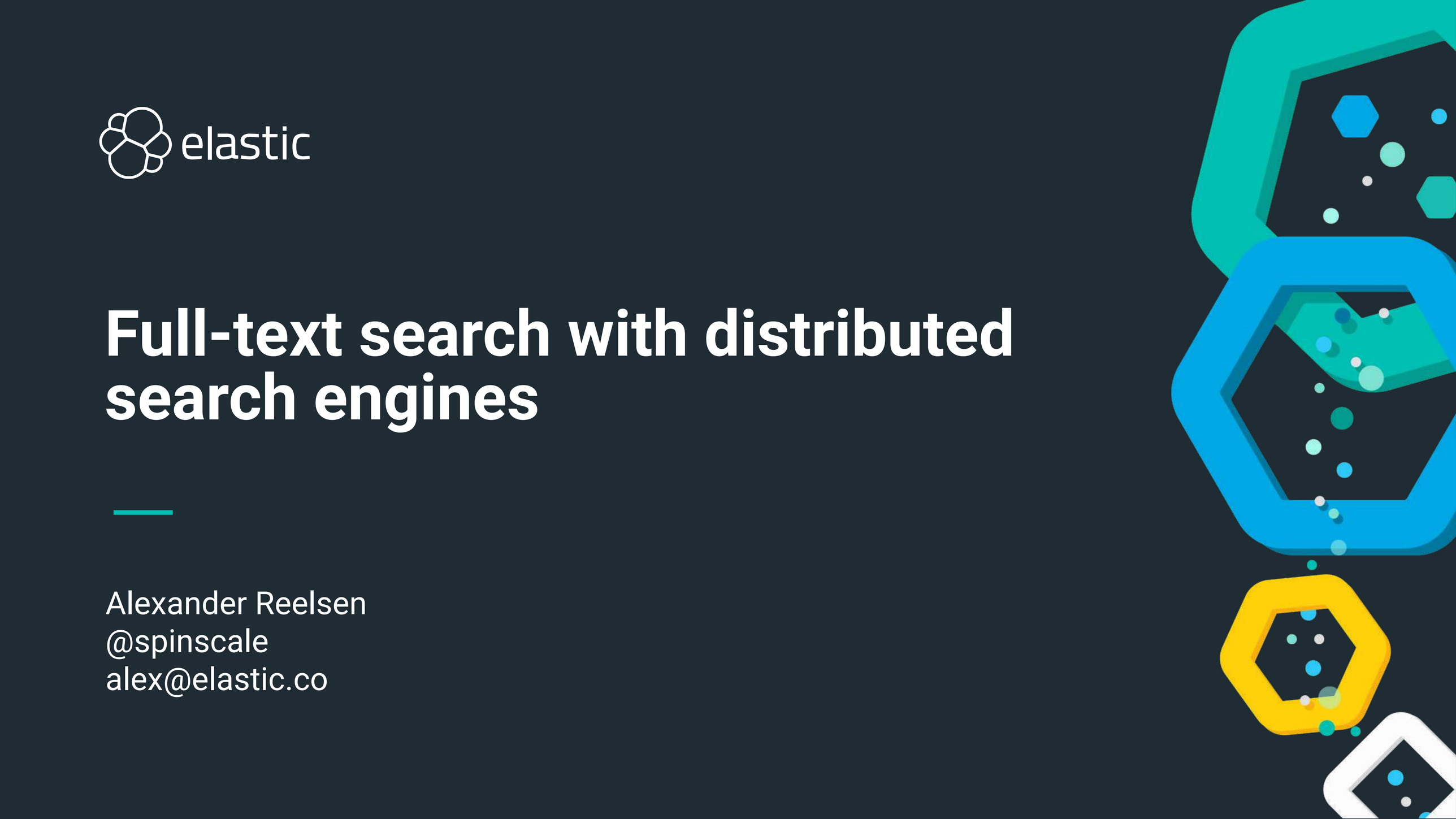


Full-text search with distributed search engines

Alexander Reelsen
@spinscale
alex@elastic.co

Agenda

- Overview
- Indexing: Analysis, Tokenization, Filtering, on disk data structures
- Searching: Scoring, Algorithms & Optimization
- Aggregations
- Distributed systems and search
- Q & A

elastic

Overview

Full text search introduction

```
SELECT * FROM products
         WHERE name LIKE = '%topf%'
```

```
# grep "topf" my_dataset.txt
```

Problem

- Scales linearly with the data set size
- Relevancy
- Spell correction
- Synonyms
- Phrases

Inverted Index

elastic

The quick brown fox jumped over the lazy dog

Tokenize

The	1
quick	1
brown	1
fox	1
jumped	1
over	1
the	1
lazy	1
dog	1

elastic

Sort

The	1
brown	1
dog	1
fox	1
jumped	1
lazy	1
over	1
quick	1
the	1

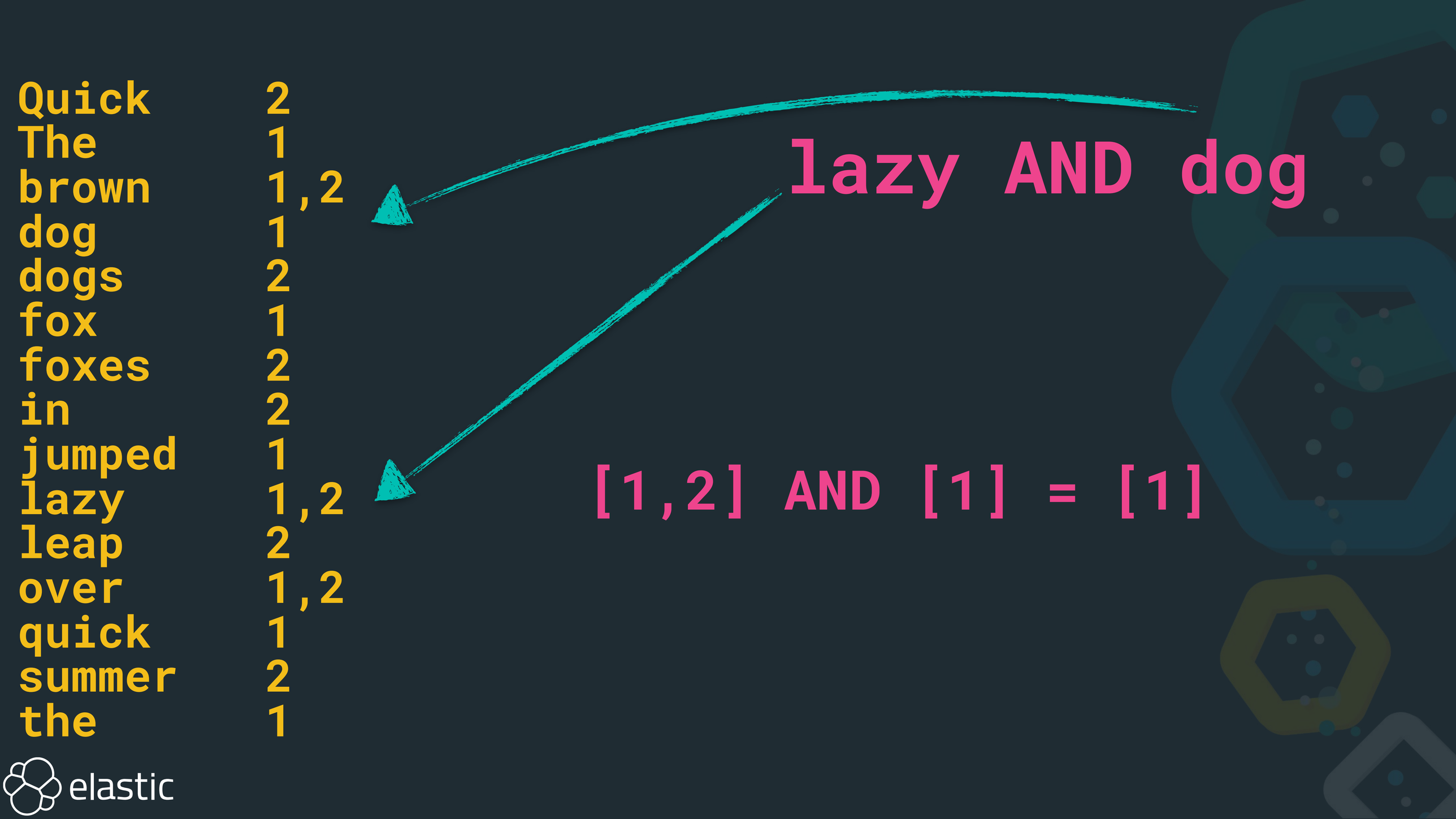
elastic

Quick brown foxes leap over lazy dogs in summer

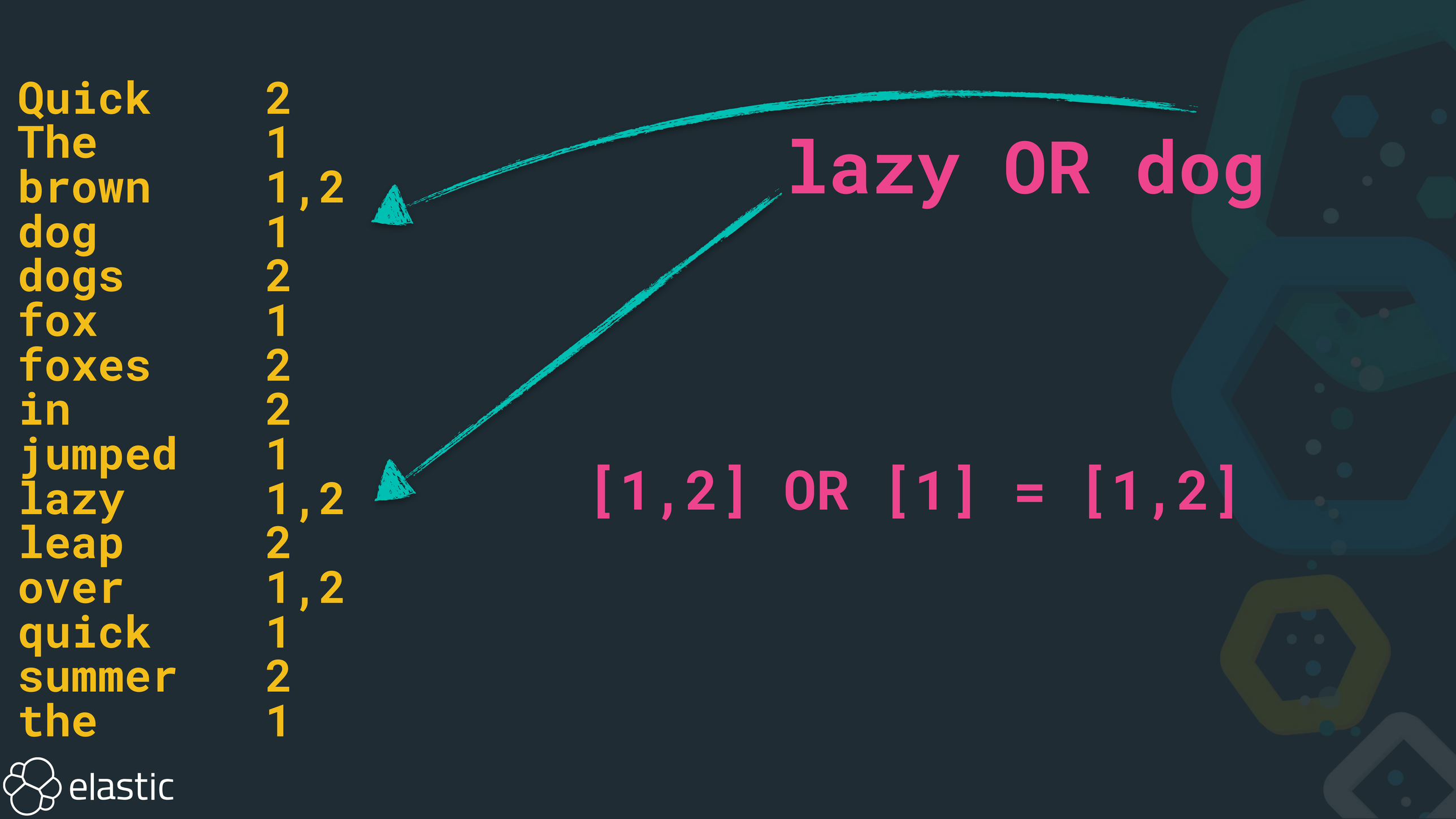
elastic

Quick	2
The	1
brown	1,2
dog	1
dogs	2
fox	1
foxes	2
in	2
jumped	1
lazy	1,2
leap	2
over	1,2
quick	1
summer	2
the	1

Quick	2
The	1
brown	1,2
dog	1
dogs	2
fox	1
foxes	2
in	2
jumped	1
lazy	1,2
leap	2
over	1,2
quick	1
summer	2
the	1


Quick 2
The 1
brown 1,2
dog 1
dogs 2
fox 1
foxes 2
in 2
jumped 1
lazy 1,2
leap 2
over 1,2
quick 1
summer 2
the 1
lazy AND dog
[1,2] AND [1] = [1]

elastic


Quick 2
The 1
brown 1,2
dog 1
dogs 2
fox 1
foxes 2
in 2
jumped 1
lazy 1,2
leap 2
over 1,2
quick 1
summer 2
the 1
lazy OR dog
[1,2] OR [1] = [1,2]

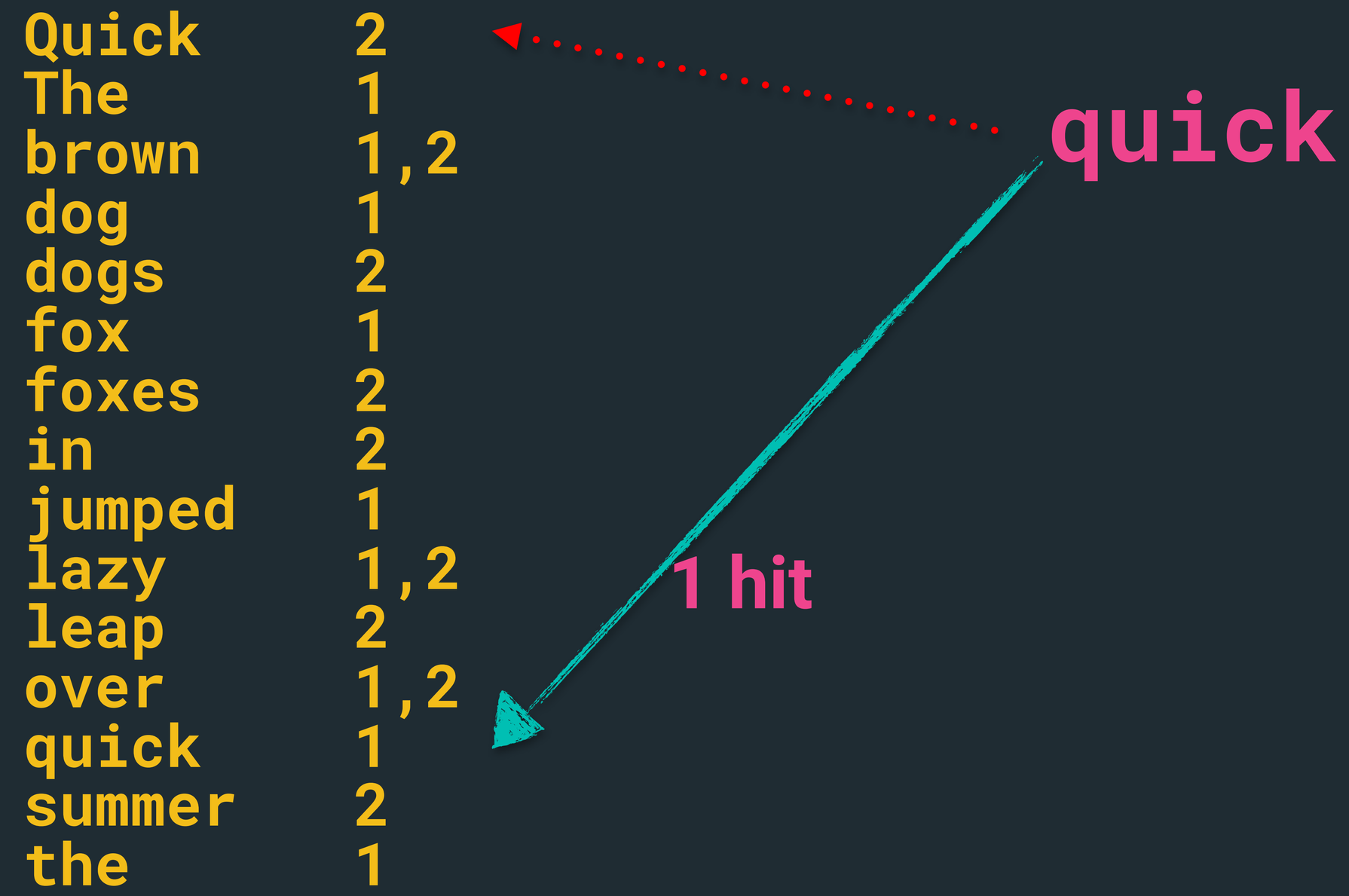
Technologies used today

- Apache Lucene (search library)
- Elasticsearch (distributed search engine built on top of Apache Lucene)

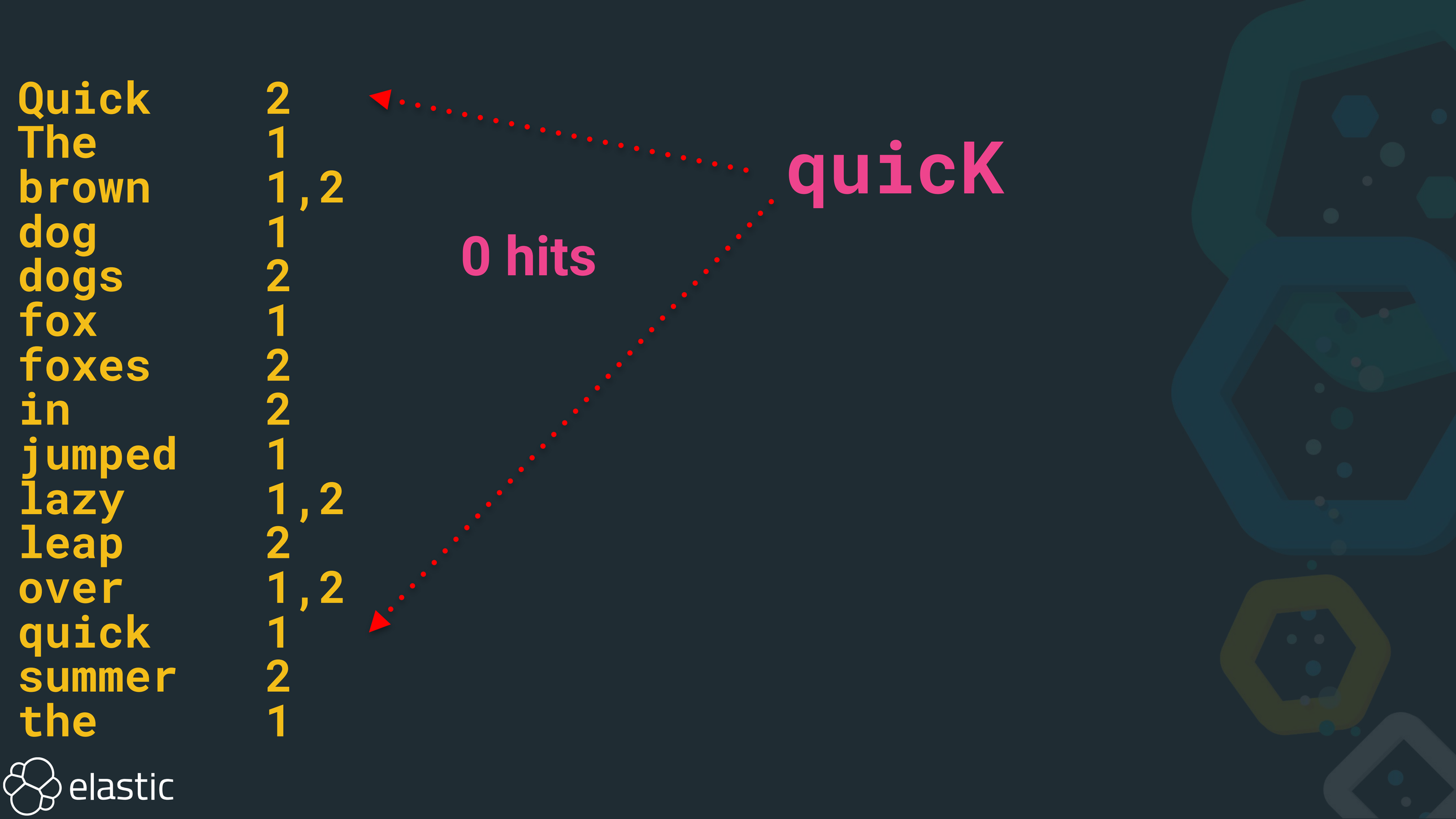
elastic

Indexing

Analysis, Tokenization, Filtering
Data structures


Quick 2
The 1
brown 1,2
dog 1
dogs 2
fox 1
foxes 2
in 2
jumped 1
lazy 1,2
leap 2
over 1,2
quick 1
summer 2
the 1
quick
1 hit

elastic

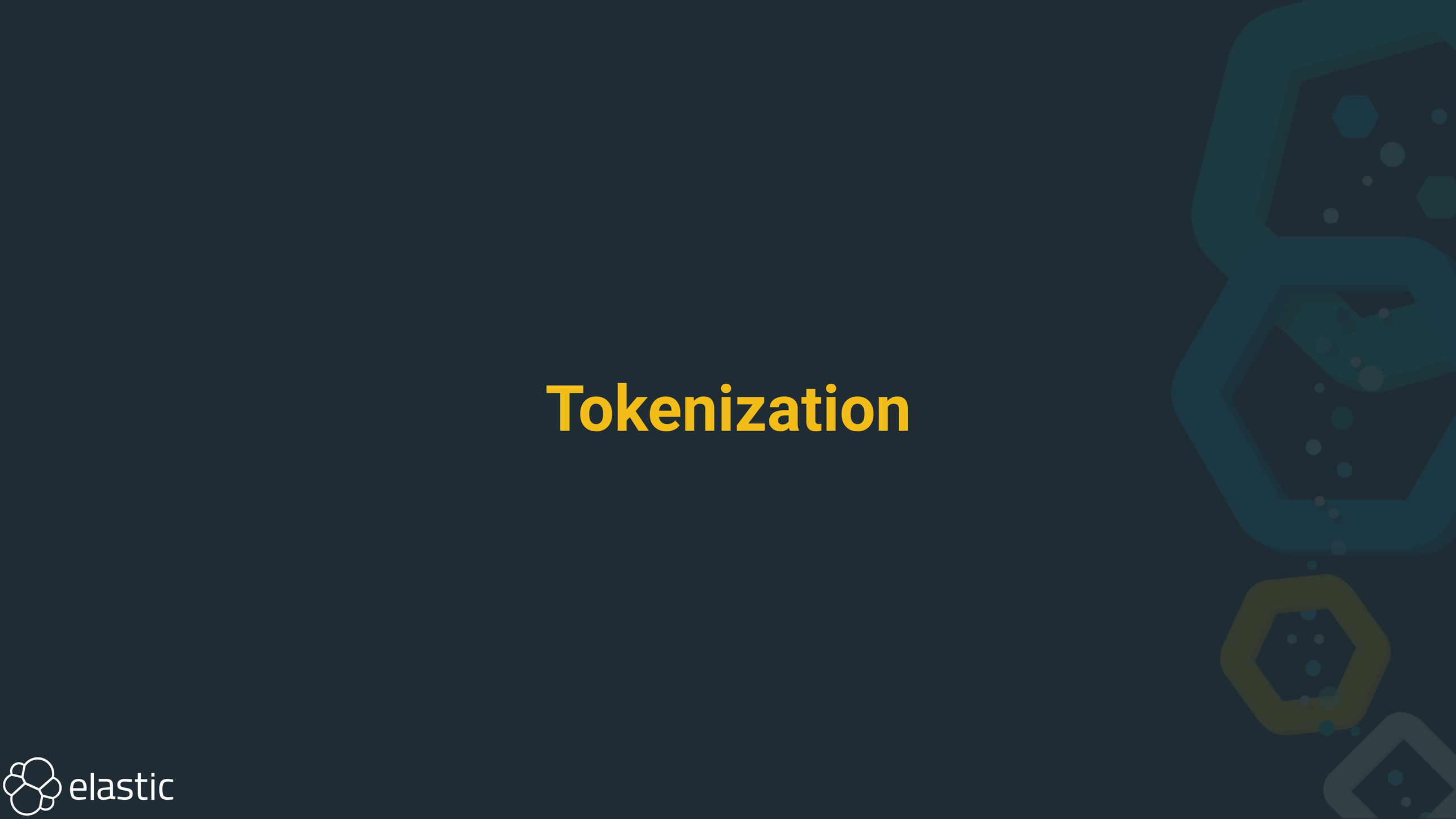

Quick 2
The 1
brown 1,2
dog 1
dogs 2
fox 1
foxes 2
in 2
jumped 1
lazy 1,2
leap 2
over 1,2
quick 1
summer 2
the 1
quicK
0 hits

elastic

Analysis: Tokenizer & Token Filters

elastic

Tokenization



elastic

Tokenization

```
quick brown fox
```

elastic

Tokenization

```
quick_brown_fox
```

Tokenization

```
quick_brown_fox
the lazy, white dog.
```

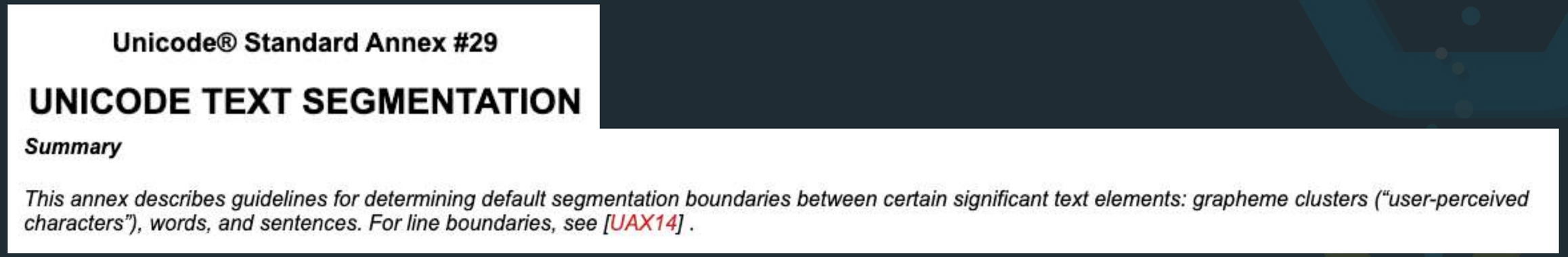
elastic

Tokenization

```
quick_brown_fox
the_lazy_white_dog
```

Tokenization

```
quick_brown_fox
the_lazy_white_dog
```



Unicode® Standard Annex #29

UNICODE TEXT SEGMENTATION

Summary

This annex describes guidelines for determining default segmentation boundaries between certain significant text elements: grapheme clusters ("user-perceived characters"), words, and sentences. For line boundaries, see [UAX14] .

