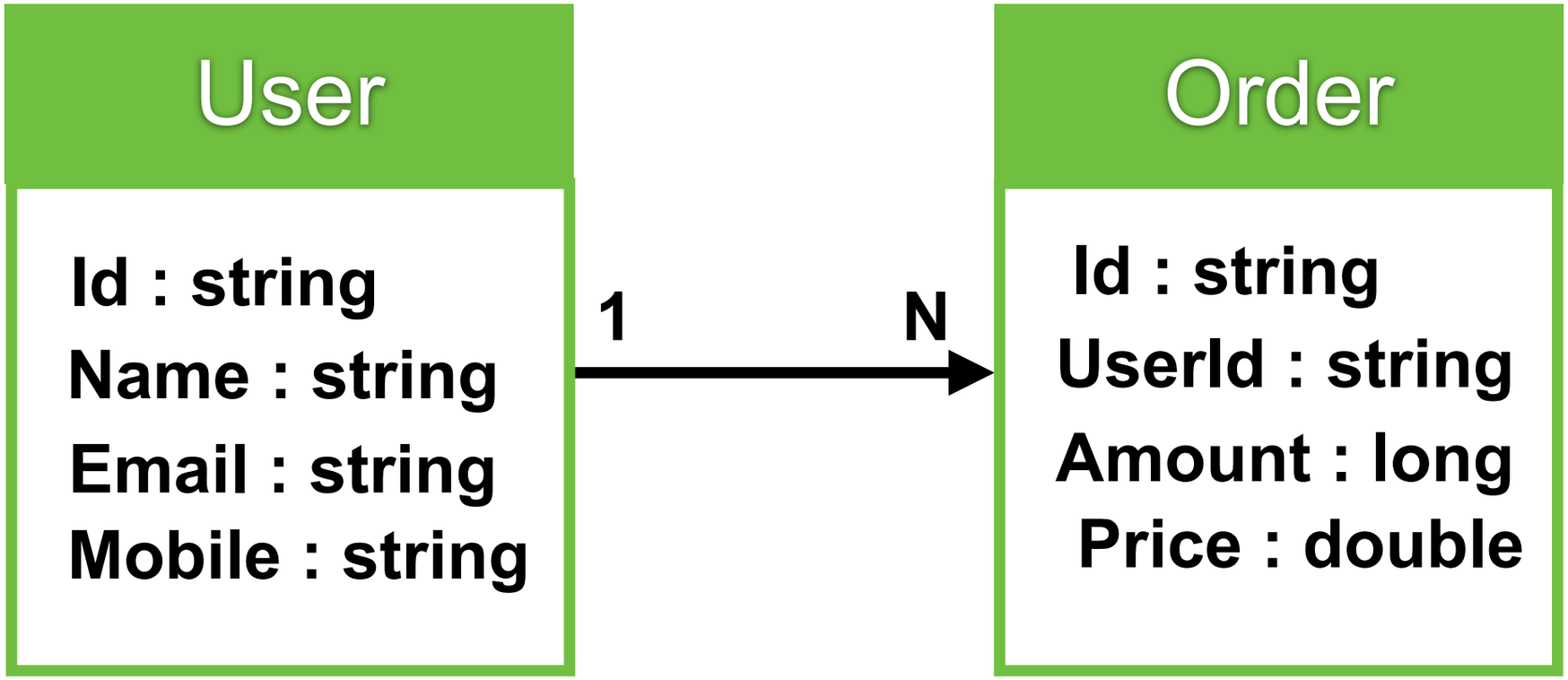


Goals ?

Performance
Scale
Near realtime search

Large Data

Example of Data



Data Modeling in ES

Application join
Denormalization
Parent-children
Nested object

Denormalization

Order

```
{
    id: string
    amount: long
    price: double
    user: {
        id: string
        name: string
        email: string
        mobile: string
    }
}
```

Parent-children

Join field type

edit

The `join` data type is a special field that creates parent/child relation within documents of the same index. The `relations` section defines a set of possible relations within the documents, each relation being a parent name and a child name.

WARNING

We don't recommend using multiple levels of relations to replicate a relational model. Each level of relation adds an overhead at query time in terms of memory and computation. For better search performance, denormalize your data instead.

A parent/child relation can be defined as follows:

https://www.elastic.co/guide/en/elasticsearch/reference/current/parent-join.html

Parent-children in ES

```
PUT order
{
  "mappings": {
    "properties": {
      "user_order_join_field": {
        "type": "join",
        "relations": {
          "order": "user"
        }
      }
    }
  }
}
```

Limitation

Parent and child documents must be indexed on the **same shard**

Indexing data

Order

```
POST order/_doc/1?routing=123
{
  "id": 1,
  "amount": 5,
  "price": 1000,
  "user_order_join_field":{
    "name": "order"
  }
}
```

User

```
PUT order/_doc/123?routing=123
{
  "id": 123,
  "name": "somkiat",
  "email": "xxx@xxx.com",
  "mobile": "0888888888",
  "user_order_join_field": {
    "name": "user",
    "parent": 1
  }
}
```

Search data

Order

```
GET order/_search
{
  "query": {
    "has_child": {
      "type": "user",
      "query": {
        "term": {
          "id": {
            "value": 123
          }
        }
      }
    }
  }
}
```

User

```
GET order/_search
{
  "query": {
    "has_parent": {
      "parent_type": "order",
      "query": {
        "term": {
          "id": {
            "value": 1
          }
        }
      }
    }
  }
}
```

Nested object

Use when you want to maintain the relationship of each object in an array

Nested field type

edit

The `nested` type is a specialised version of the `object` data type that allows arrays of objects to be indexed in a way that they can be queried independently of each other.

TIP

When ingesting key-value pairs with a large, arbitrary set of keys, you might consider modeling each key-value pair as its own nested document with `key` and `value` fields. Instead, consider using the flattened data type, which maps an entire object as a single field and allows for simple searches over its contents. Nested documents and queries are typically expensive, so using the `flattened` data type for this use case is a better option.

https://www.elastic.co/guide/en/elasticsearch/reference/current/nested.html

Nested object

Query all order for each user ?

Nested object

Query all order for each user ?

User

```
PUT user
{
  "mappings": {
    "properties": {
      "orders": {
        "type": "nested"
      }
    }
  }
}
```

Q/A