https://unicode.org/reports/tr29/

elastic

Tokenization

```
quick_brown_fox
the_lazy_white_dog
https://www.jade-hs.de
```

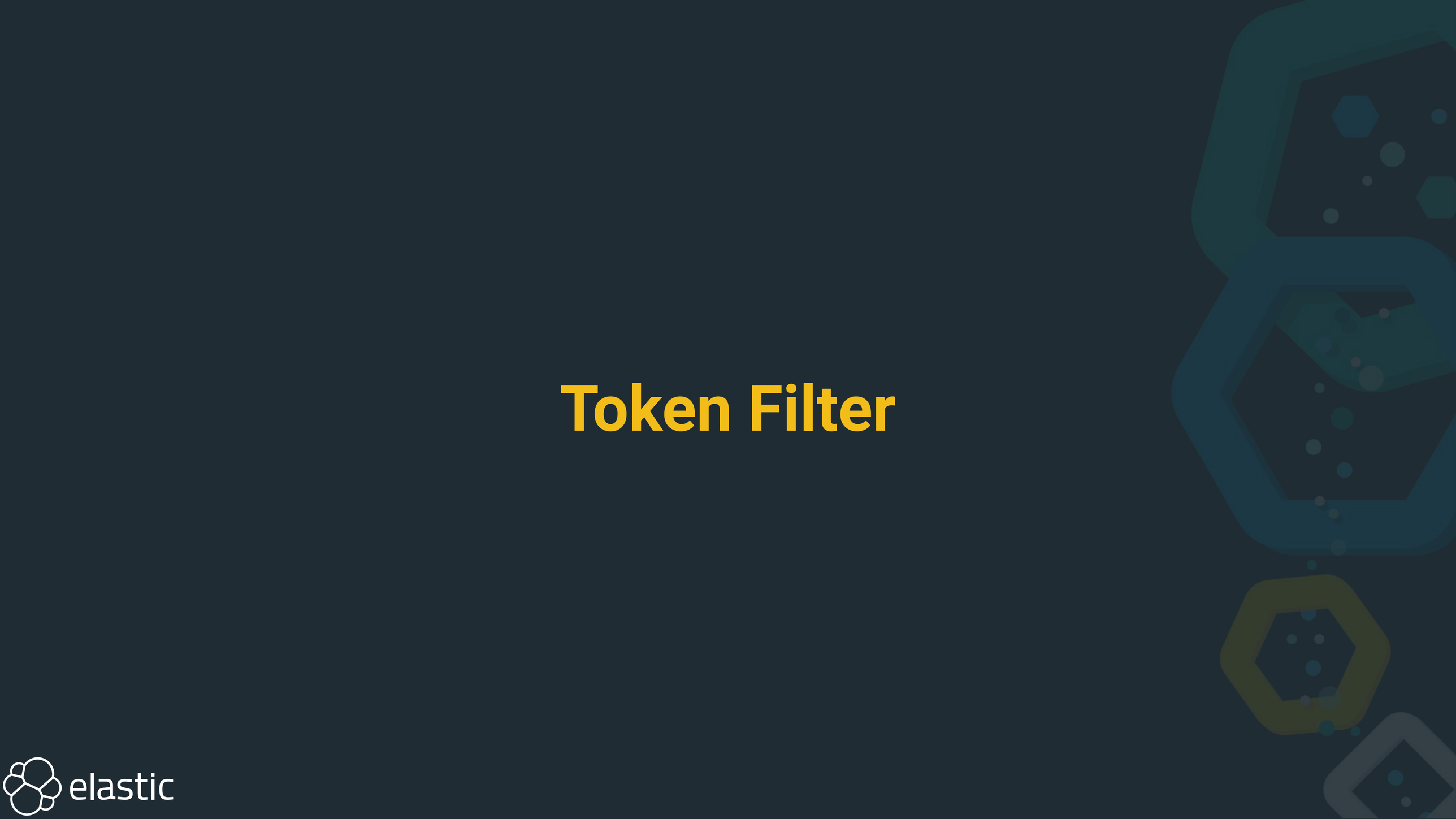
elastic

Tokenization

```
quick_brown_fox
the_lazy_white_dog
https_www.jade_hs.de
```

elastic

Token Filter



Quick	2
The	1
brown	1,2
dog	1
dogs	2
fox	1
foxes	2
in	2
jumped	1
lazy	1,2
leap	2
over	1,2
quick	1
summer	2
the	1

Token filter

Token filter

Lowercase

elastic

Token filter

	Lowercase	Stopwords
The	the	
Quick	quick	quick
brown	brown	brown
fox	fox	fox

Token filter

	Lowercase	Stopwords	Synonyms
The Quick brown fox	the quick brown fox	quick brown fox	quick,fast brown fox

Token filter

	Lowercase	Stopwords	Synonyms
The Quick brown fox	the quick brown fox	quick brown fox	quick,fast brown fox

Tokens can be changed, added, removed

elastic

Token filter

	Lowercase	Stopwords	Synonyms
The Quick brown fox	the quick brown fox	 quick brown fox	 quick,fast brown fox

Queries need to be processed as well!

elastic

More analysis strategies

- Phonetic analysis: `Meyer vs. Meier`
- Stemming: `foxes → fox`
- Compounding: `Blumentopf → blumen topf`
- Folding: `Spaß → Spass`

(On-Disk) Data structures

elastic

What else is in an inverted index?

- Documents: Find documents
- Term frequencies: Relevancy
- Positions: Positional Queries
- Offsets: Highlighting
- Stored fields: The original data

Segment: Unit of work

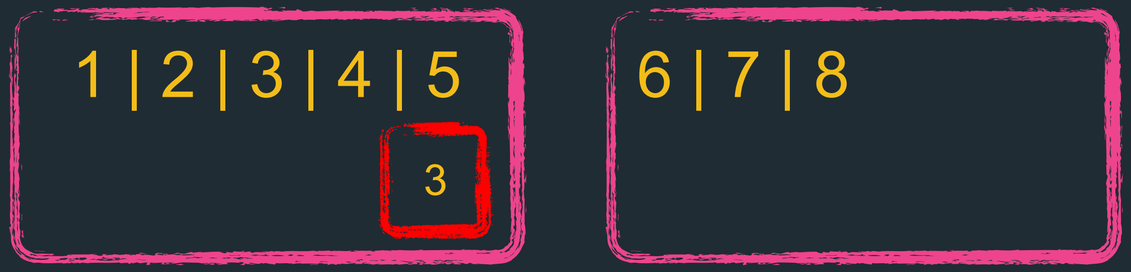
- A fully self sufficient inverted index
- An index consists of a number of segments
- New segments are created for newly added documents
- Segments are immutable!

Read-only data structures

- Pro: Write-once, sequentially
- Pro: Lock-free reading
- Pro: File system cache
- Contra: in-place updates & deletes
- Contra: Housekeeping
- Contra: Transactions

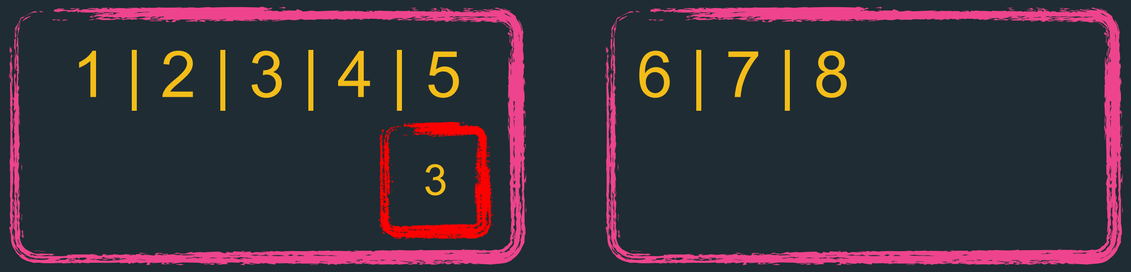
Segment: Deletes

- Mark a document as deleted in a special file
- Exclude it from searches
- No space is freed!



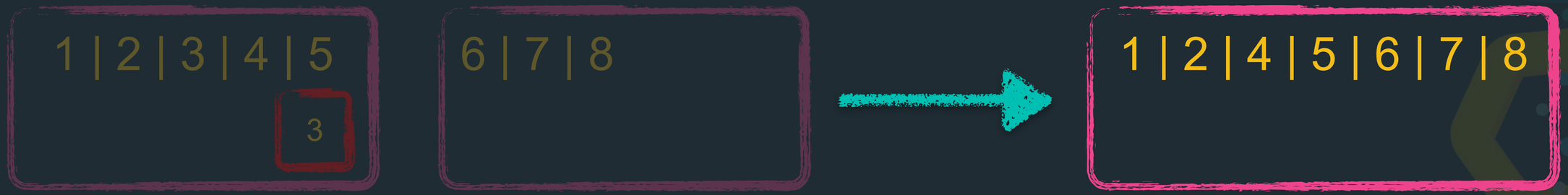
Segment: Merging

- Number of segments needs to be kept reasonable
- Merge multiple segments into one (smaller index)
- Delete expired documents



Segment: Merging

- Number of segments needs to be kept reasonable
- Merge multiple segments into one (smaller index)
- Delete expired documents

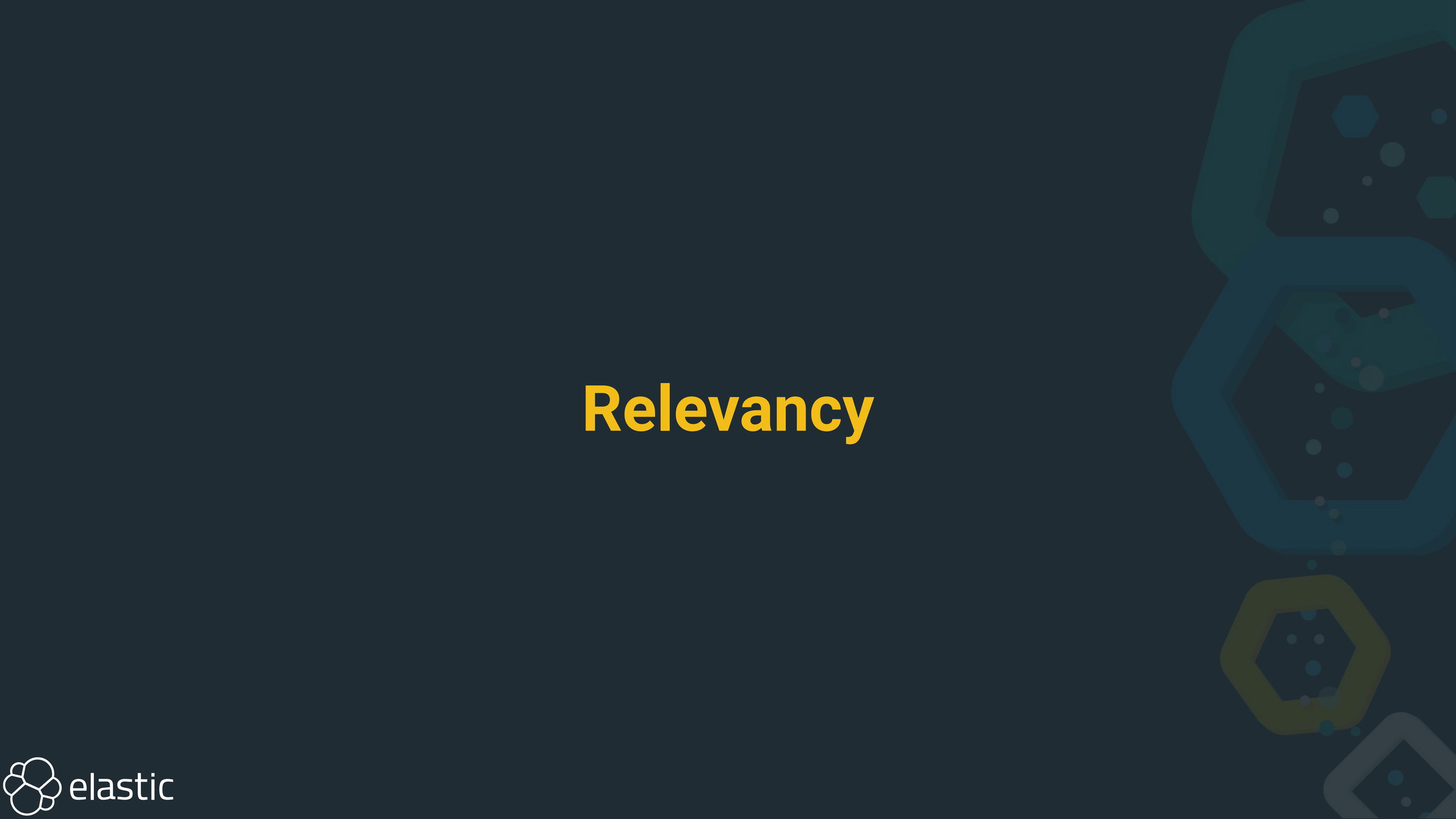


elastic

Searching

Precision vs. recall
Scoring
Algorithms and optimizations

Relevancy

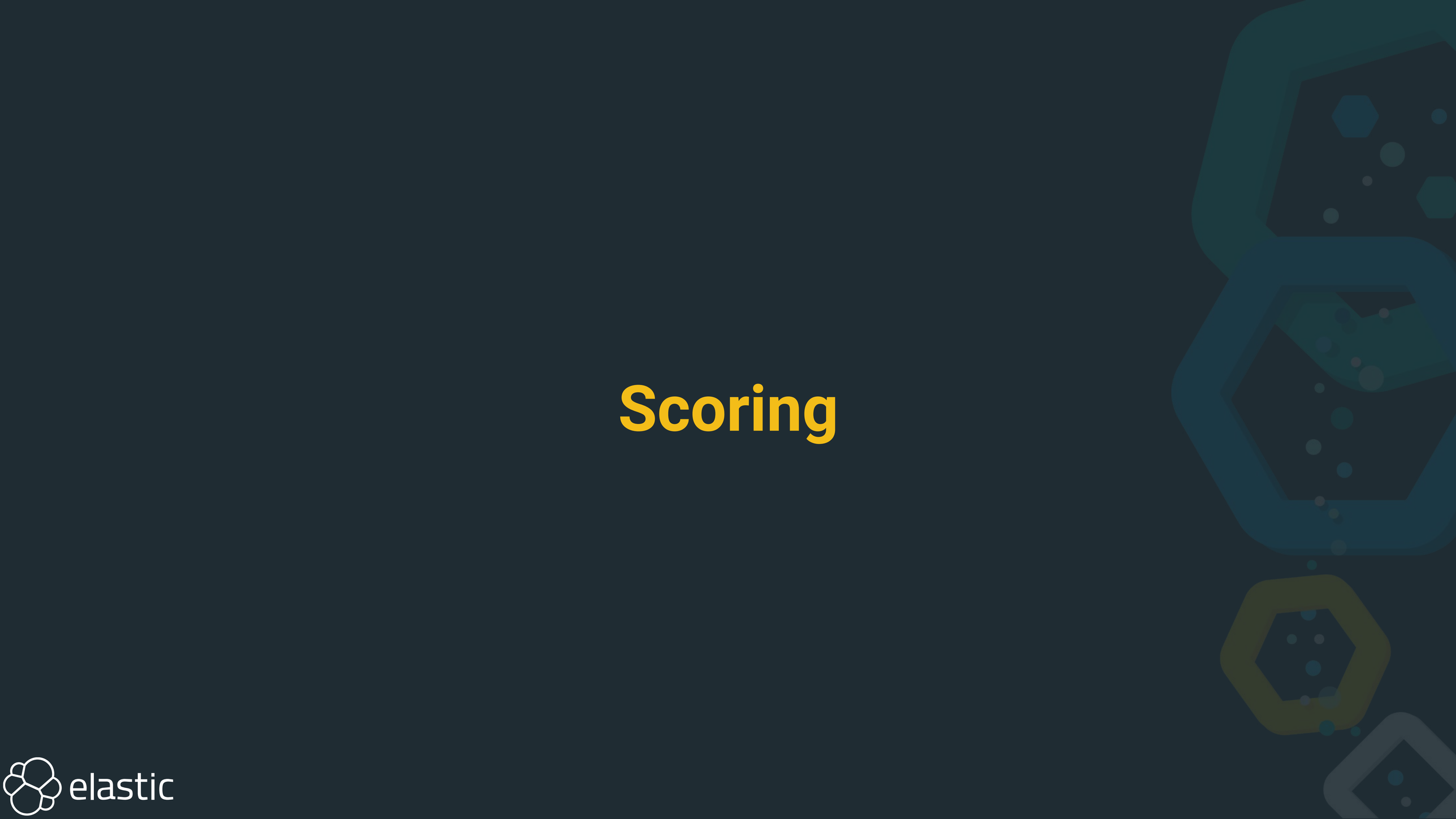


elastic

Relevancy

- Textbook answer: How well matches a document a query?
- Business answer: Are the top search results those that make me the most money?
  - marketplace
  - hotel booking website
  - newspaper website

Scoring



elastic

Scoring: `lazy dog`

- Naive: increase a counter if a term is matched
- "the lazy dog" => score 2
- "the lazy frog" => score 1
- "the lazy lazy lazy lazy cat" => score 4 or 1?

Scoring: More than term frequency

- How about incorporating information about the whole document corpus in scoring?
- Are lesser common terms more relevant?
- news paper: `"dieselgate news"`

Scoring: TF-IDF

- Term frequency: number of times a term occurs in a field
- Inverse document frequency: inverse function of the number of documents in which it occurs

Scoring: Vector space model

- Each term is a dimension
- The length is based on tf-idf calculation
- Similarity is the angle between vectors
- Cosine similarity: best match == angle 0°

elastic

Scoring: TF-IDF in Lucene

$$score(q,d) = \sum (tf(t \text{ in } d) \cdot idf(t)^2 \cdot t.getBoost() \cdot norm(t,d))$$

elastic

Lucene's Practical Scoring Function is derived from the above. The color codes demonstrate how it relates to those of the *conceptual* formula:

$$\text{score}(q,d) = \sum_{t \text{ in } q} \left(\text{tf}(t \text{ in } d) \cdot \text{idf}(t)^2 \cdot t.\text{getBoost}() \cdot \text{norm}(t,d) \right)$$

Lucene Practical Scoring Function

where

1. ***tf(t in d)*** correlates to the term's *frequency*, defined as the number of times term *t* appears in the currently scored document *d*. Documents that have more occurrences of a given term receive a higher score. Note that *tf(t in q)* is assumed to be *1* and therefore it does not appear in this equation, However if a query contains twice the same term, there will be two term-queries with that same term and hence the computation would still be correct (although not very efficient). The default computation for *tf(t in d)* in `ClassicSimilarity` is:

$$\texttt{tf(t in d)} = \text{frequency}^{1/2}$$

2. ***idf(t)*** stands for Inverse Document Frequency. This value correlates to the inverse of *docFreq* (the number of documents in which the term *t* appears). This means rarer terms give higher contribution to the total score. *idf(t)* appears for *t* in both the query and the document, hence it is squared in the equation. The default computation for *idf(t)* in `ClassicSimilarity` is:

$$\texttt{idf(t)} = 1 + \log\left(\frac{\text{docCount}+1}{\text{docFreq}+1} \right)$$

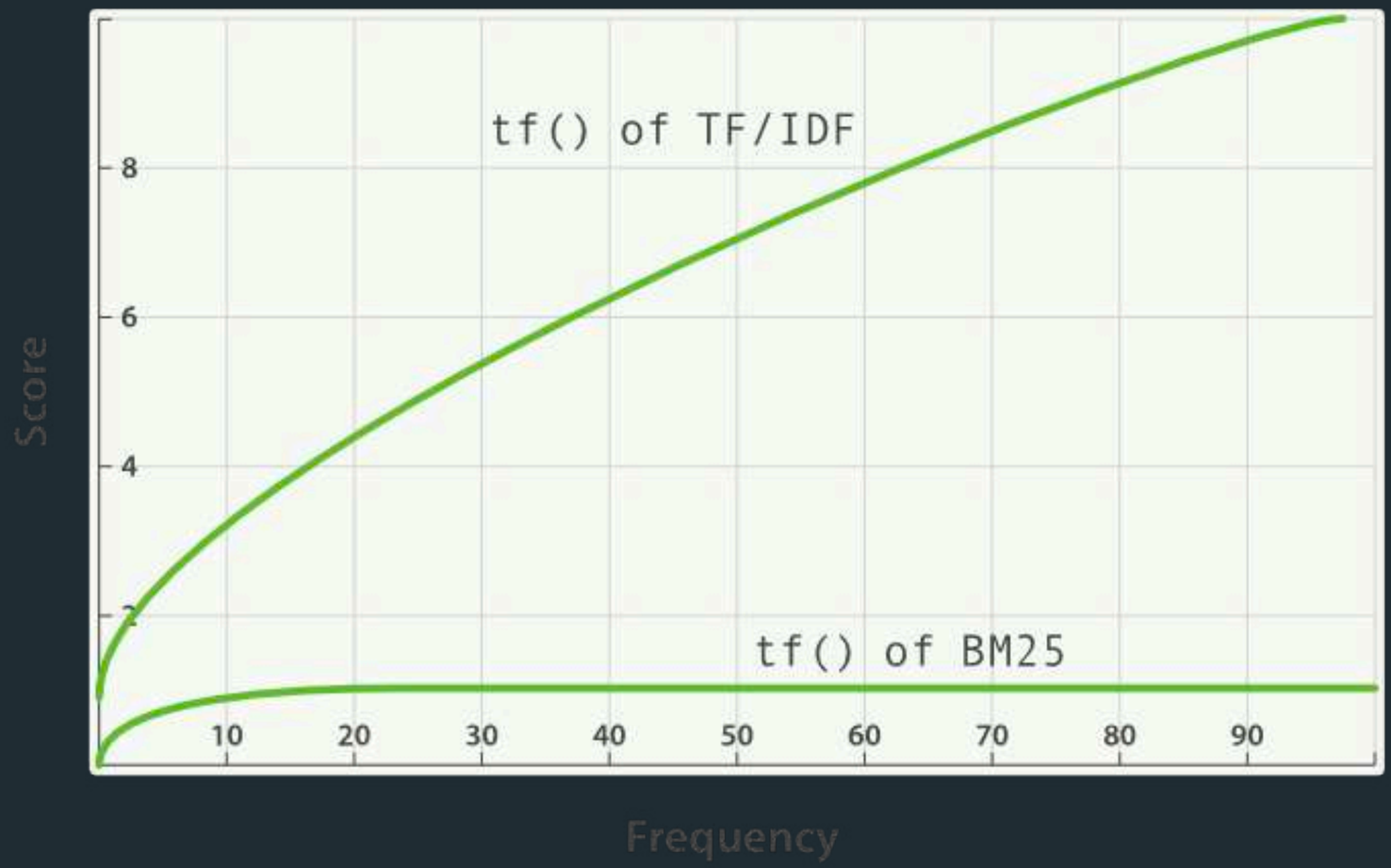
3. ***t.getBoost()*** is a search time boost of term *t* in the query *q* as specified in the query text (see query syntax), or as set by wrapping with `BoostQuery`. Notice that there is really no direct API for accessing a boost of one term in a multi term query, but rather multi terms are represented in a query as multi `TermQuery` objects, and so the boost of a term in the query is accessible by calling the sub-query `getBoost()`.

4. ***norm(t,d)*** is an index-time boost factor that solely depends on the number of tokens of this field in the document, so that shorter fields contribute more to the score.

BM25

- Default in Apache Lucene/Elasticsearch
- Works better with stopwords (high TF)
- Term frequency saturation
- Improved field length normalization (per document)

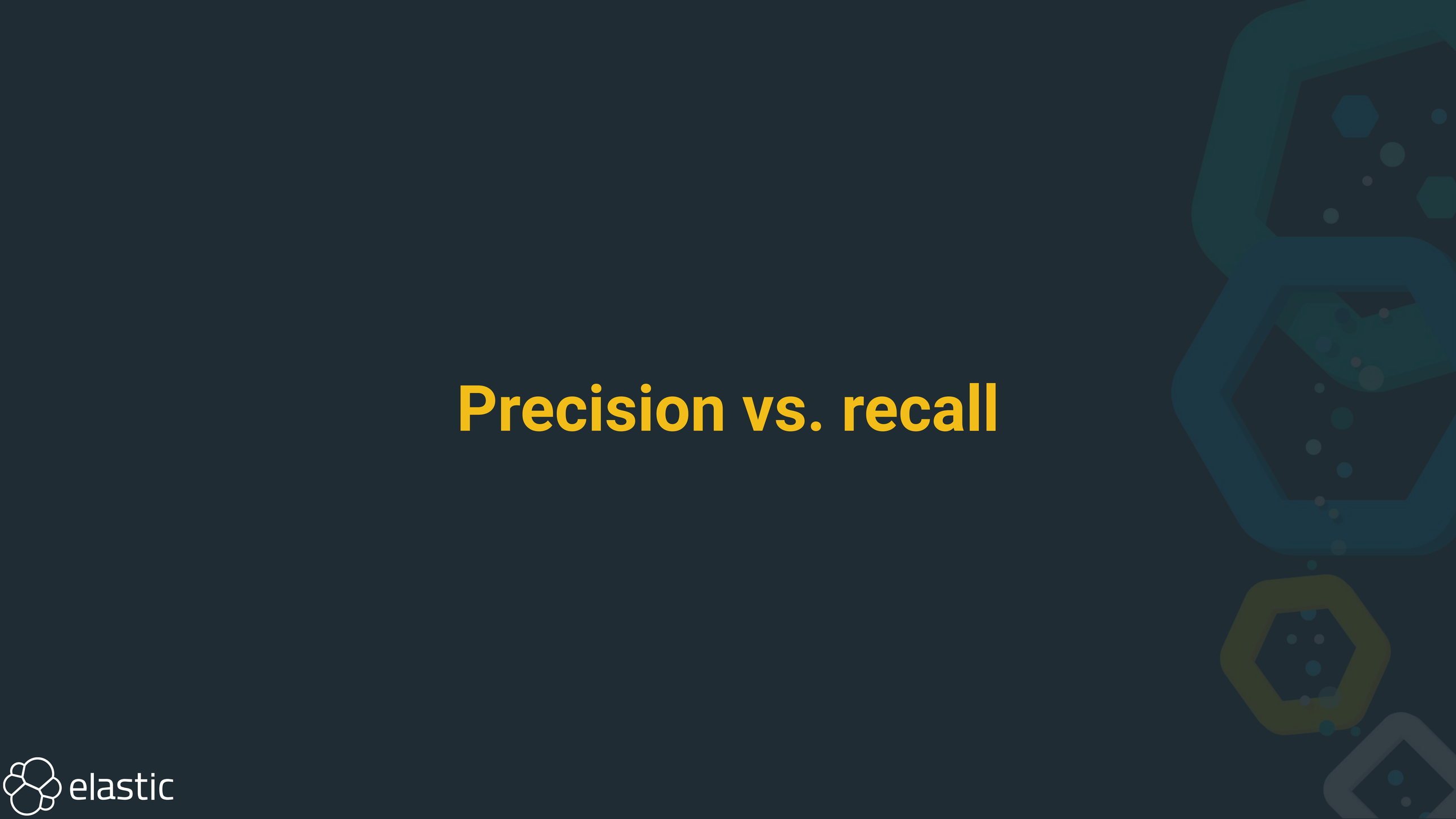
BM25



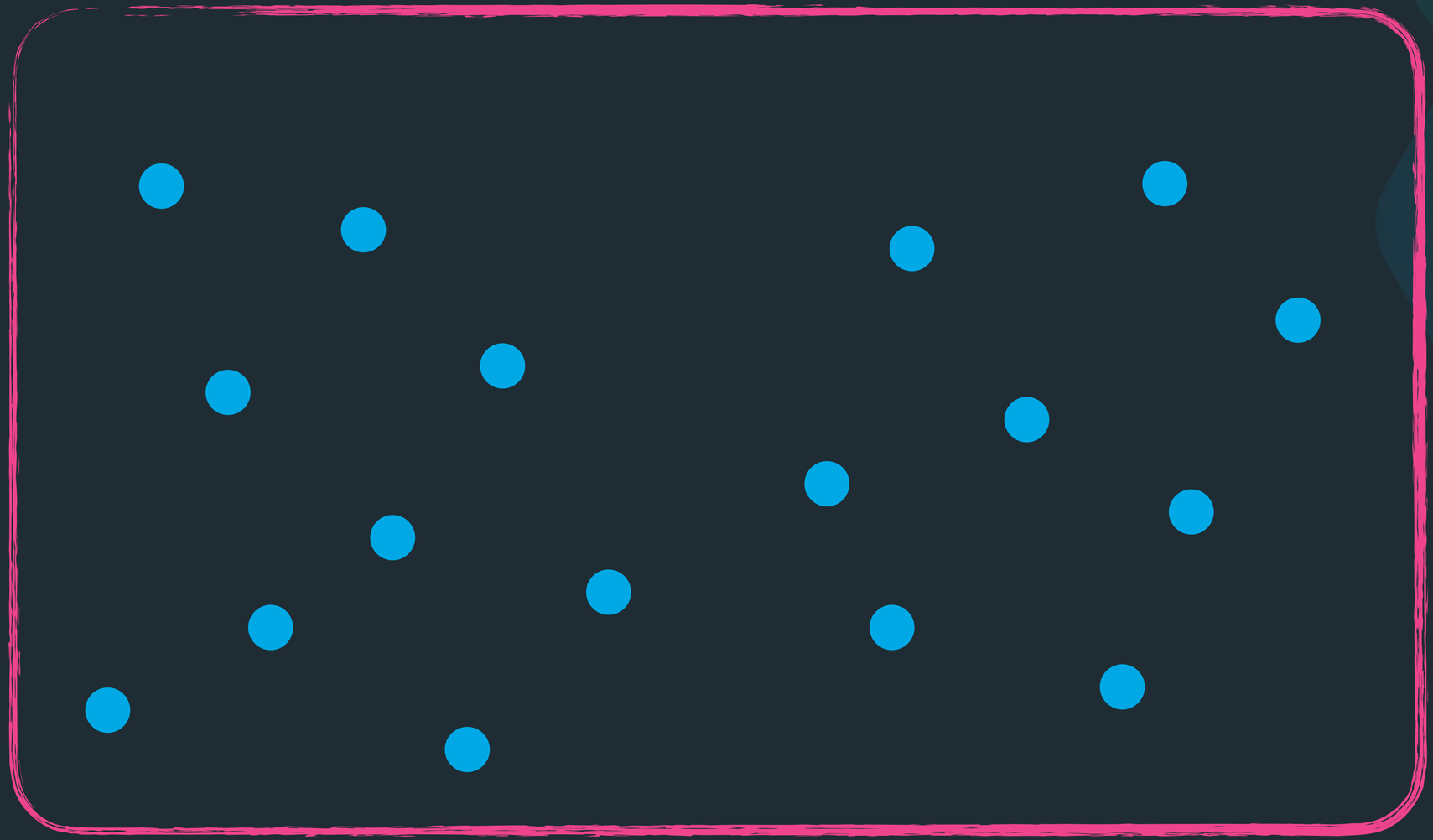
https://www.elastic.co/guide/en/elasticsearch/guide/2.x/pluggable-similarites.html

elastic

Precision vs. recall

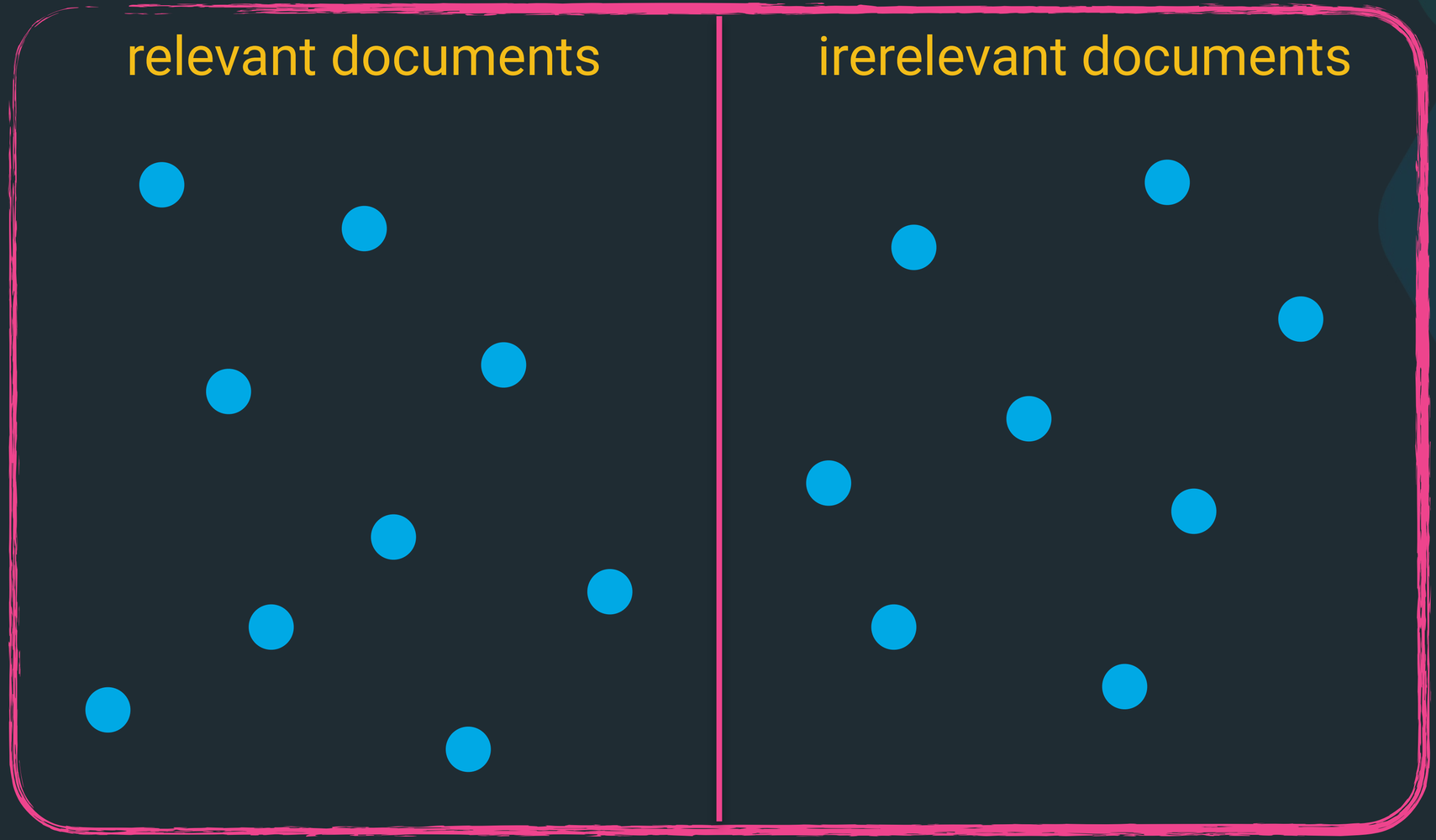


Precision and Recall



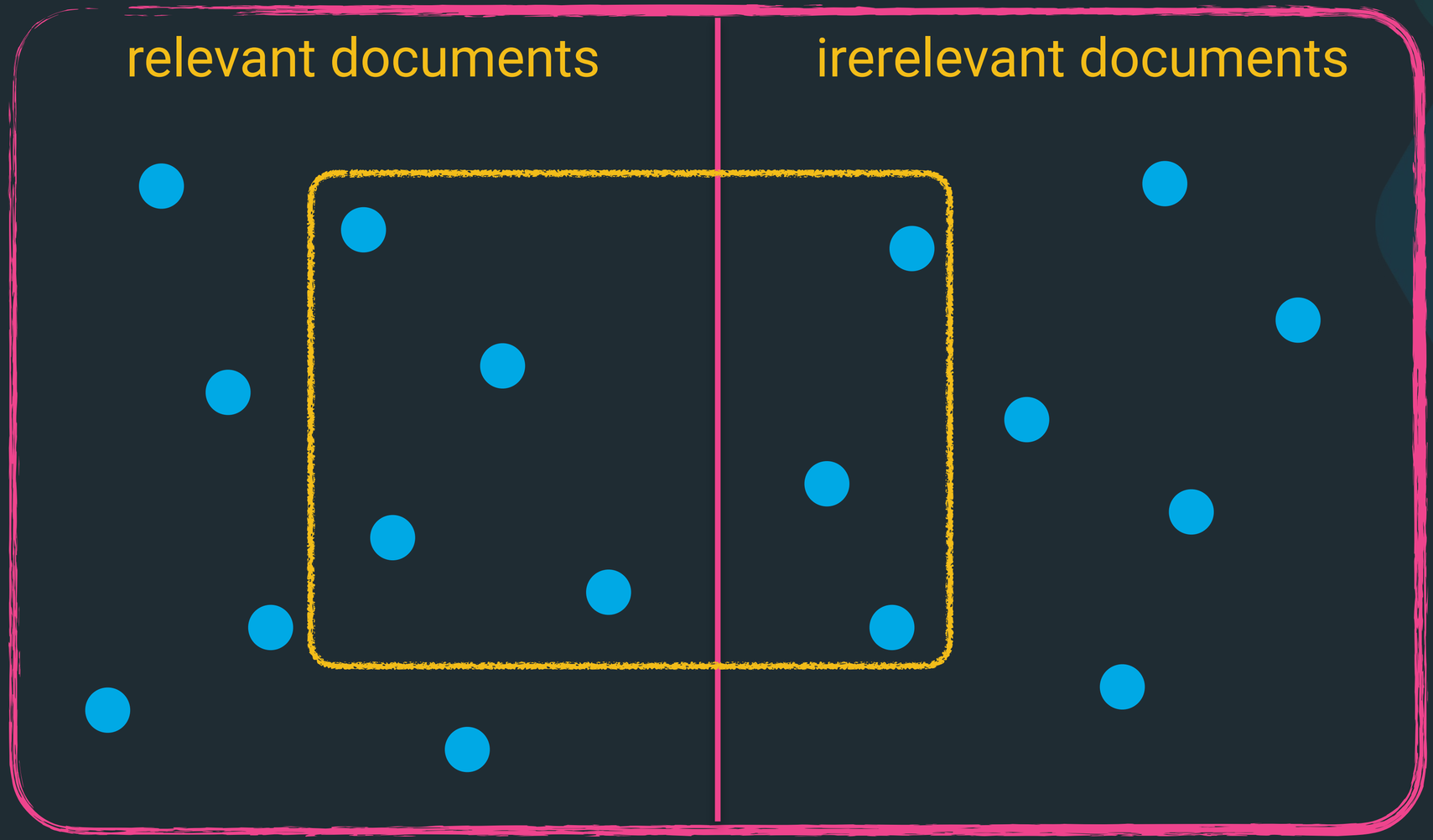
elastic

Precision and Recall



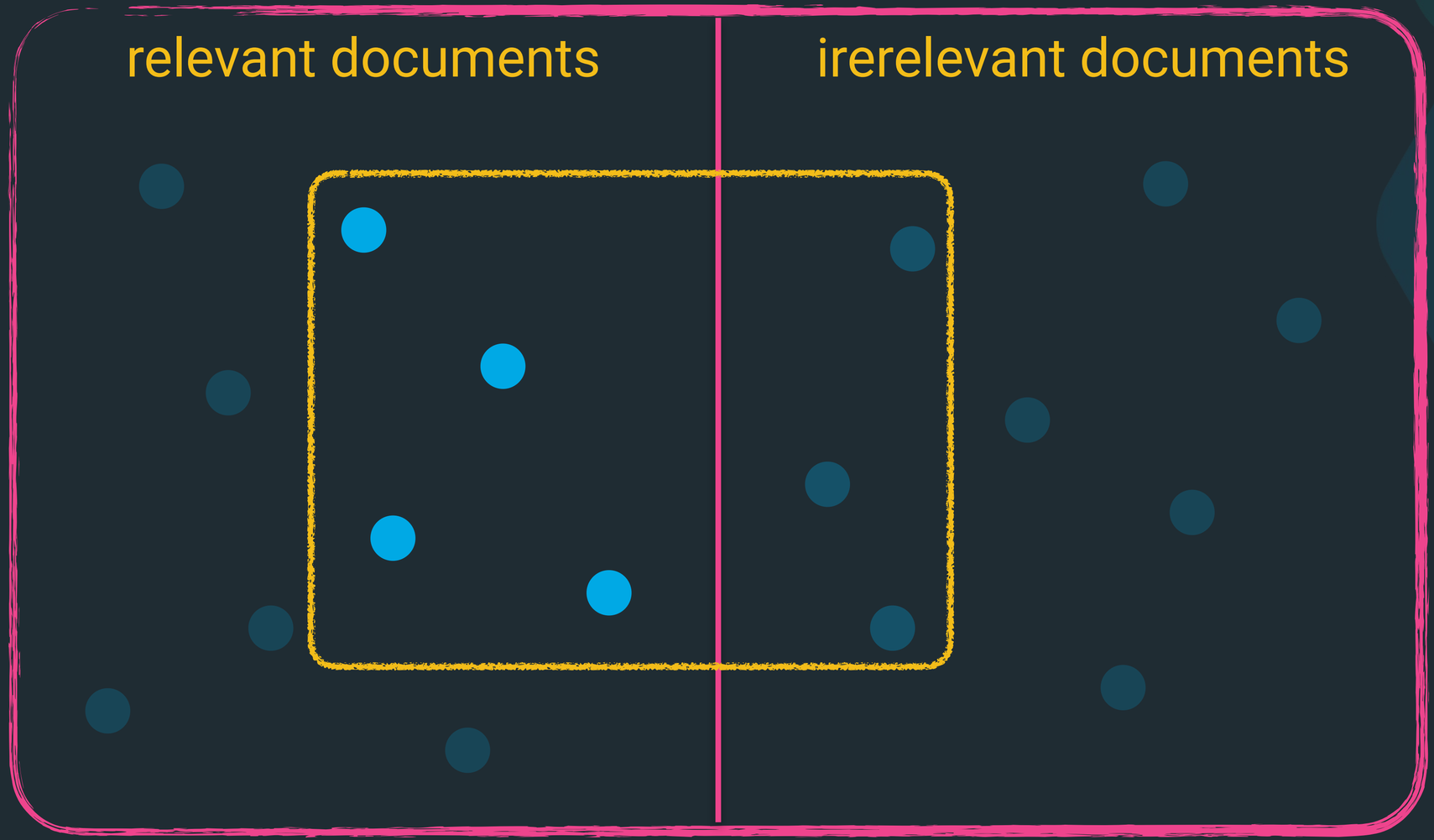
elastic

Precision and Recall



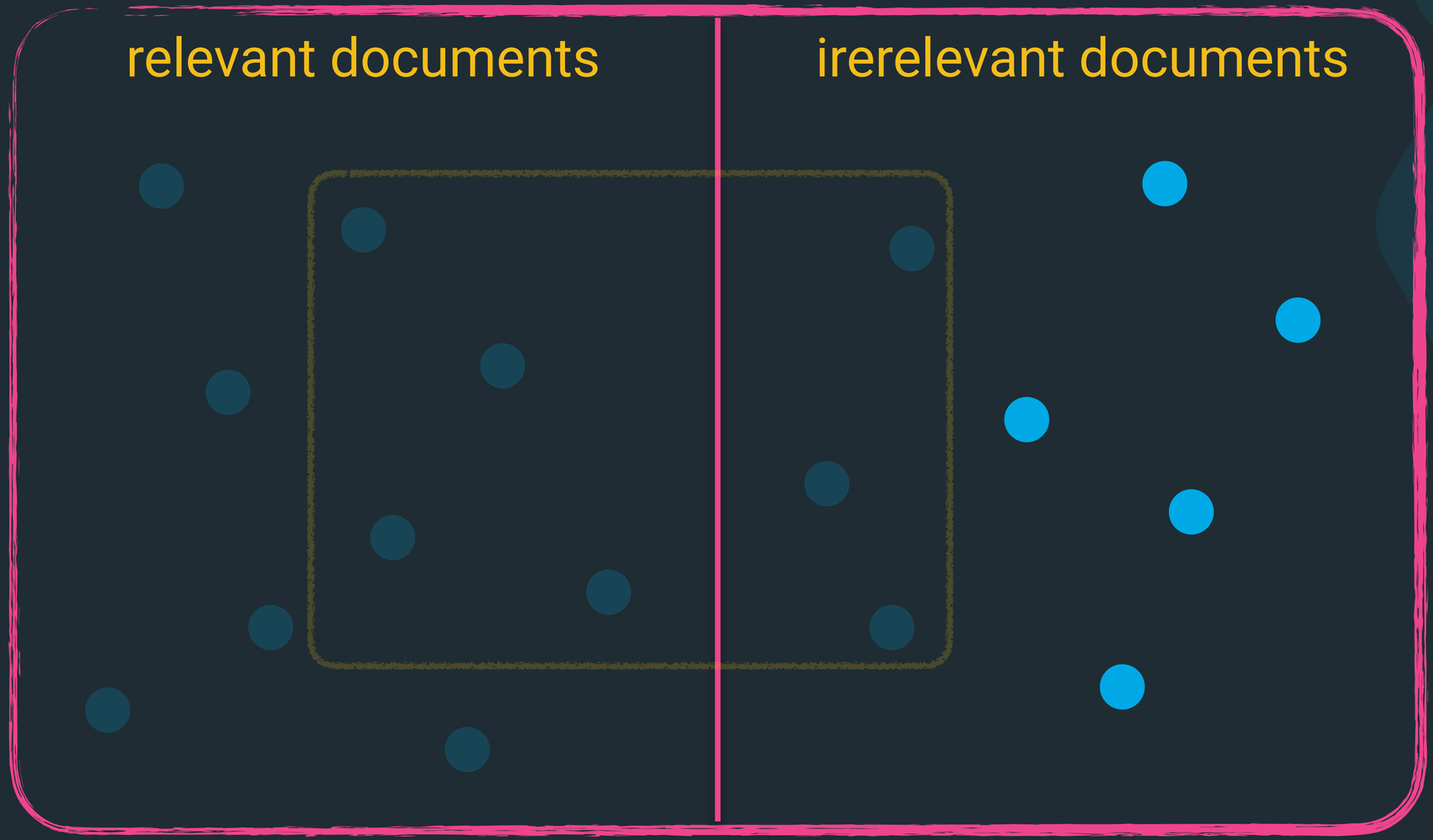
elastic

True positives



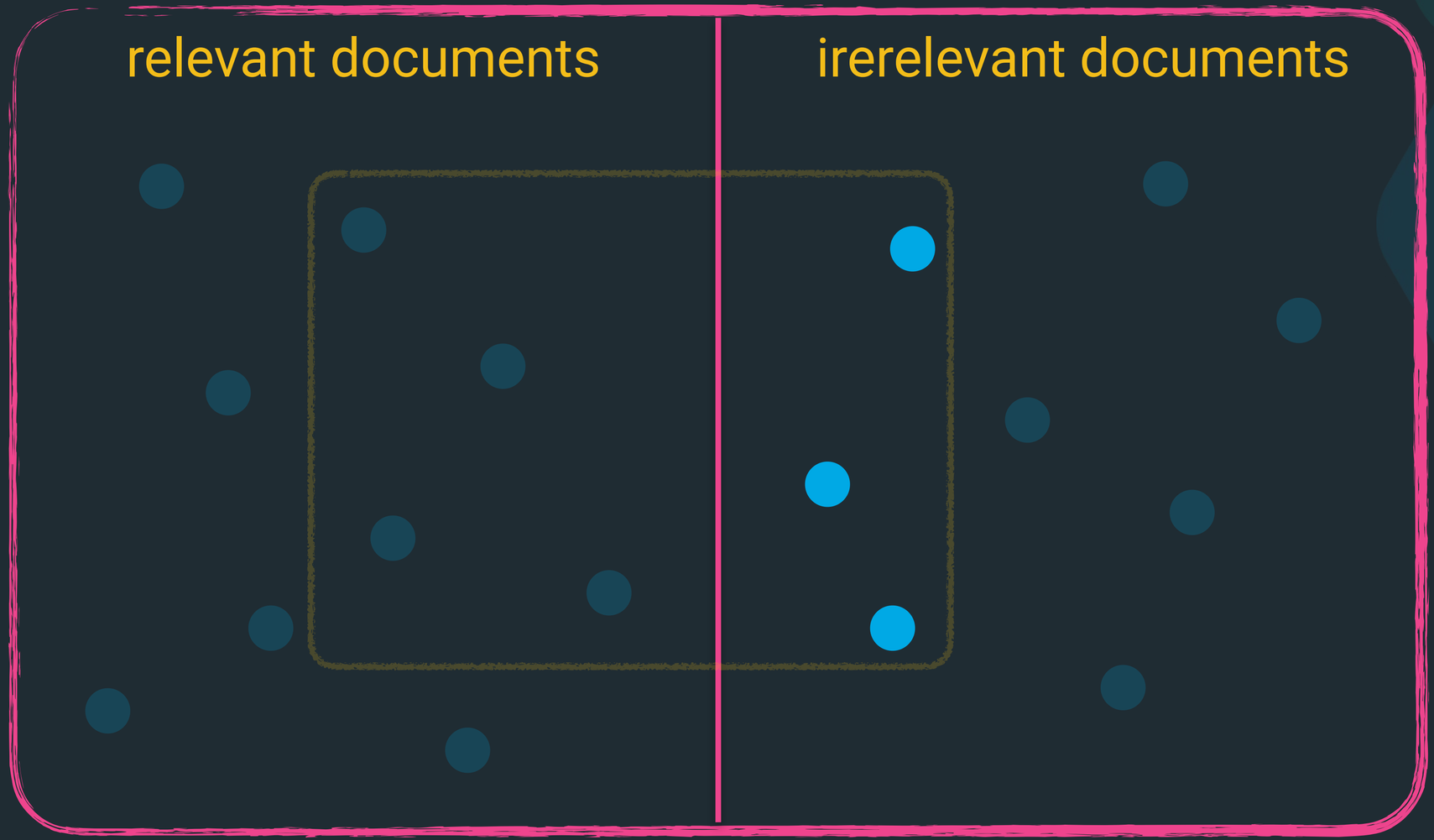
elastic

True negatives



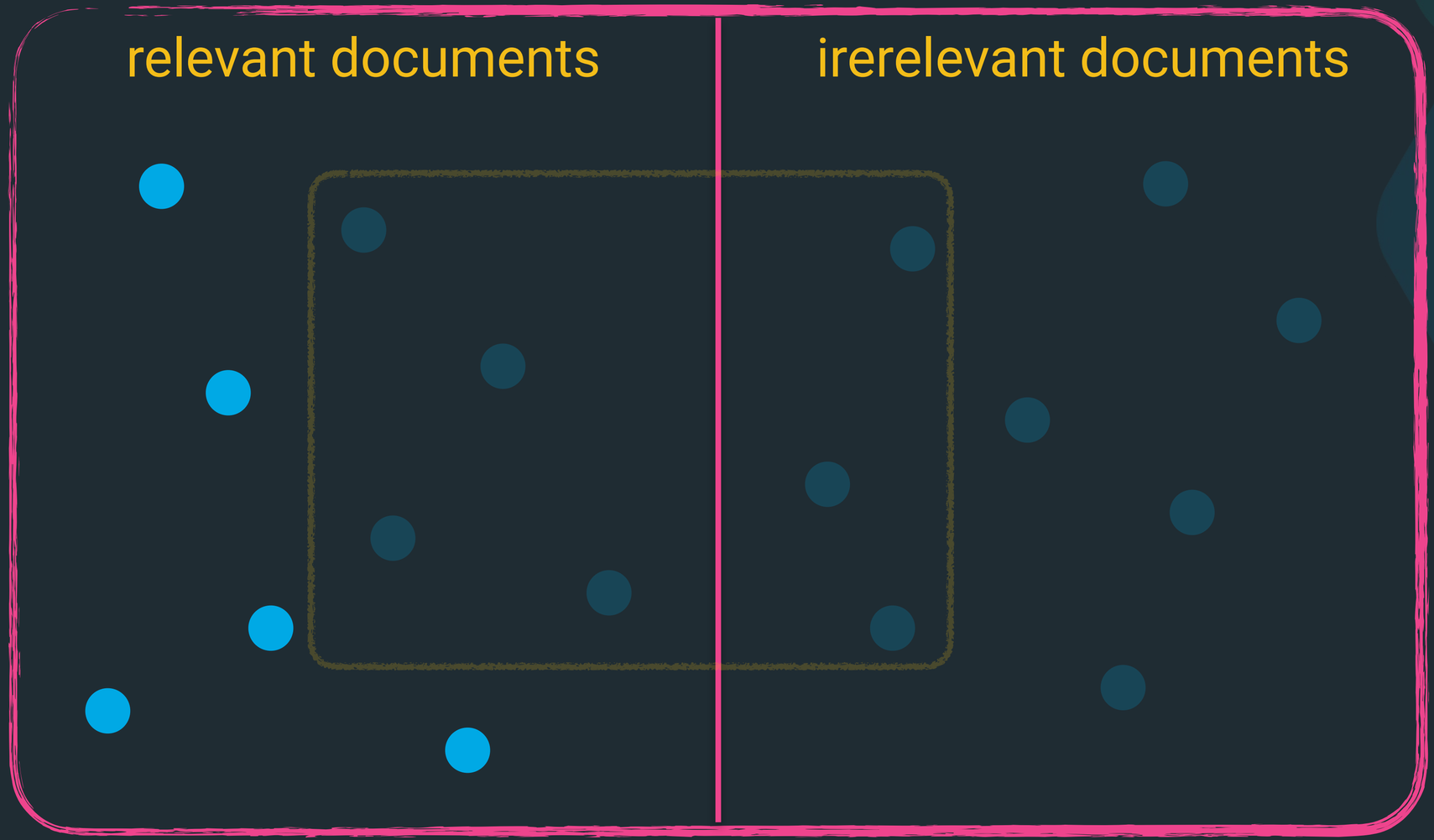
elastic

False positives



elastic

False negatives



elastic

Precision and recall

- Precision: How many selected documents are relevant?
- Recall: How many relevant documents are selected

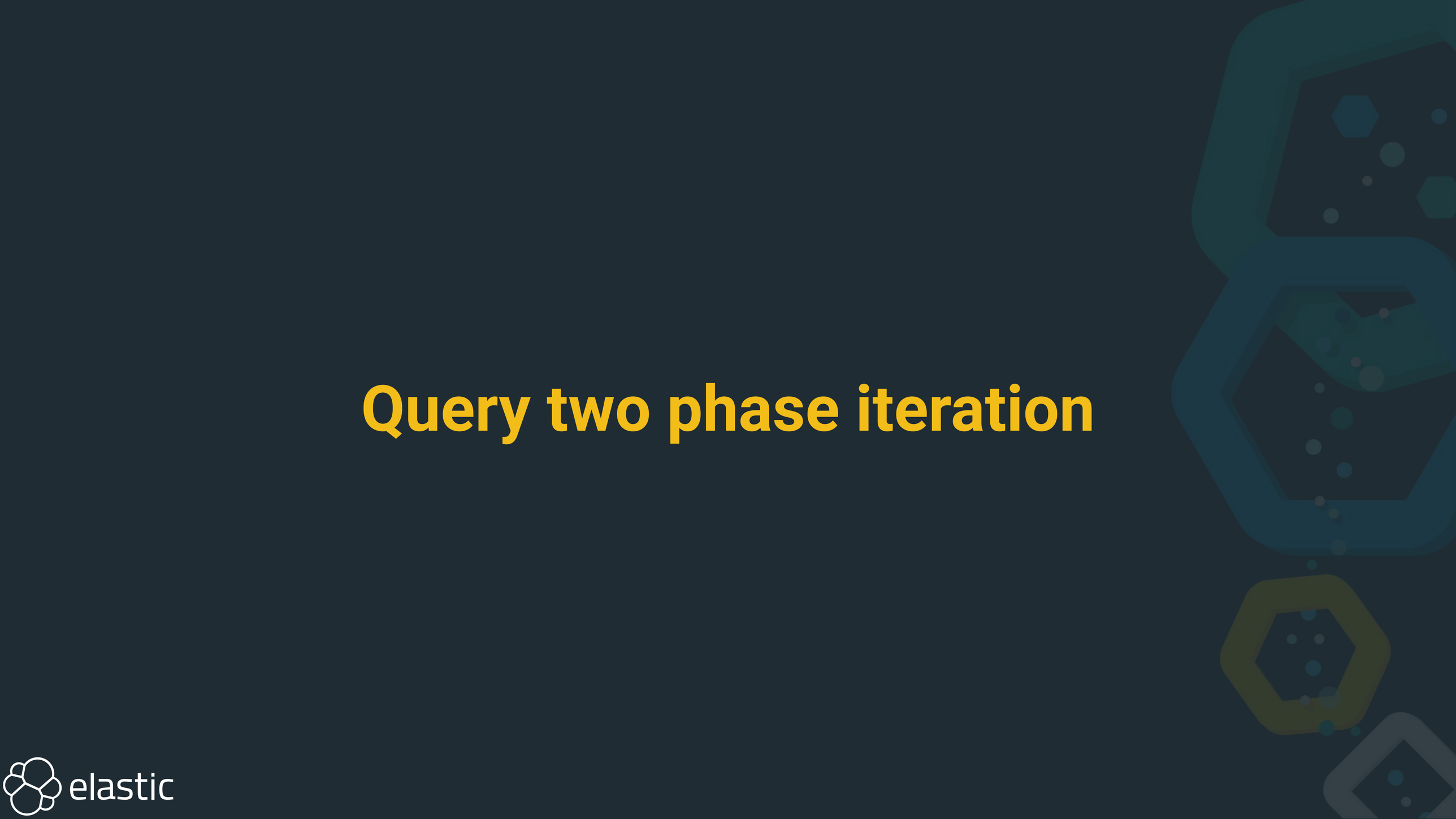
Under the hood

elastic

Optimizations everywhere

- leap frogging, skip lists
- top-k
- two phase iterations
- integer compression

Query two phase iteration



elastic

Two phase iteration: Phrase query

- Phrase query: `"quick fox"`
- Approximation phase: document contains terms `quick` and `fox`
- Verification phase: read positions of terms

Two phase iteration: Geo distance query

- Geo distance query: Distance from reference point
- Approximation phase: bbox around point
- Verification phase: exact distance calculation

Two phase iteration: Geo distance query

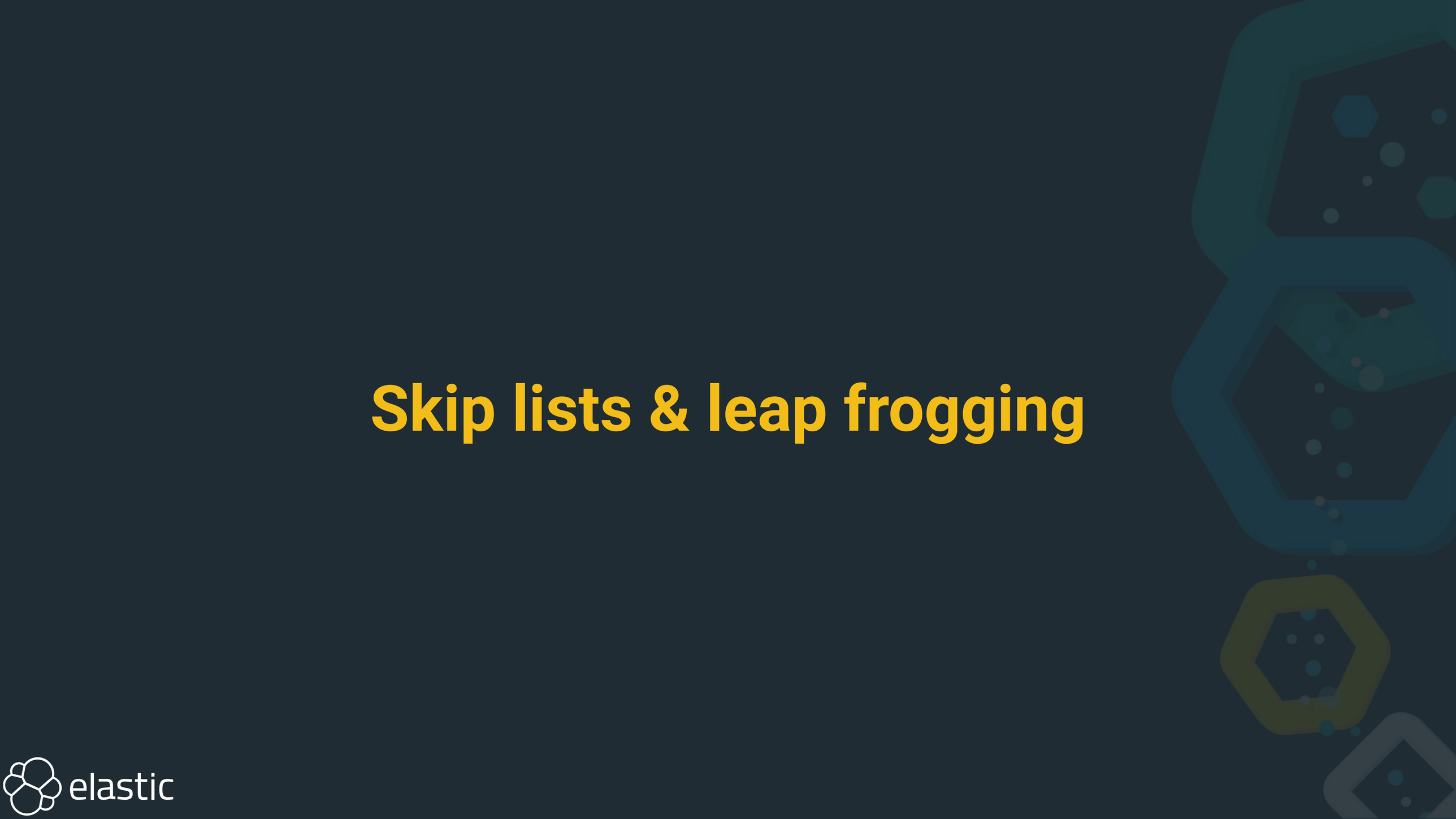
```
GET /my_locations/_search
{
  "query": {
    "bool" : {
      "filter" : {
        "geo_distance" : {
          "distance" : "200km",
          "pin.location" : {
            "lat" : 40,
            "lon" : -70
          }
        }
      }
    }
  }
}
```

elastic

Two phase iteration: several queries

- Powerful when several queries are used
- `"quick fox" AND brown`
- Approximation: `quick AND fox AND brown`
- Verification: `"quick fox"` position check for hits

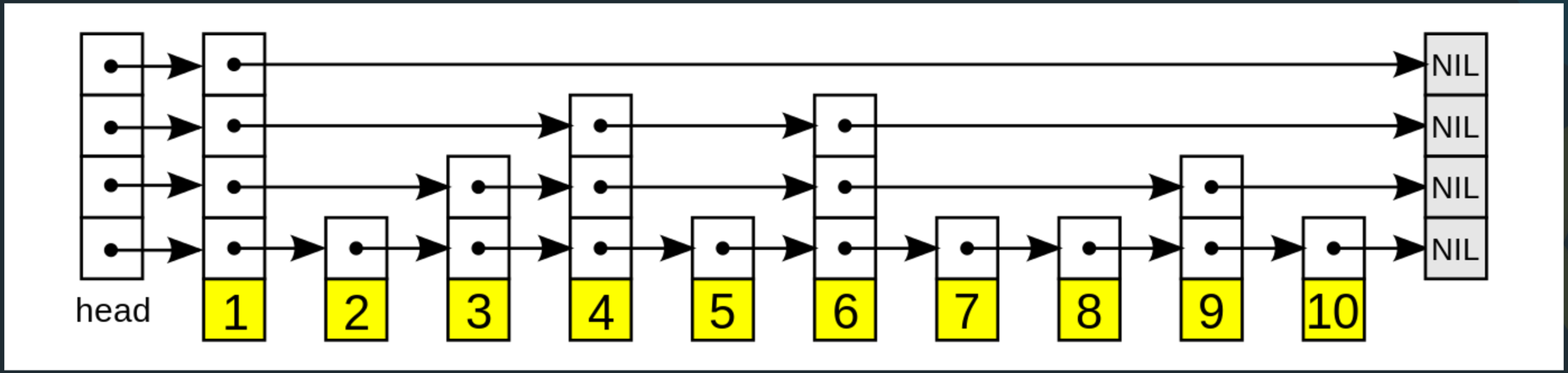
Skip lists & leap frogging



elastic

Skip lists

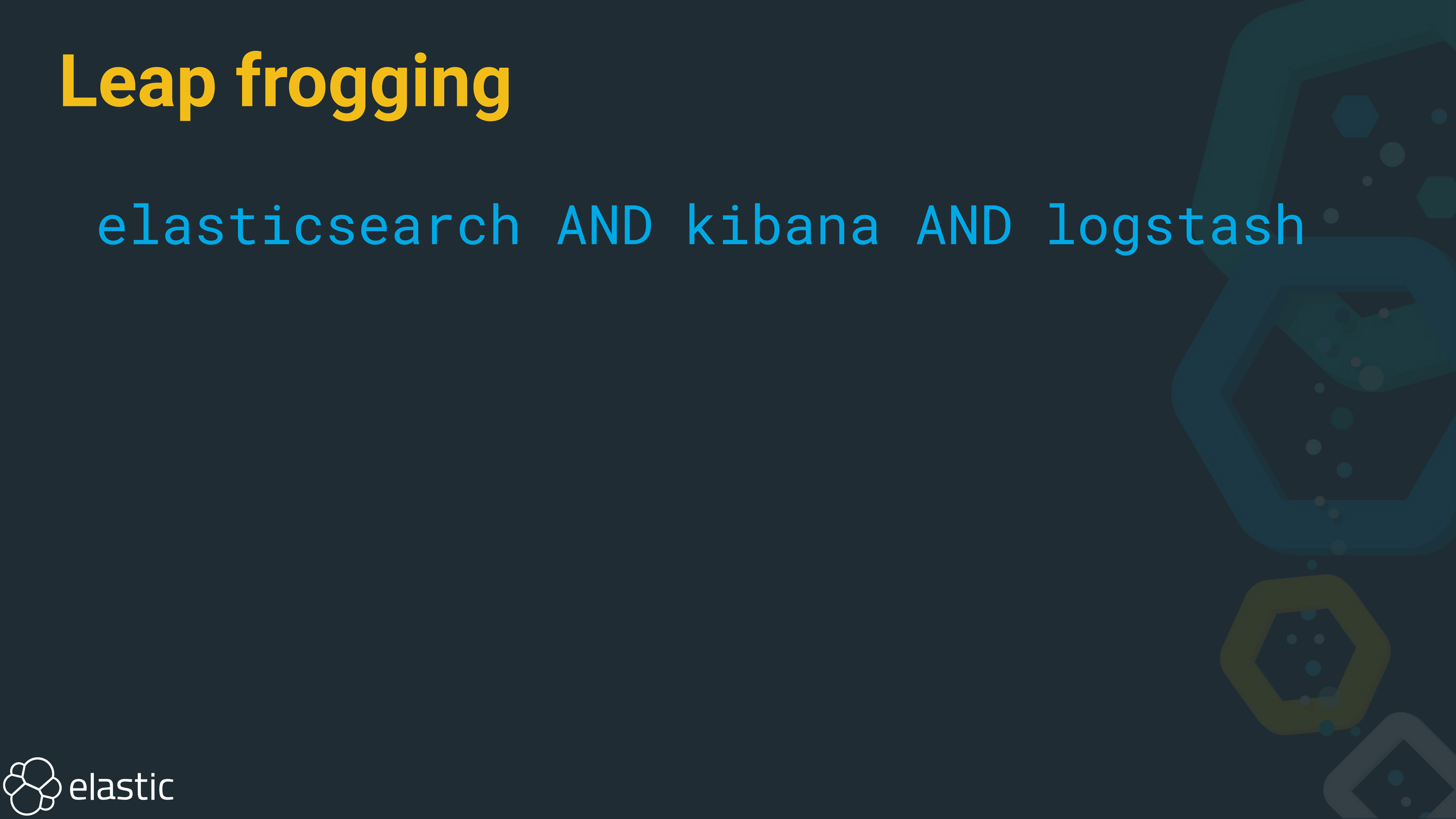
- Term dictionary is a sorted skip list
- Skip list is a linked list with 'express lanes' to leap forward



https://en.wikipedia.org/wiki/Skip_list

Leap frogging

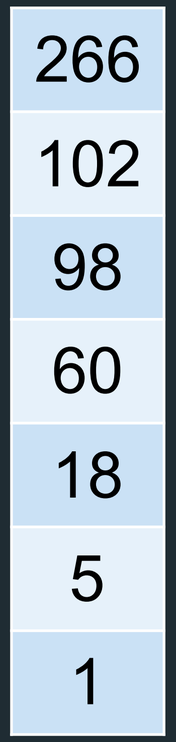
```
elasticsearch AND kibana AND logstash
```

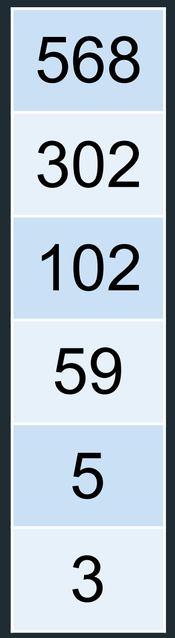


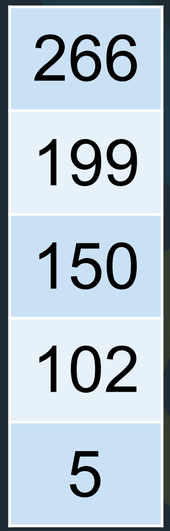
elastic

Leap frogging

```
elasticsearch AND kibana AND logstash
```



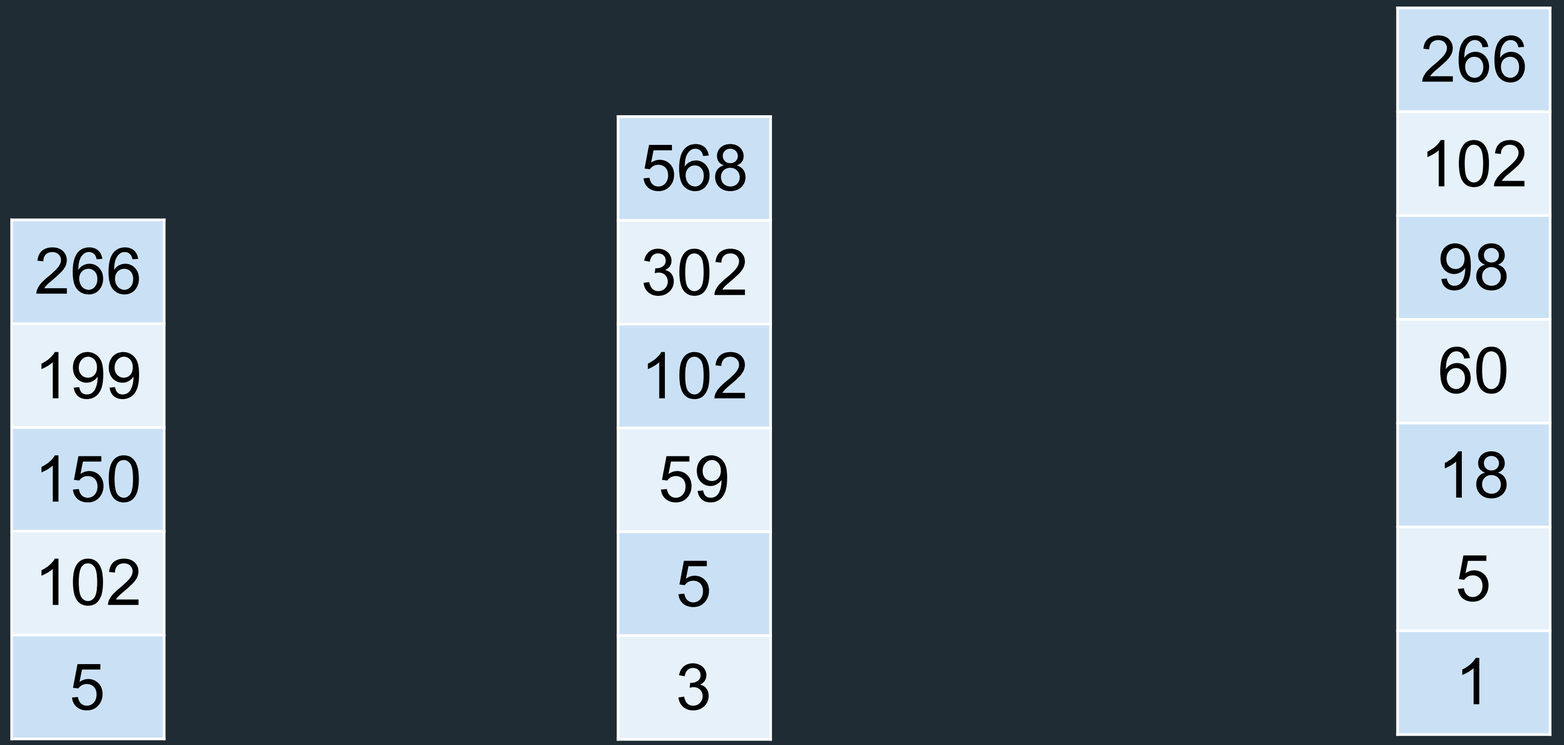




elastic

Leap frogging

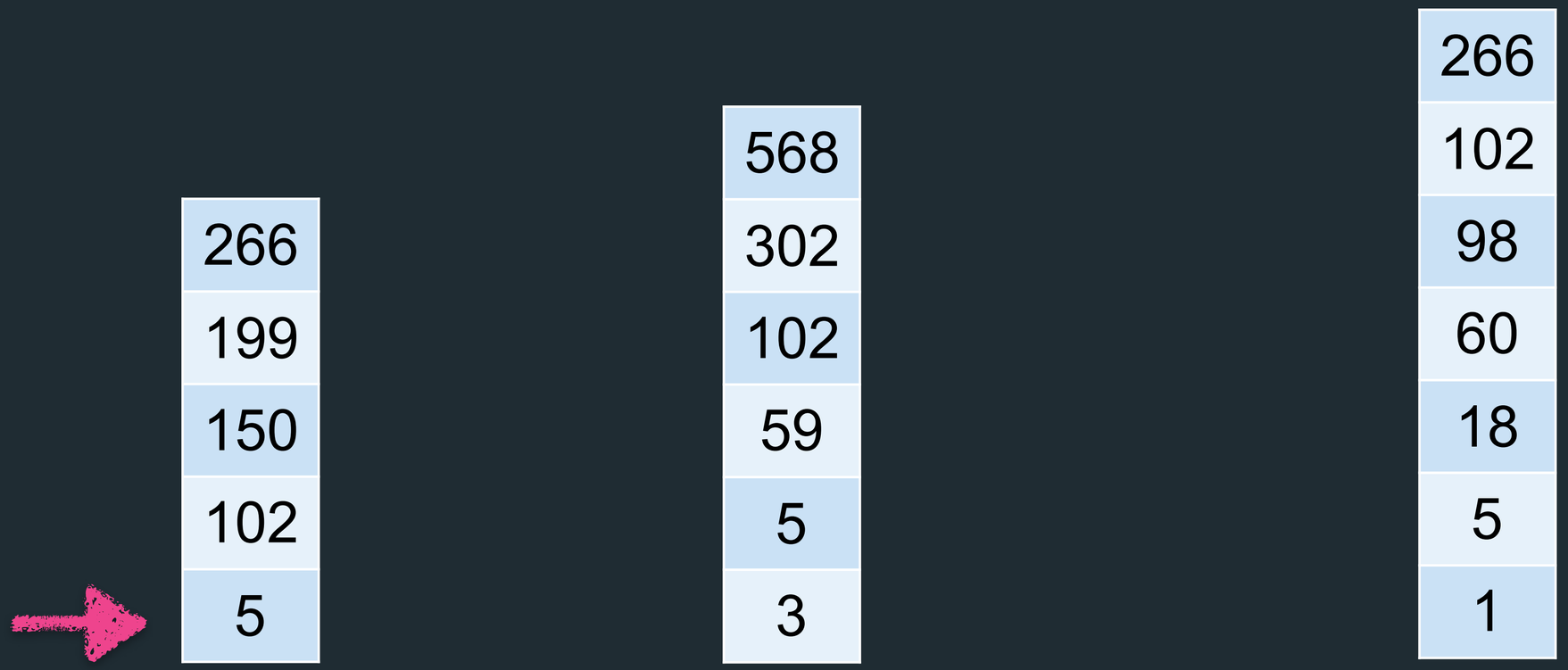
```
logstash AND kibana AND elasticsearch
```



elastic

Leap frogging

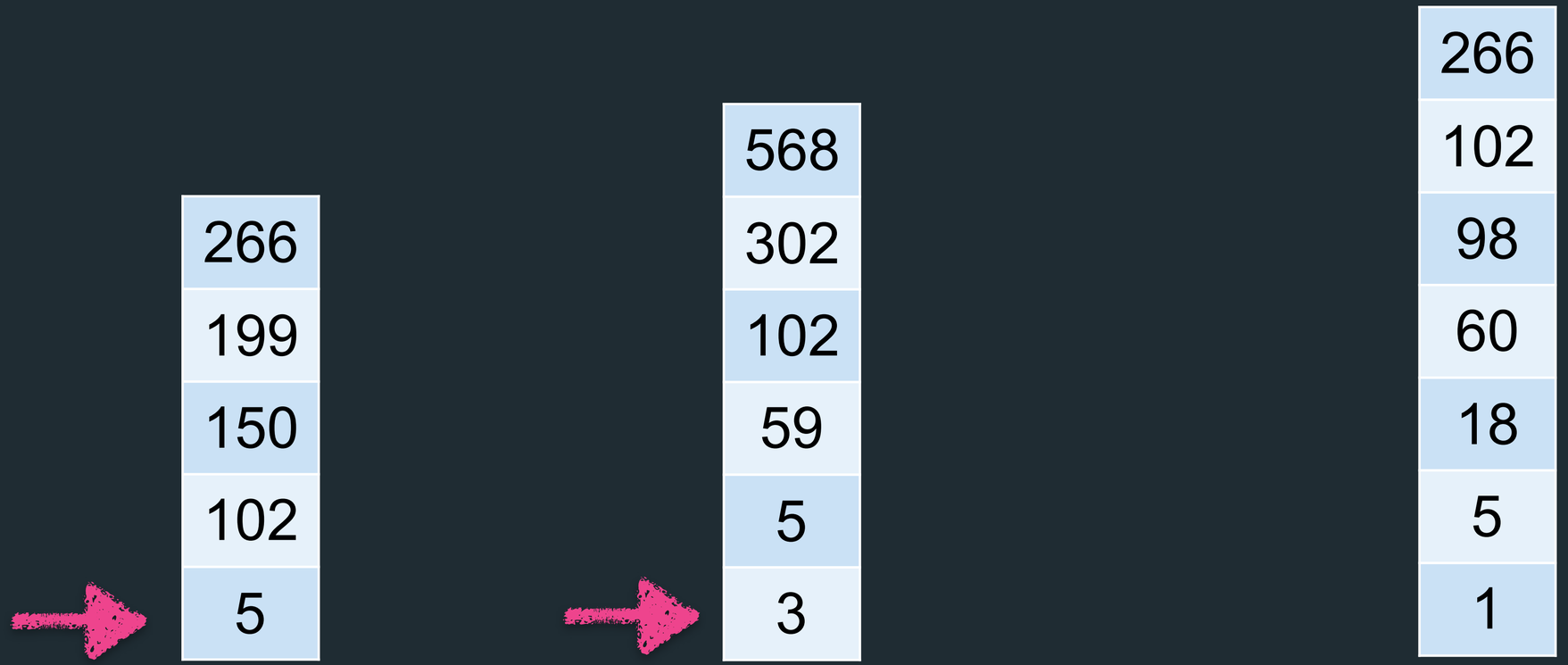
```
logstash AND kibana AND elasticsearch
```



elastic

Leap frogging

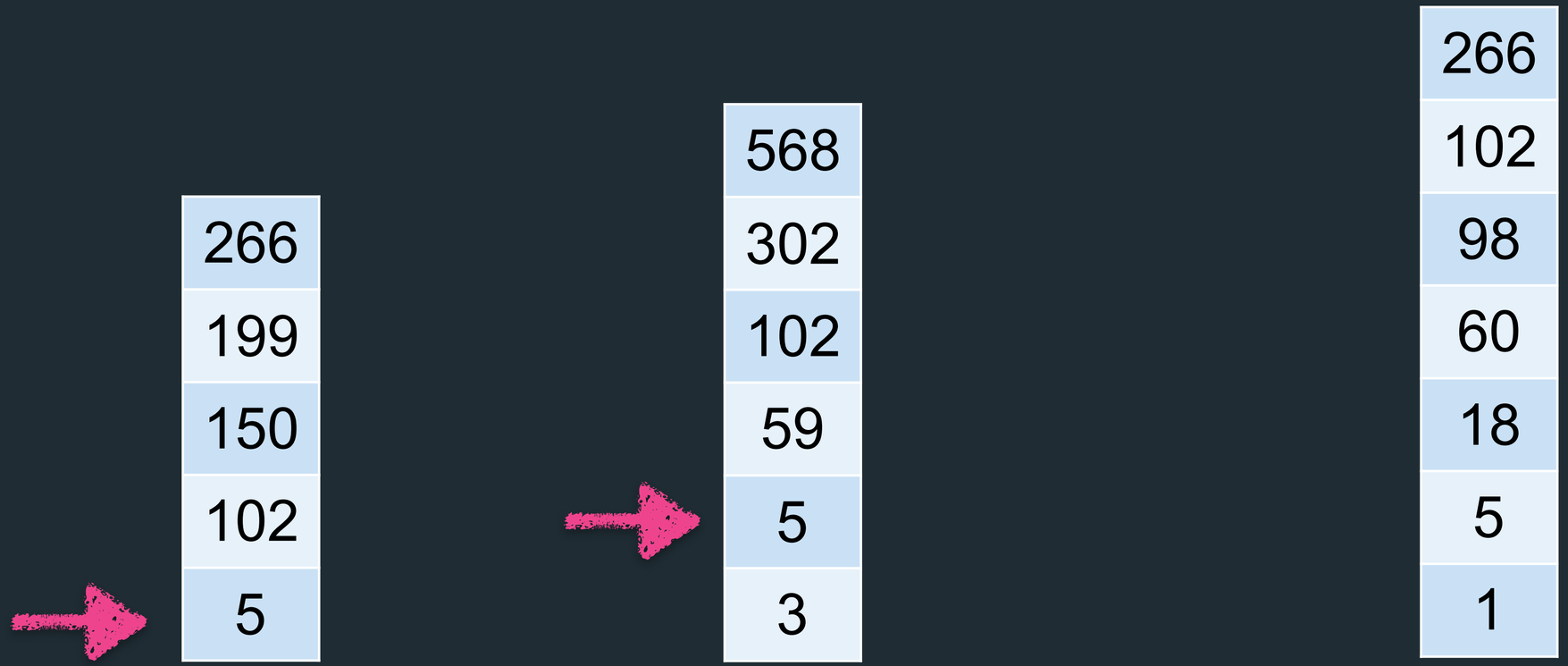
```
logstash AND kibana AND elasticsearch
```



elastic

Leap frogging

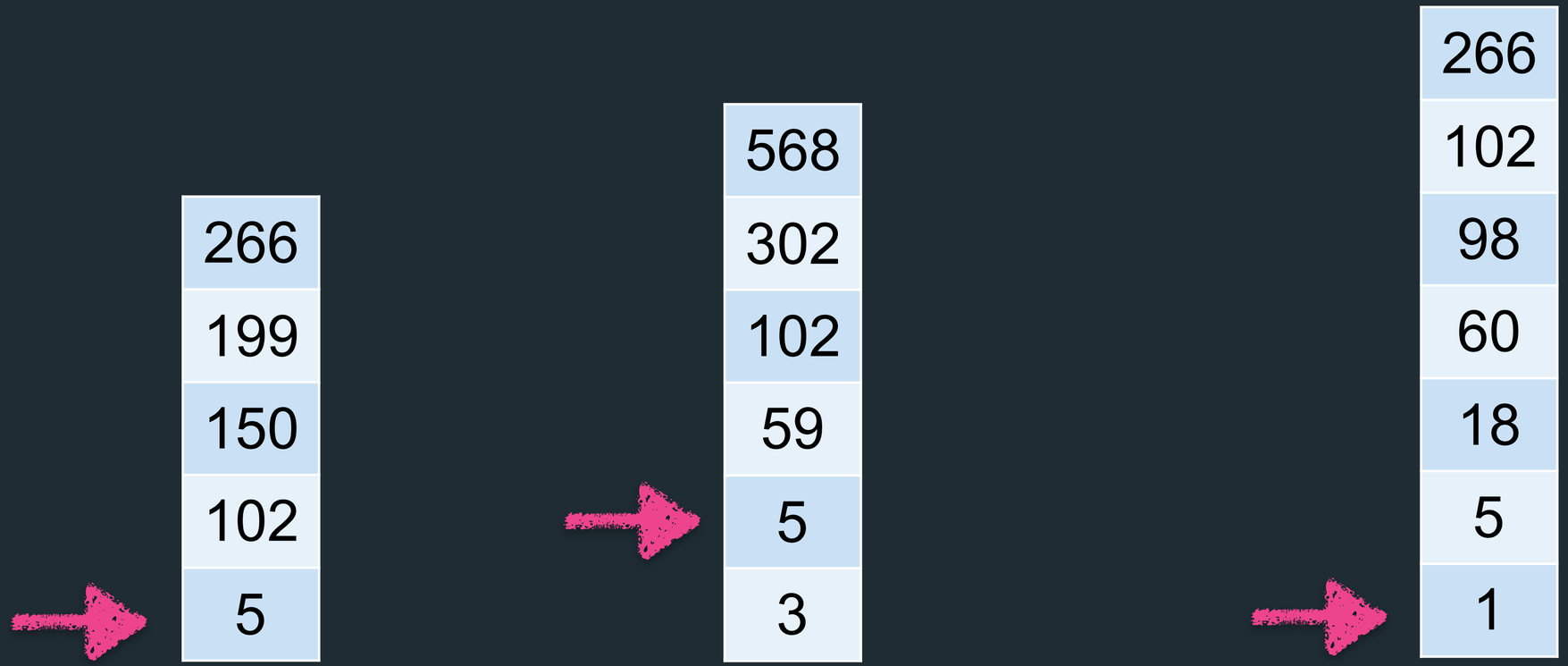
```
logstash AND kibana AND elasticsearch
```



elastic

Leap frogging

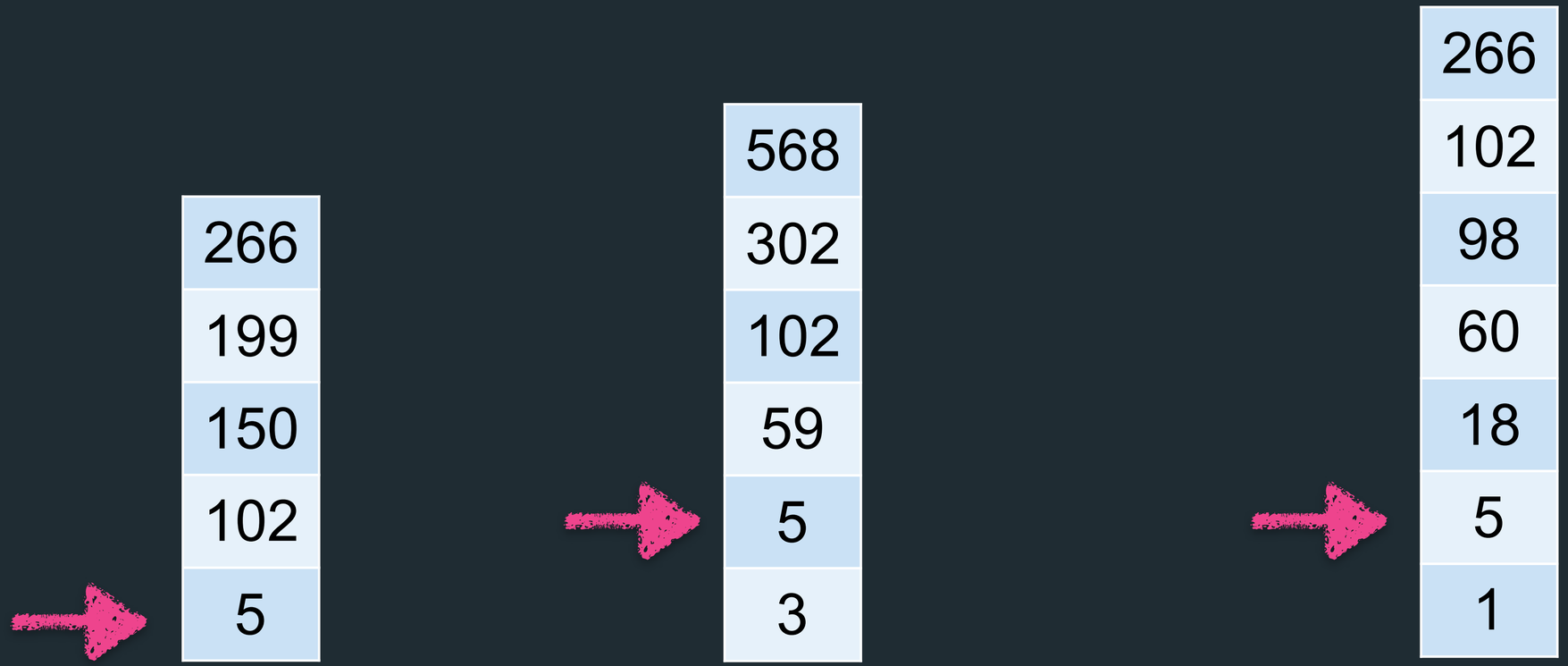
```
logstash AND kibana AND elasticsearch
```



elastic

Leap frogging

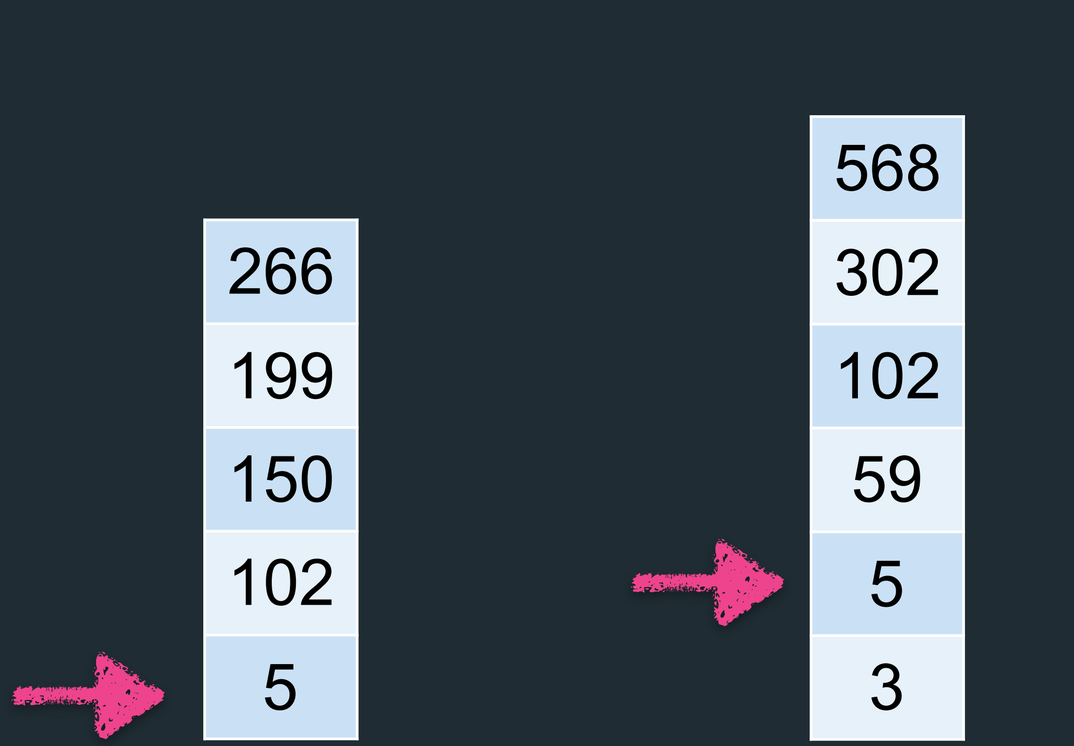
```
logstash AND kibana AND elasticsearch
```



elastic

Leap frogging

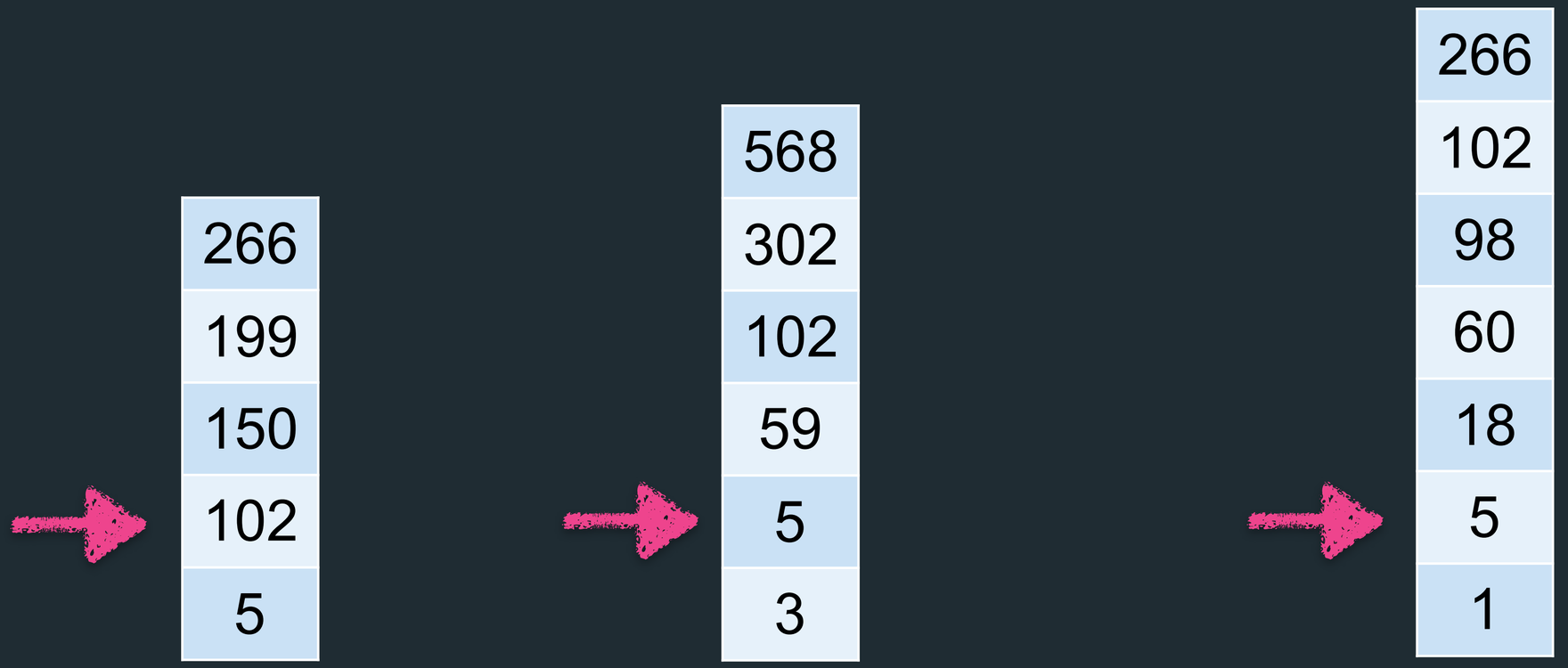
```
logstash AND kibana AND elasticsearch
```



elastic

Leap frogging

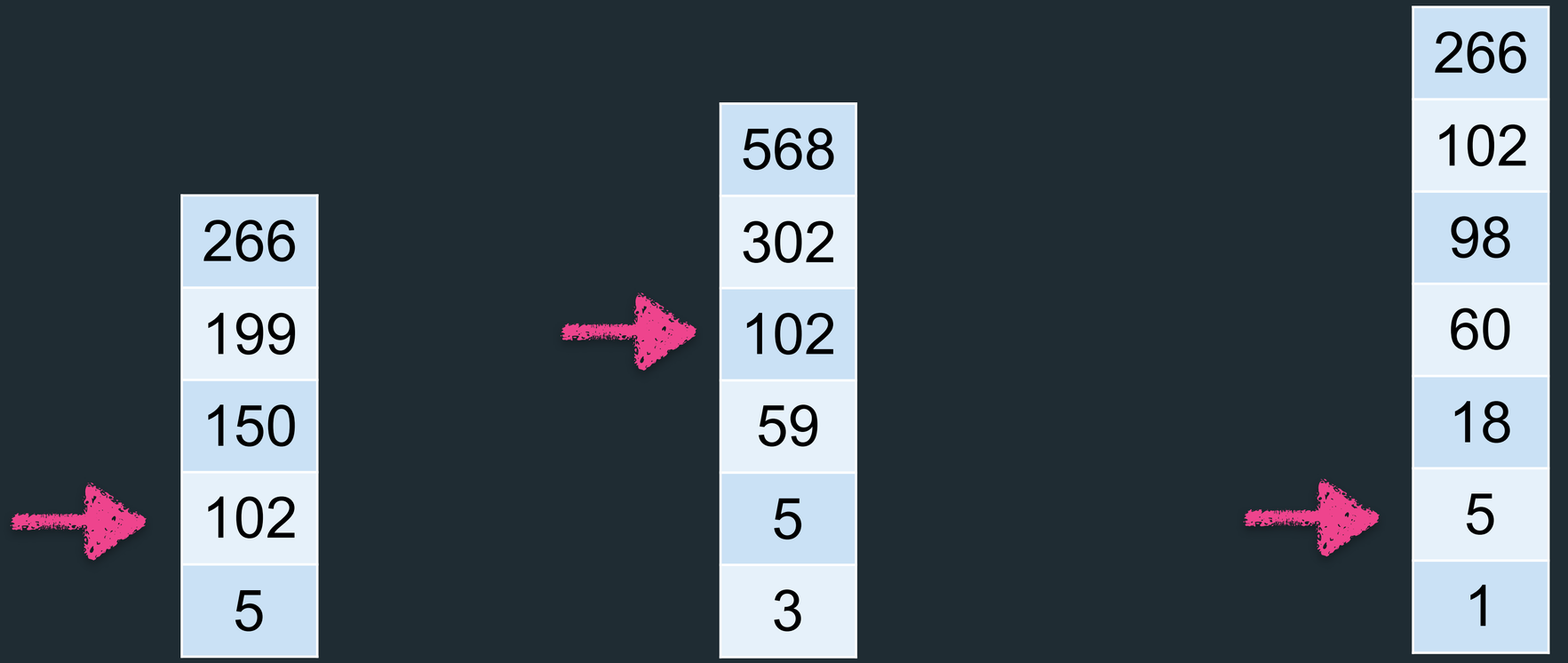
```
logstash AND kibana AND elasticsearch
```



elastic

Leap frogging

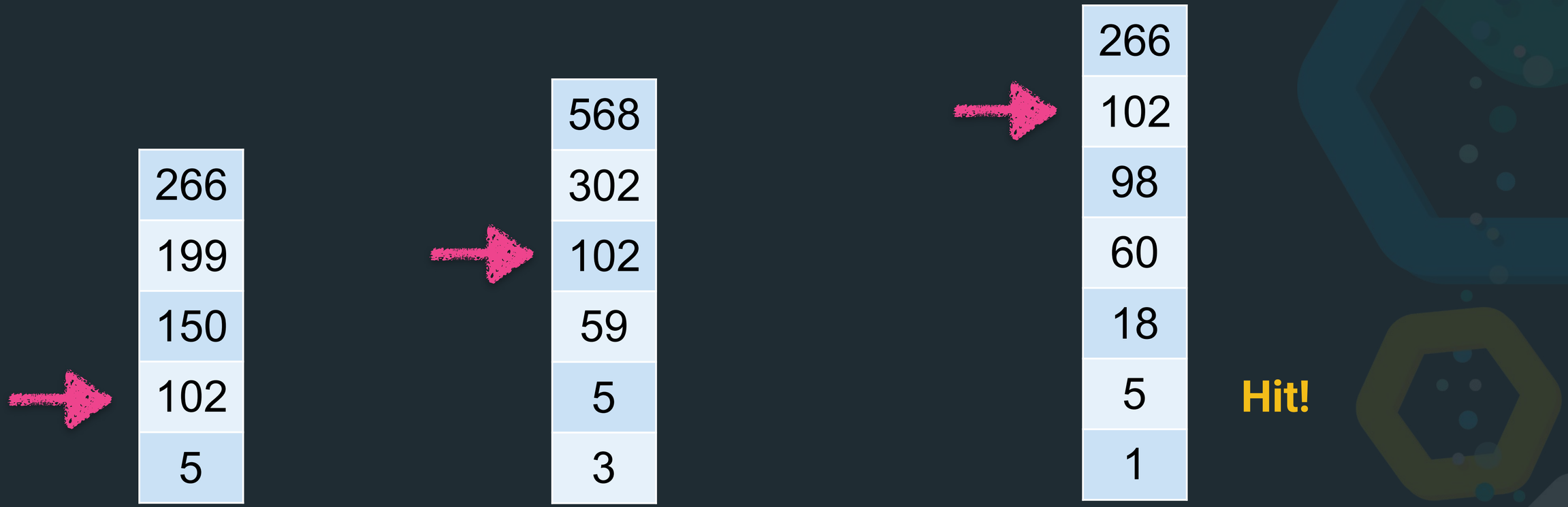
```
logstash AND kibana AND elasticsearch
```



elastic

Leap frogging

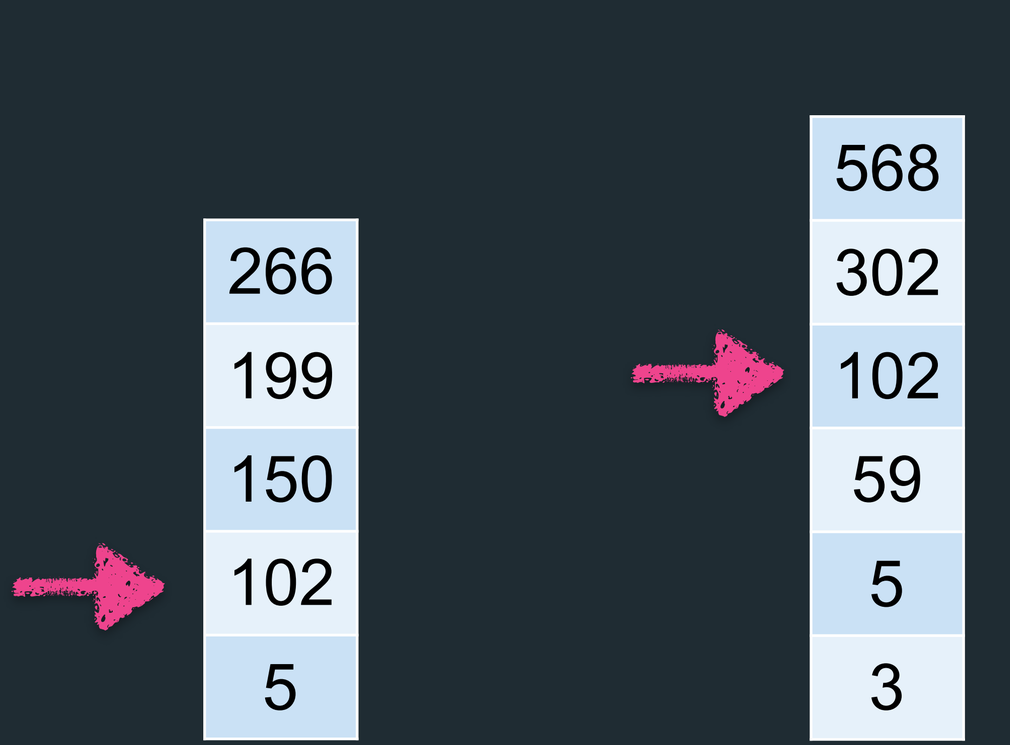
```
logstash AND kibana AND elasticsearch
```



elastic

Leap frogging

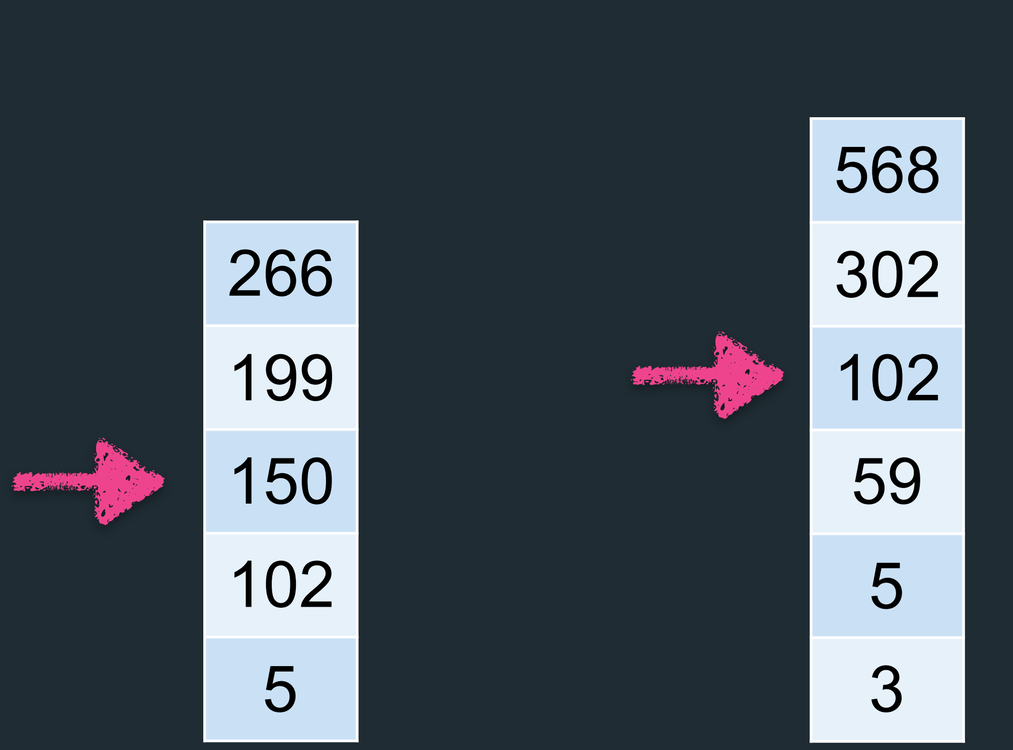
```
logstash AND kibana AND elasticsearch
```



elastic

Leap frogging

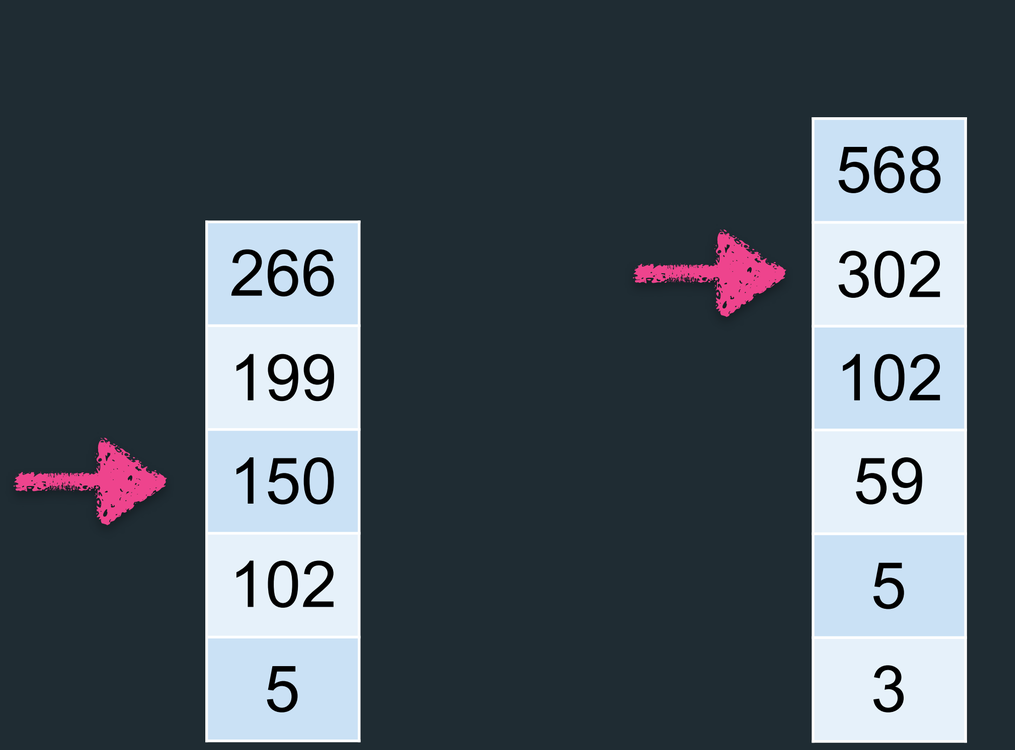
```
logstash AND kibana AND elasticsearch
```



elastic

Leap frogging

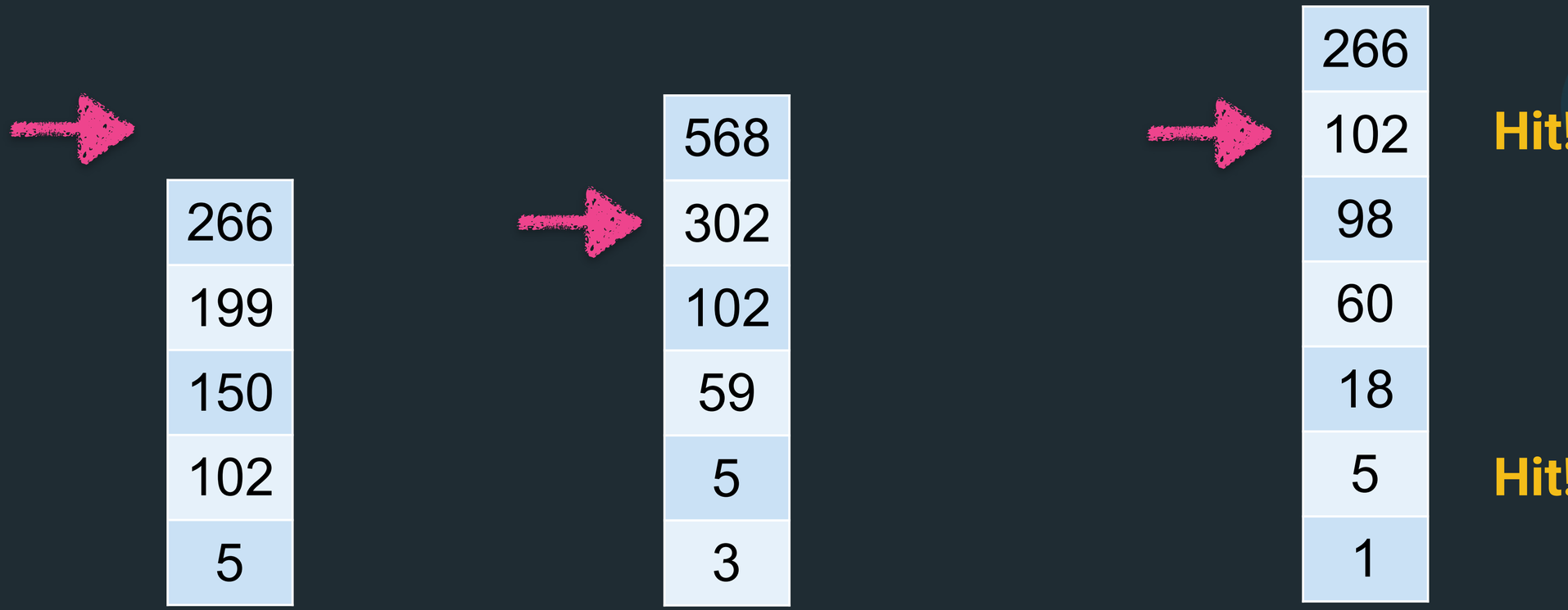
```
logstash AND kibana AND elasticsearch
```



elastic

Leap frogging

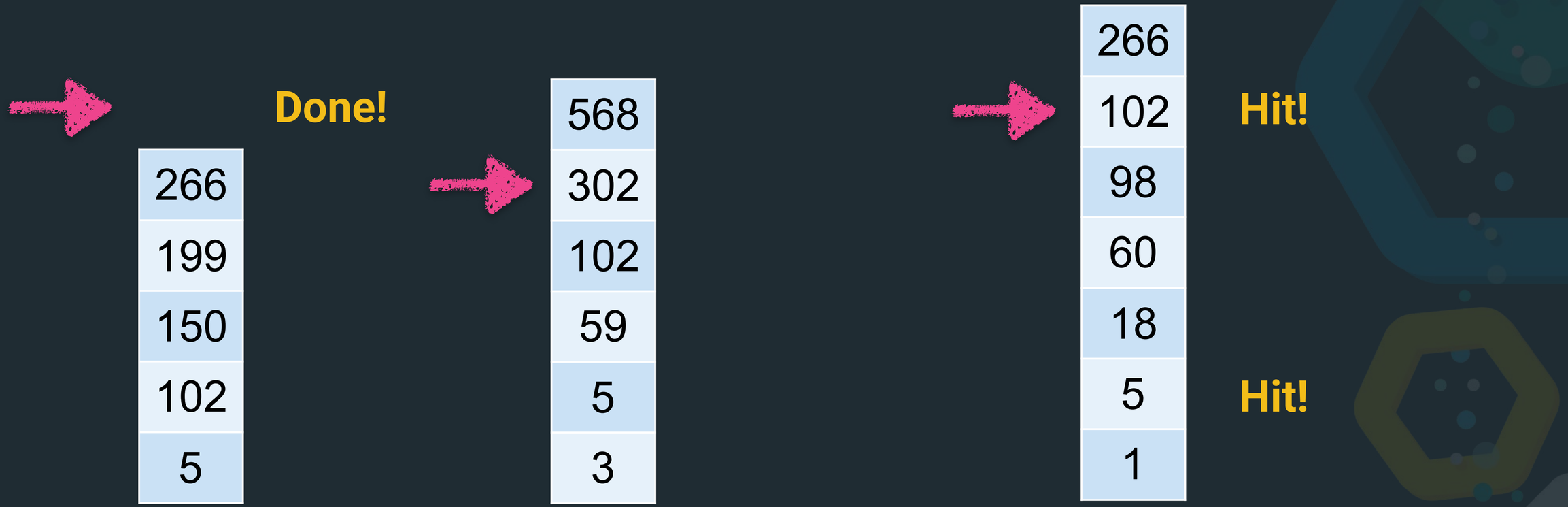
```
logstash AND kibana AND elasticsearch
```



elastic

Leap frogging

```
logstash AND kibana AND elasticsearch
```



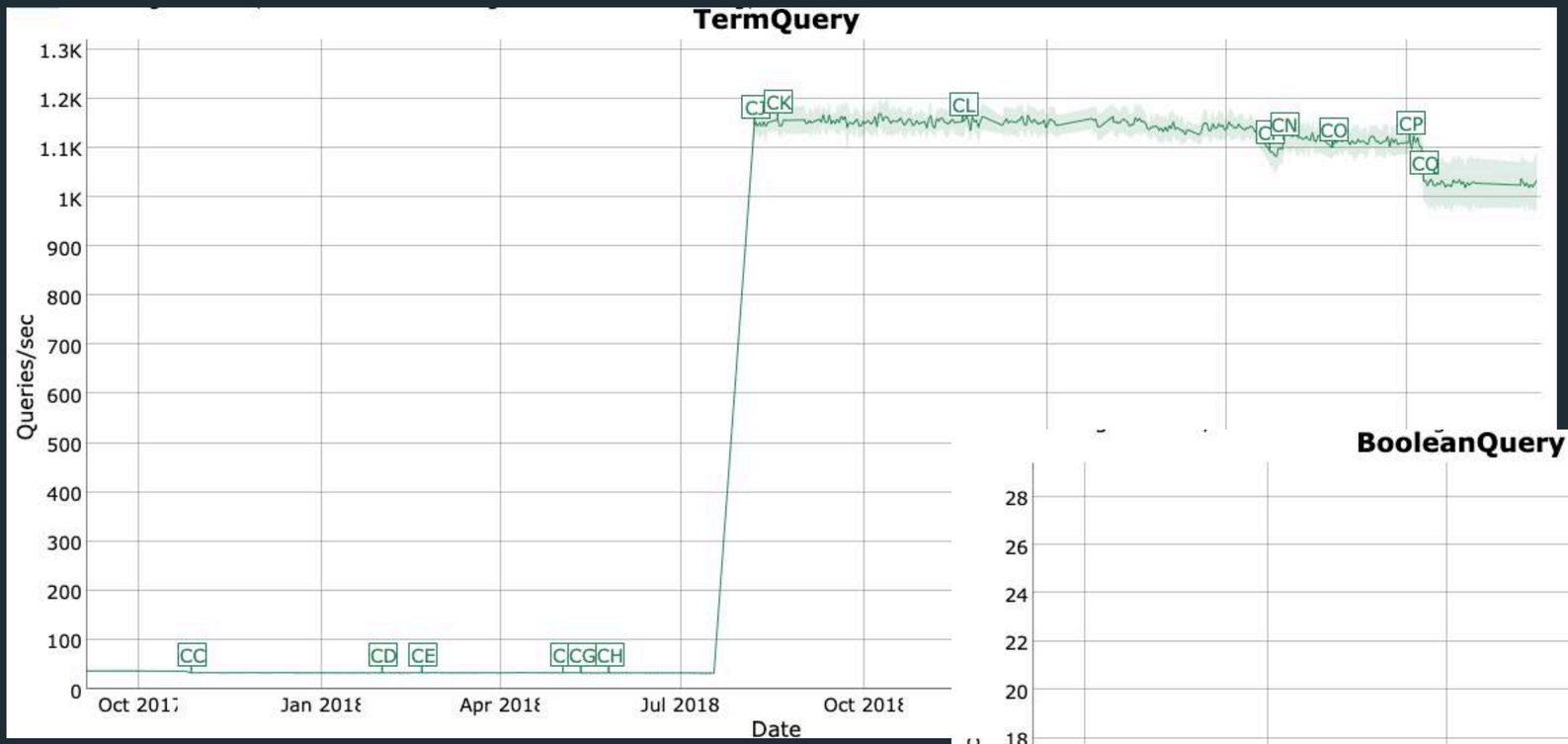
elastic

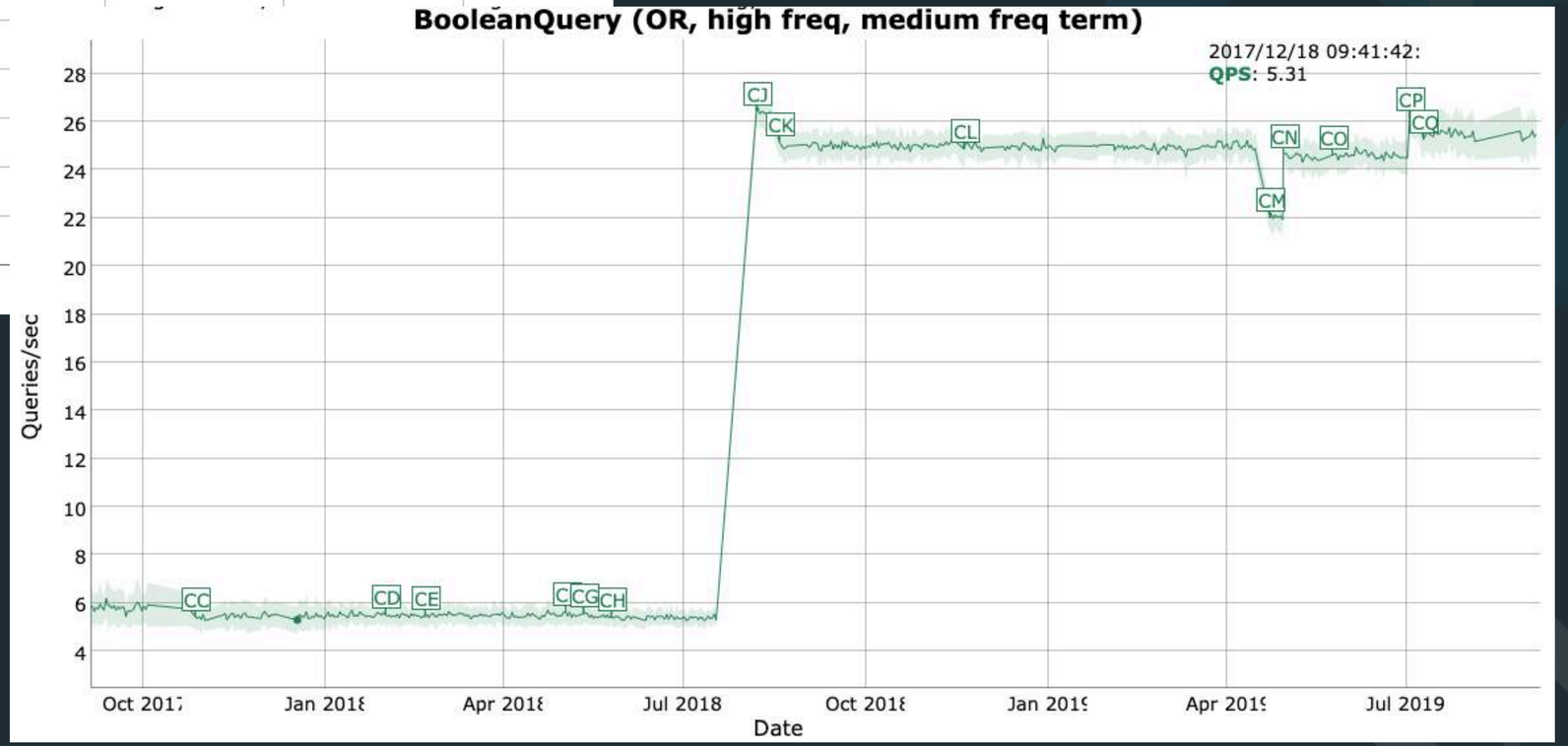
Top-k retrieval

Top-k retrieval

- `elasticsearch OR kibana`
- top 10 results wanted
- maximum score for `kibana` is 3.0
- maximum score for `elasticsearch` is 5.0
- collecting documents: when 10th hit has score > 3, then only documents with `elasticsearch` need to be collected
- total hit count is not accurate

Top-k retrieval





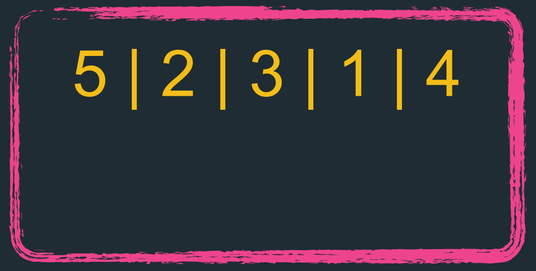
elastic

Index sorting

elastic

Order index by field values

- each segment is sorted before write
- criteria can be chosen by the user

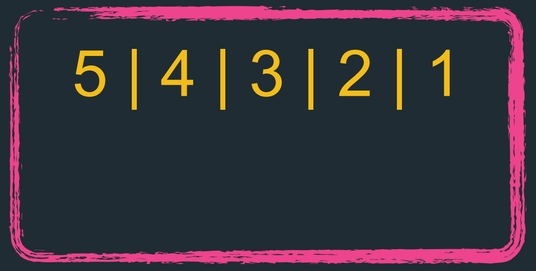


retrieve	sort	top 2
5 | 2 | 3 | 1 | 4	5 | 4 | 3 | 2 | 1	5 | 4

elastic

Order index by field values

- each segment is sorted before write
- criteria can be chosen by the user



early termination

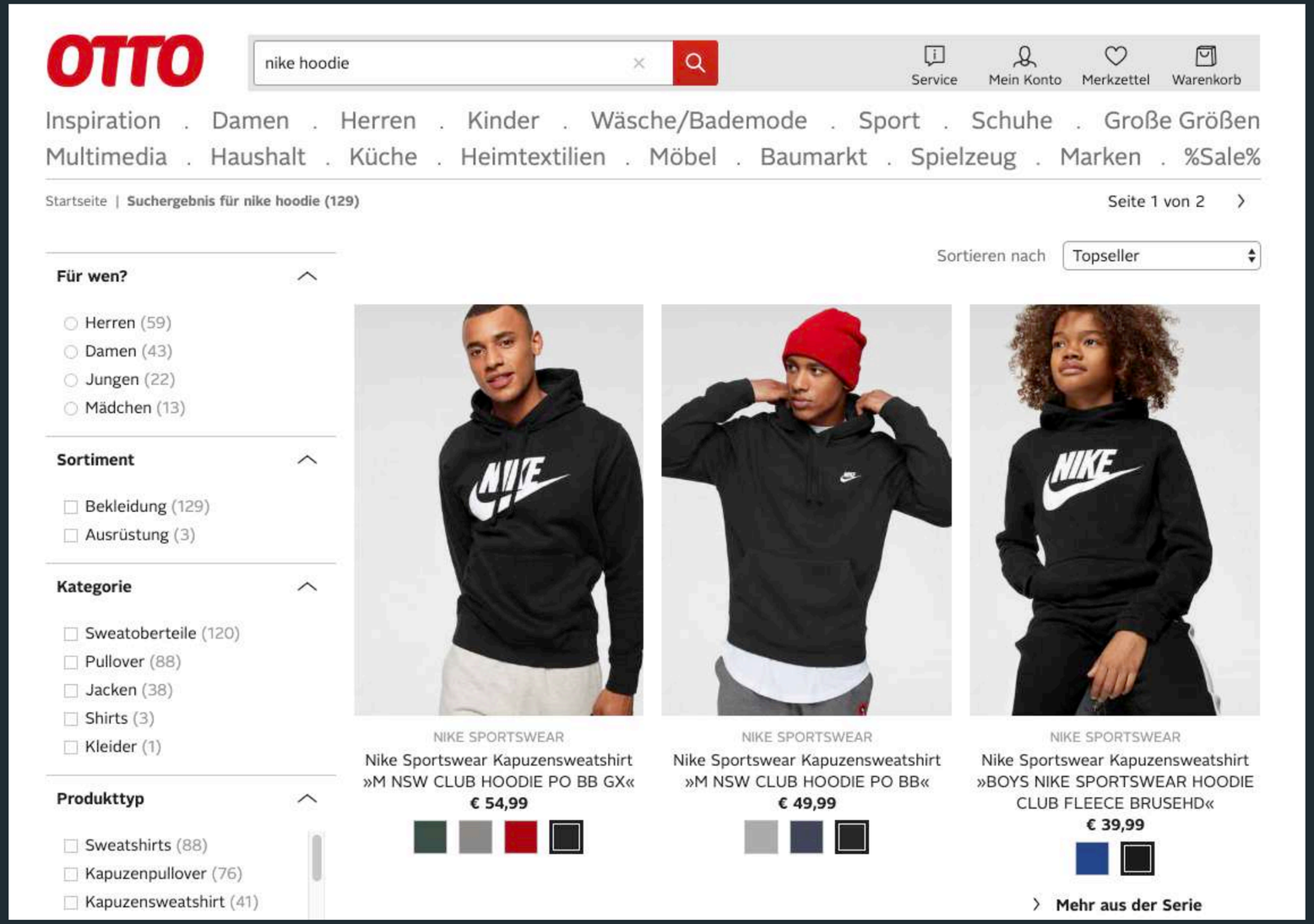
5 | 4

elastic

Aggregations

Reducing data

Aggregations



elastic

OTTO

nike hoodie

Service | Mein Konto | Merkzettel | Warenkorb

Inspiration . Damen . Herren . Kinder . Wäsche/Bademode . Sport . Schuhe . Große Größen
Multimedia . Haushalt . Küche . Heimtextilien . Möbel . Baumarkt . Spielzeug . Marken . %Sale%

Startseite | **Suchergebnis für nike hoodie (129)**

Seite 1 von 2 ›

aggregations

documents

Sortieren nach Topseller

Für wen?

- Herren (59)
- Damen (43)
- Jungen (22)
- Mädchen (13)

Sortiment

- Bekleidung (129)
- Ausrüstung (3)

Kategorie

- Sweatoberteile (120)
- Pullover (88)
- Jacken (38)
- Shirts (3)
- Kleider (1)

Produkttyp

- Sweatshirts (88)
- Kapuzenpullover (76)
- Kapuzensweatshirt (4)

NIKE SPORTSWEAR

Nike Sportswear Kapuzensweatshirt »M NSW CLUB HOODIE PO BB GX«

€ 54,99

NIKE SPORTSWEAR

Nike Sportswear Kapuzensweatshirt »M NSW CLUB HOODIE PO BB«

€ 49,99

NIKE SPORTSWEAR

Nike Sportswear Kapuzensweatshirt »BOYS NIKE SPORTSWEAR HOODIE CLUB FLEECE BRUSEHD«

€ 39,99

› **Mehr aus der Serie**

elast

OTTO

nike hoodie

Service | Mein Konto | Merkzettel | Warenkorb

Inspiration . Damen . Herren . Kinder . Wäsche/Bademode . Sport . Schuhe . Große Größen
Multimedia . Haushalt . Küche . Heimtextilien . Möbel . Baumarkt . Spielzeug . Marken . %Sale%

Startseite | **Suchergebnis für nike hoodie (129)**

Seite 1 von 2 ›

aggregations

documents

Sortieren nach Topseller

Für wen?

- Herren (59)
- Damen (43)
- Jungen (22)
- Mädchen (13)

Sortiment

- Bekleidung (129)
- Ausrüstung (3)

Kategorie

- Sweatoberteile (120)
- Pullover (88)
- Jacken (38)
- Shirts (3)
- Kleider (1)

Produkttyp

- Sweatshirts (88)
- Kapuzenpullover (76)
- Kapuzensweatshirt (41)

NIKE SPORTSWEAR

Nike Sportswear Kapuzensweatshirt »M NSW CLUB HOODIE PO BB GX«

€ 54,99

NIKE SPORTSWEAR

Nike Sportswear Kapuzensweatshirt »M NSW CLUB HOODIE PO BB«

€ 49,99

NIKE SPORTSWEAR

Nike Sportswear Kapuzensweatshirt »BOYS NIKE SPORTSWEAR HOODIE CLUB FLEECE BRUSEHD«

€ 39,99

› **Mehr aus der Serie**

elast

Bucketing documents

Pumps

Sneakers

Oxfords

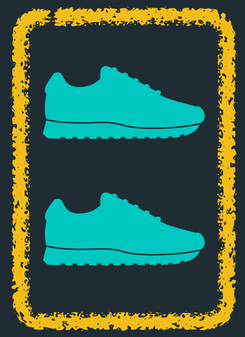
Sneakers

Boots

elastic

Bucketing documents

Sneakers



Pumps

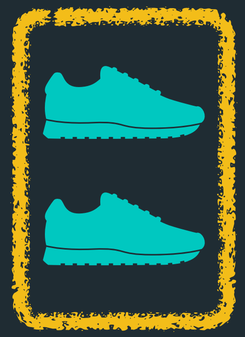
Boots

Oxfords

elastic

Bucketing documents

Sneakers



2

Pumps

1

Boots

1

Oxfords

1

bucket agg

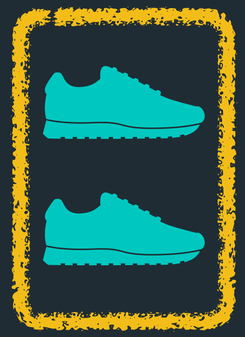
metric agg

doc_count

elastic

Bucketing documents

Sneakers



50

Pumps

95

Boots

90

Oxfords

23

bucket agg

metric agg

avg price

elastic

Aggregations

- bucket: terms, histogram, geo, range, sampler , significant text, nested
- metric: value_count, avg, min, max, sum, stats, median deviation, geo, percentile, cardinality,
- pipeline: min, max, sum, avg, derivative, stats, percentiles, cumulative sum, moving average, moving function, serial differencing

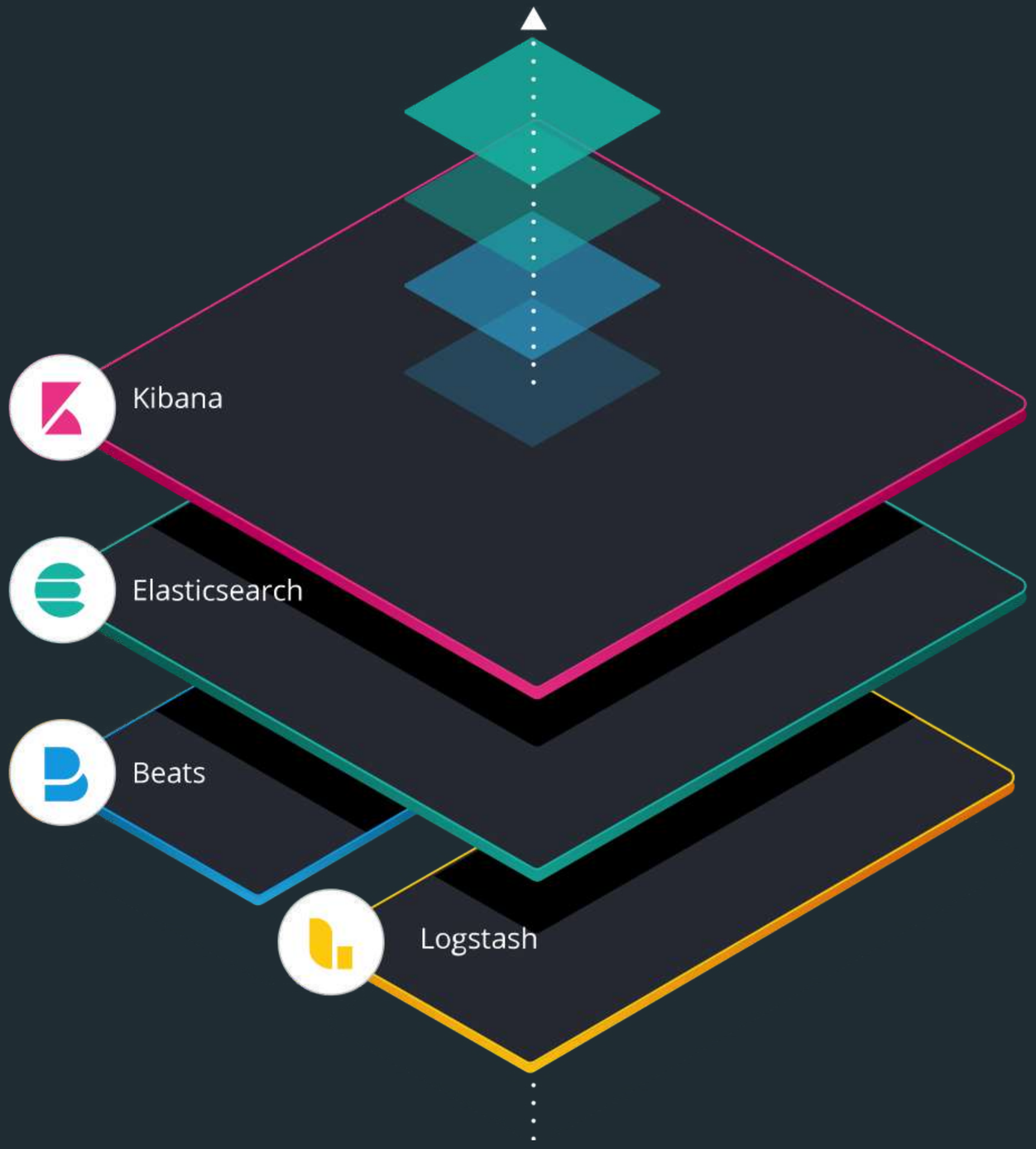
elastic

Distributed systems & search

Fanning out a search, reducing the results

Elasticsearch

elastic

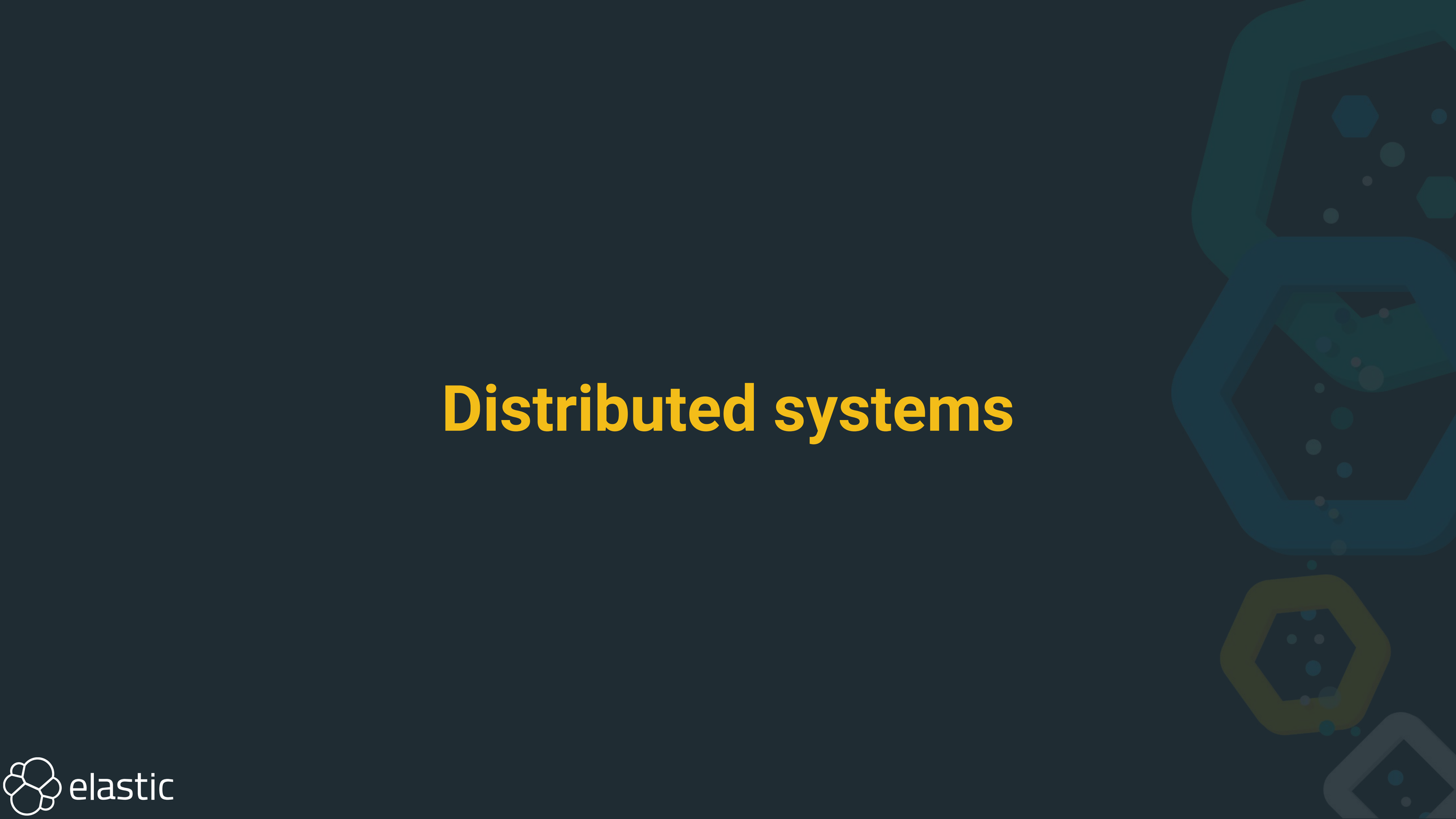


elastic

Elasticsearch in 10 seconds

- Search Engine (FTS, Analytics, Geo), near real-time
- Distributed, scalable, highly available, resilient
- Interface: HTTP & JSON
- Centrepiece of the Elastic Stack
- Uneducated conservative guess: Tens of thousands of clusters worldwide, hundreds of thousands of instances

Distributed systems



elastic

Distributed systems

- How do nodes communicate with each other?
- Who is taking and executing decisions?
- Failure detection?
- Replication strategy?
- Consistency?
- Enter consensus algorithms...

A fundamental problem in distributed computing and multi-agent systems is to achieve overall system reliability in the presence of a number of faulty processes. This often requires processes to agree on some data value that is needed during computation

https://en.wikipedia.org/wiki/Consensus_(computer_science)

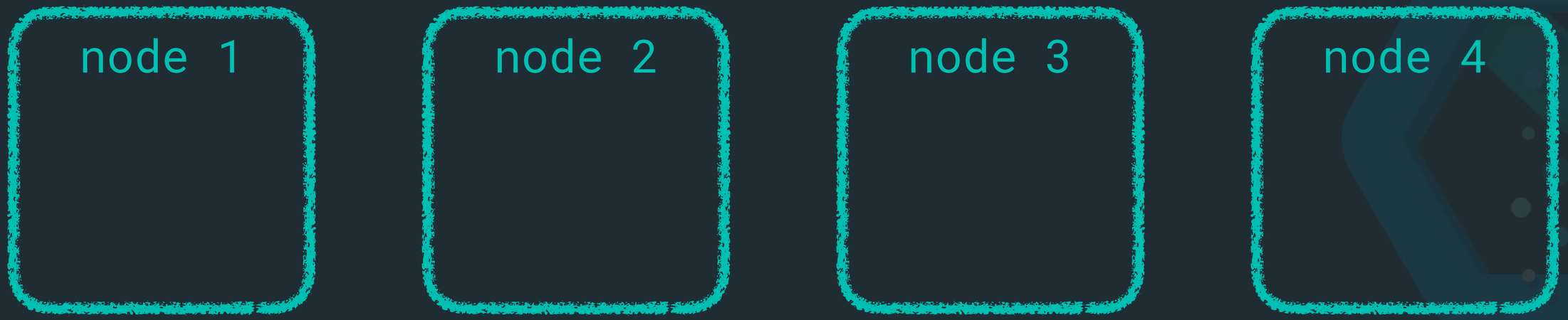
Consensus algorithms

- Leader based: Paxos, Raft
- Non leader based: BTC, gossip

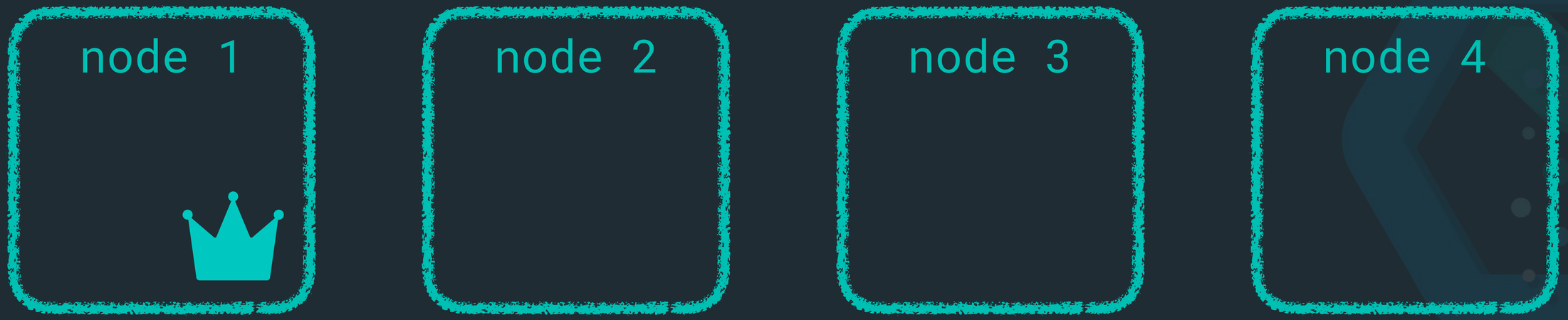
Consensus in Elasticsearch

- Custom consensus algorithm, improving the existing one
- Formally verified
- Optimized for Elasticsearch use-case (rolling restarts, growing/shrinking clusters, log-of-operations vs. cluster state)

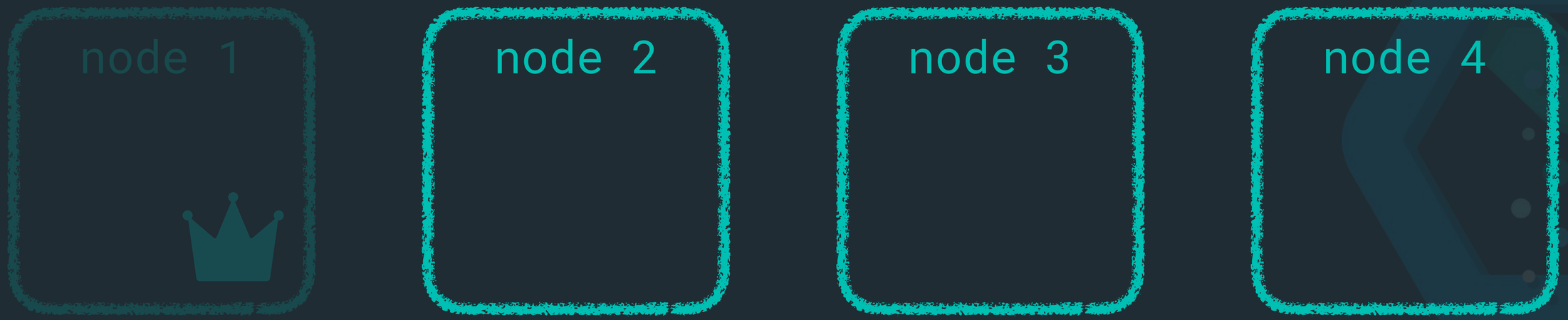
Consensus in Elasticsearch



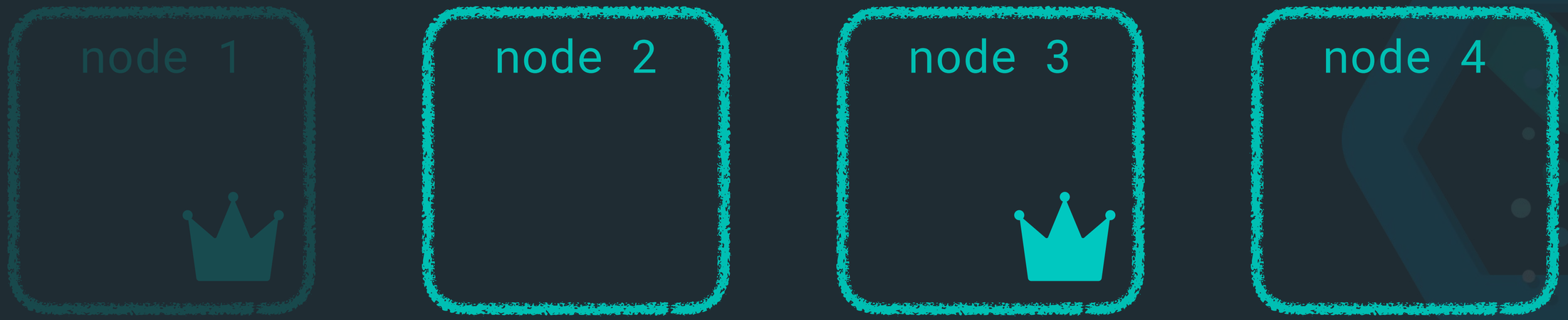
Consensus in Elasticsearch



Consensus in Elasticsearch



Consensus in Elasticsearch

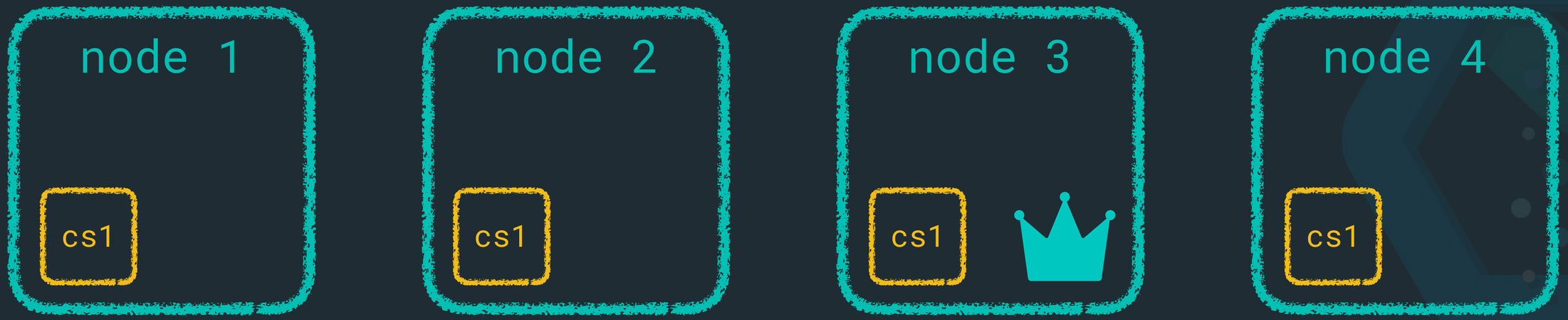


elastic

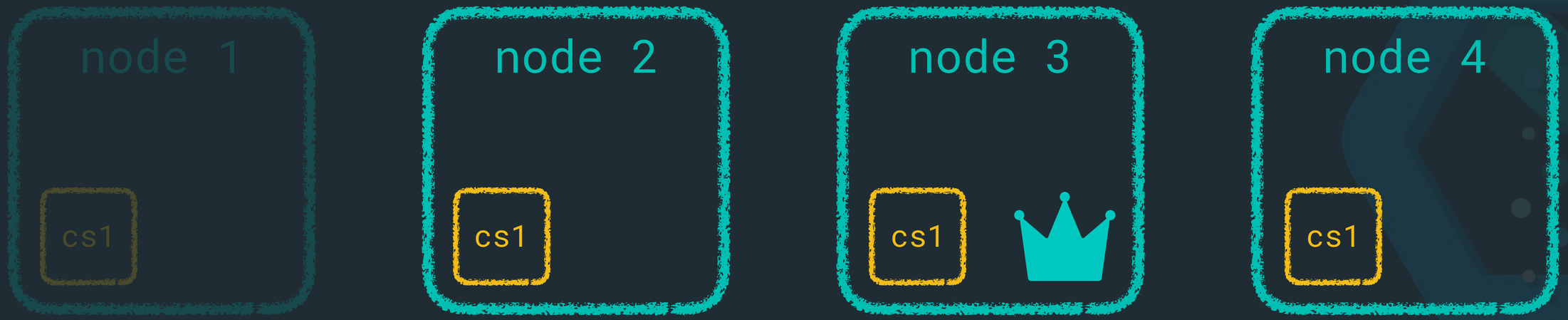
Master node tasks

- Deciding where data should be stored
- Pinging other nodes
- Reacting on node leaves/joins
- Updating cluster state
- Distributing cluster state

Consensus in Elasticsearch

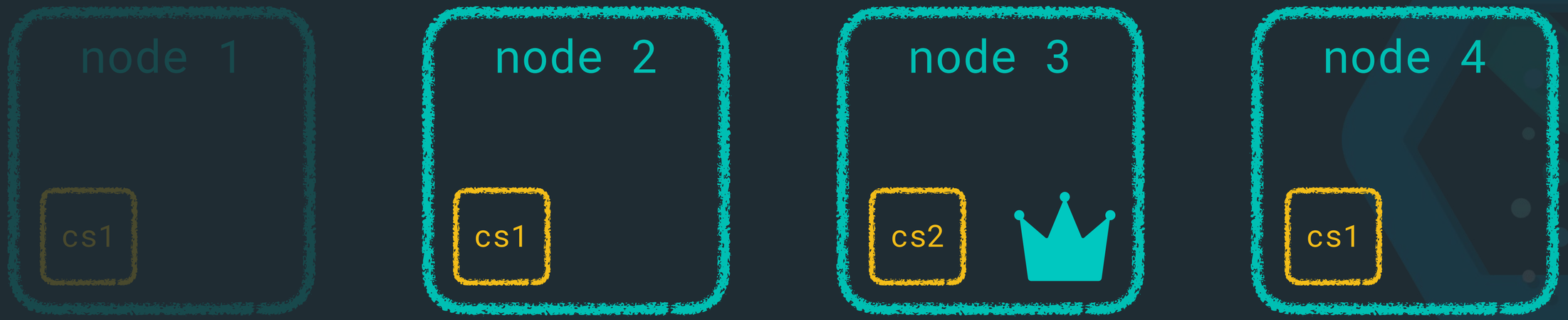


Consensus in Elasticsearch



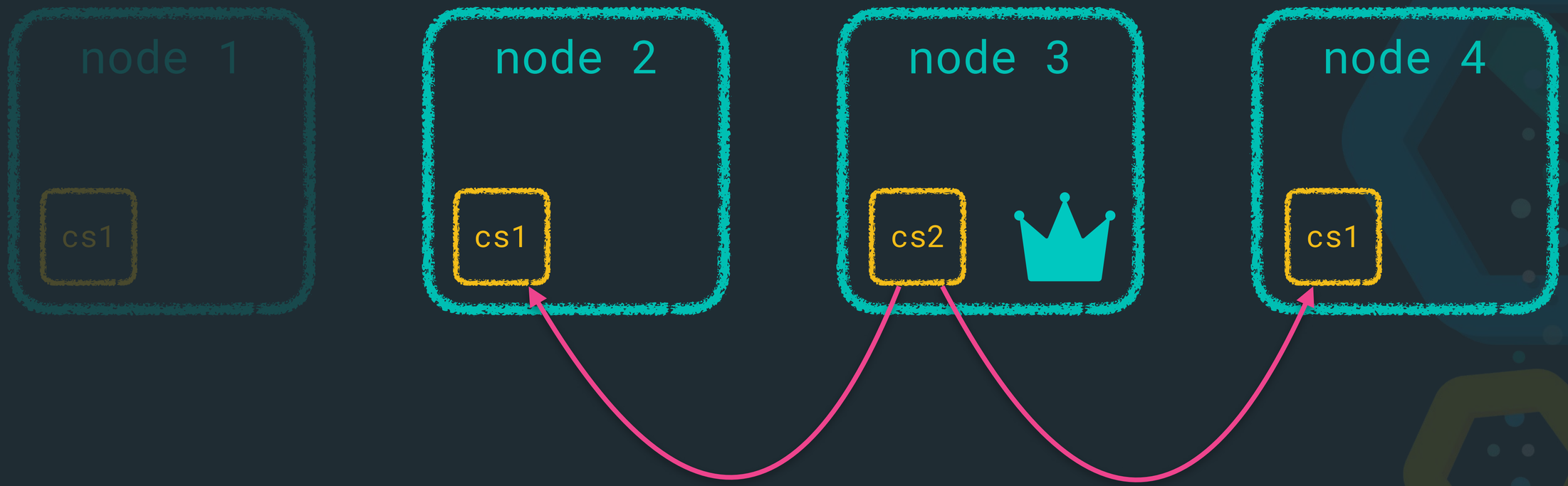
elastic

Consensus in Elasticsearch

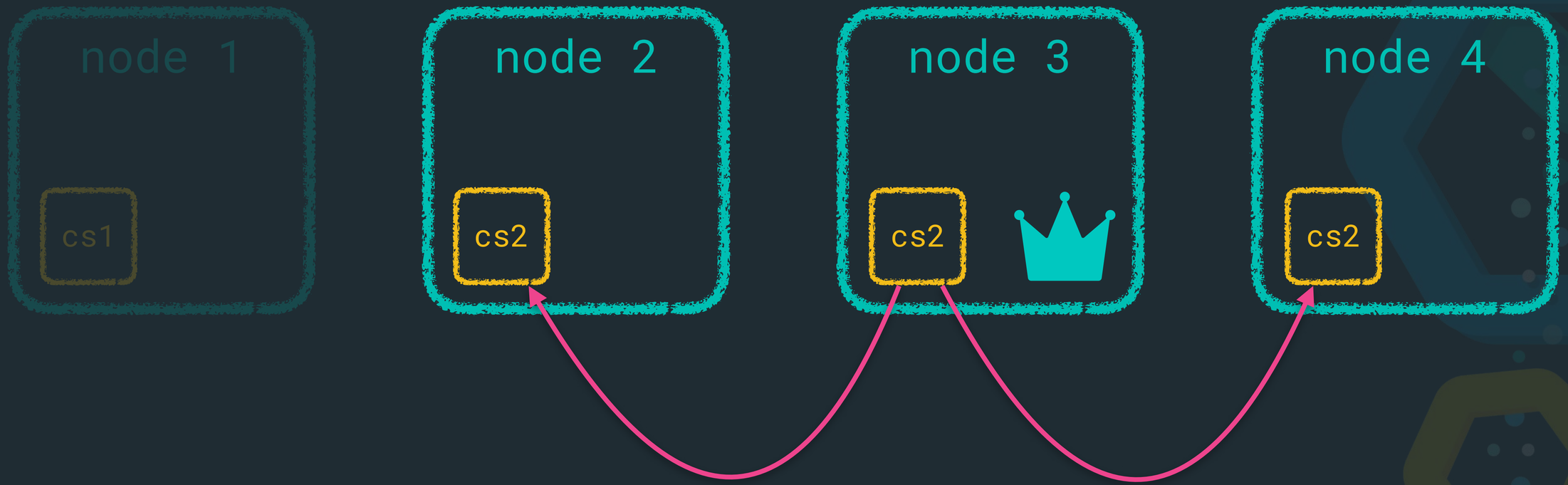


elastic

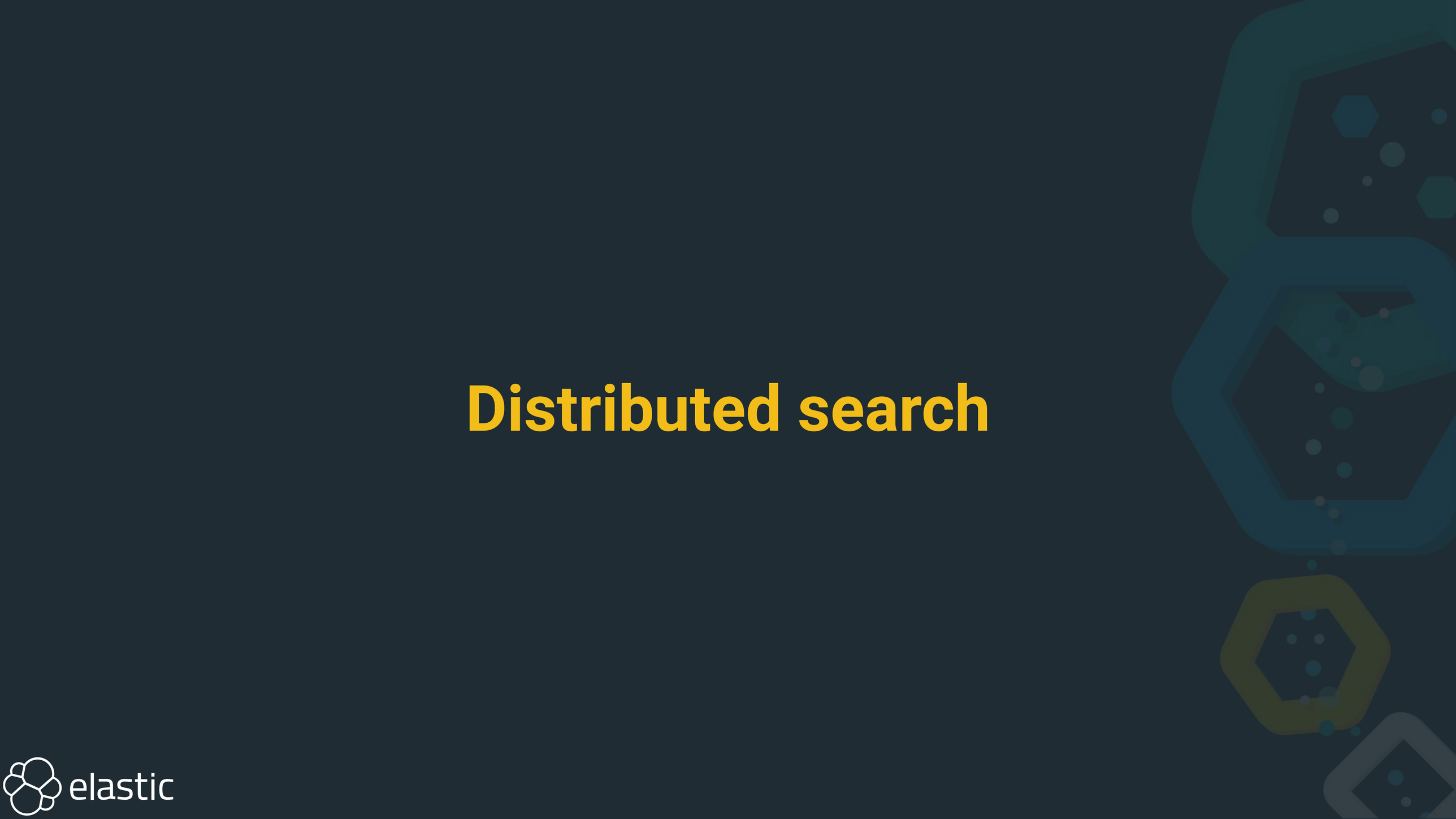
Consensus in Elasticsearch



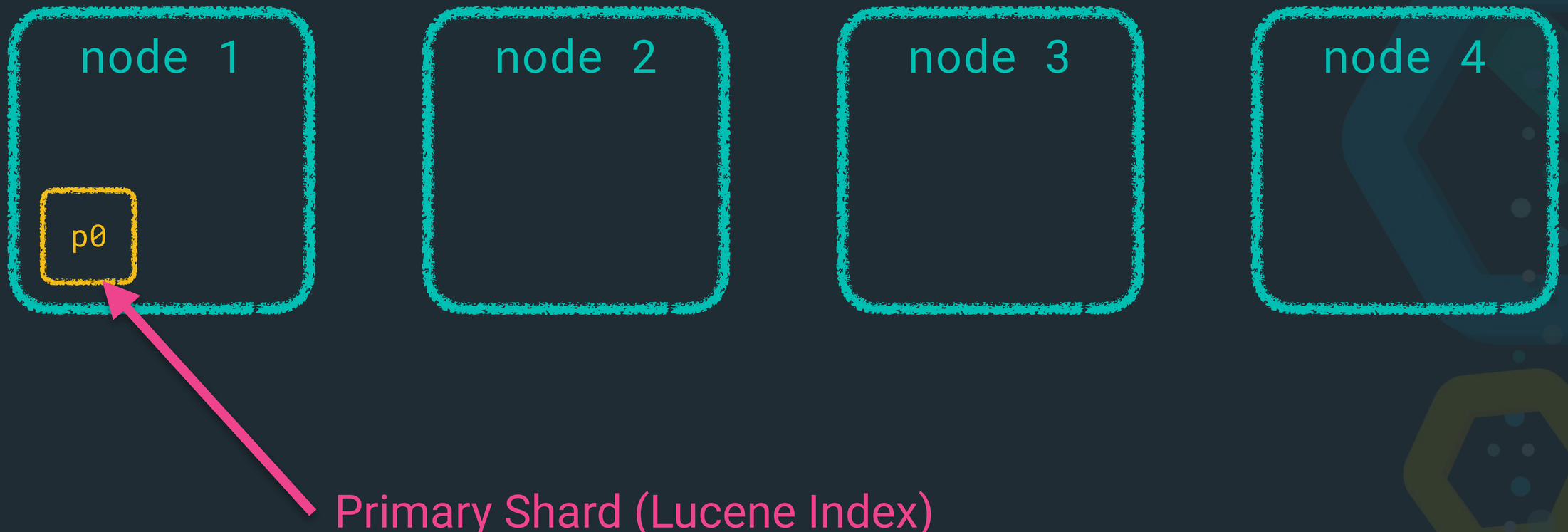
Consensus in Elasticsearch



Distributed search

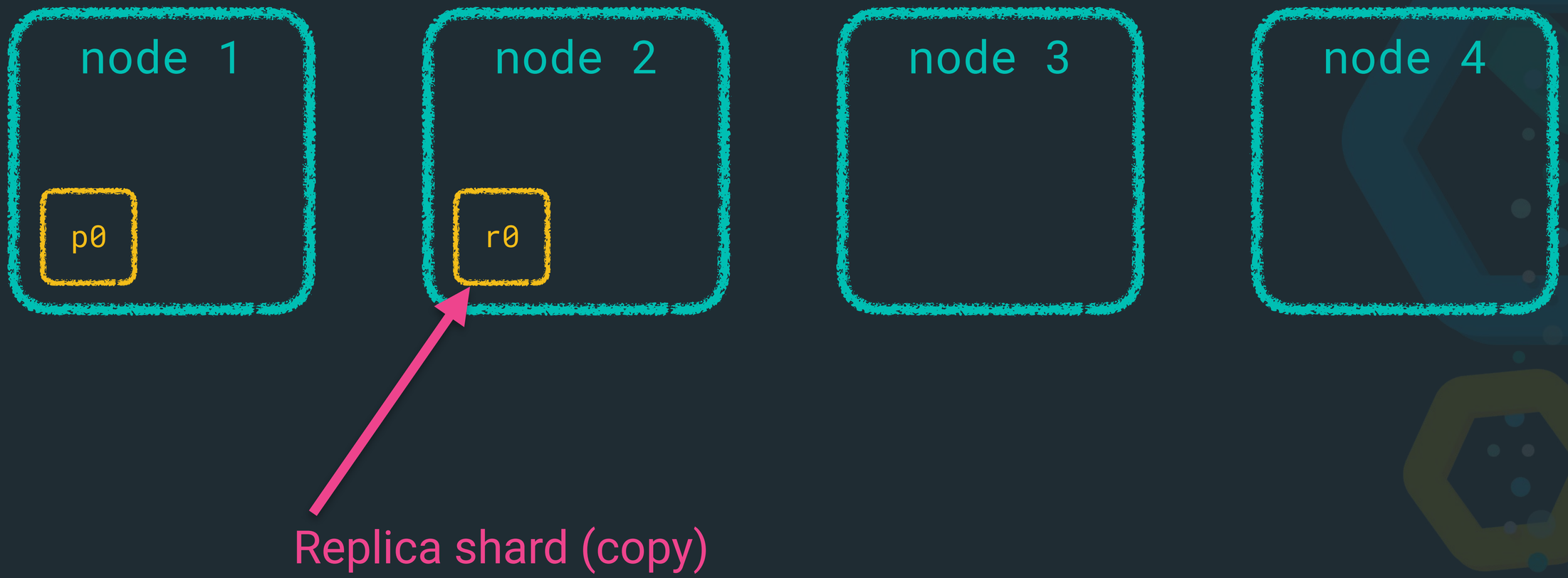


Distributed search in Elasticsearch

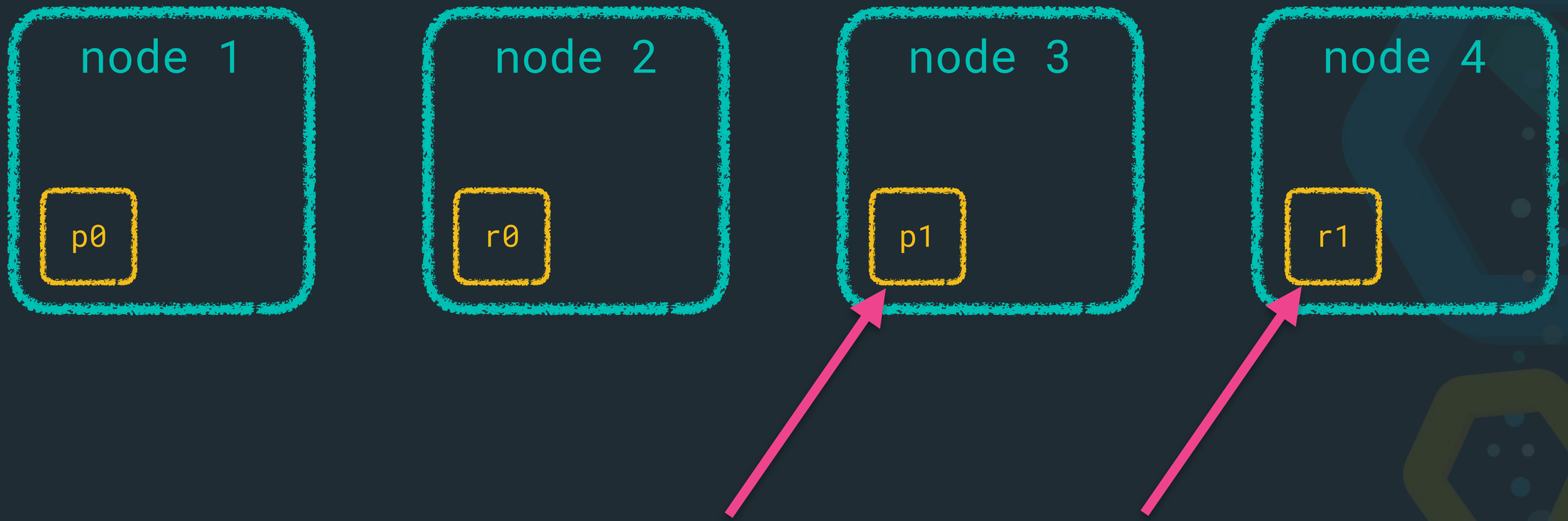


elastic

Distributed search in Elasticsearch



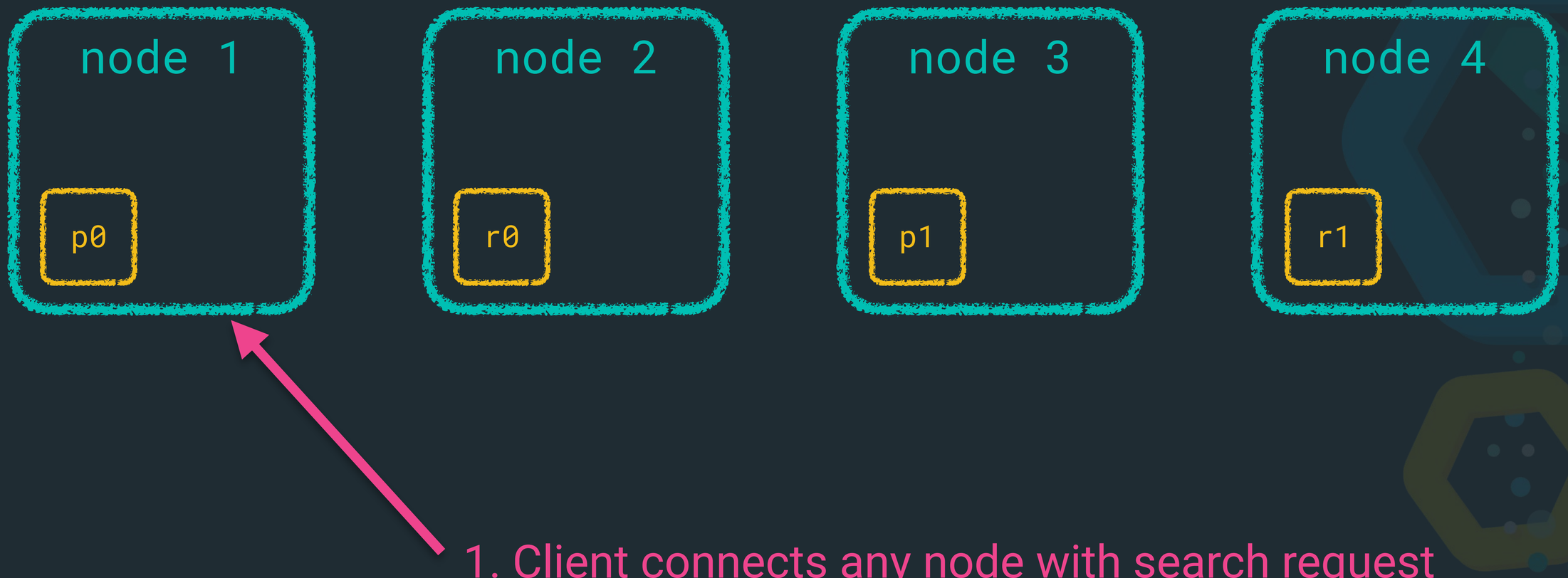
Distributed search in Elasticsearch



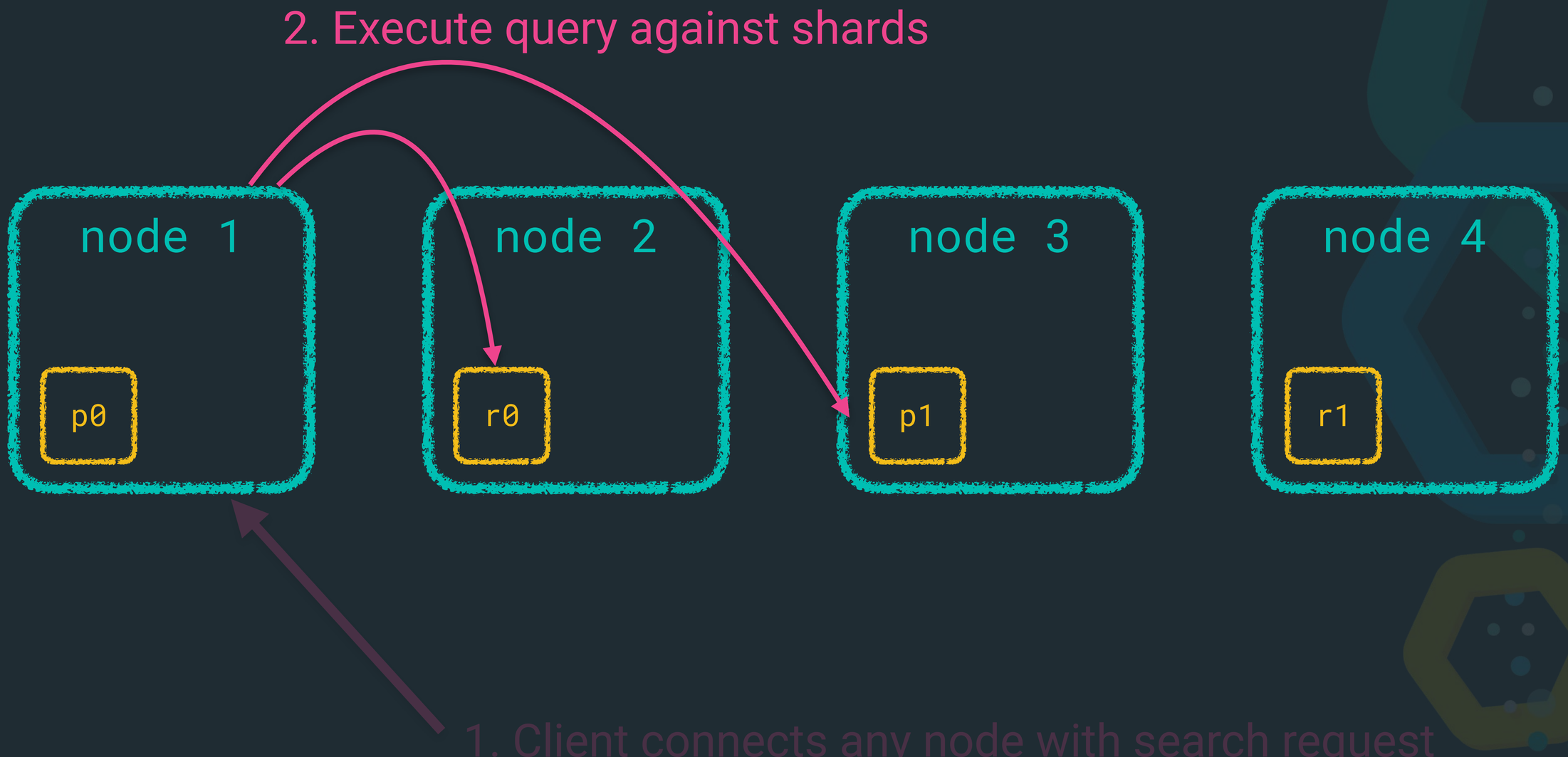
Distributed search in Elasticsearch

- Shard: Lucene index, unit of scale
- Primary shard: Write scalability
- Replica shard: Read scalability, availability

Distributed search in Elasticsearch

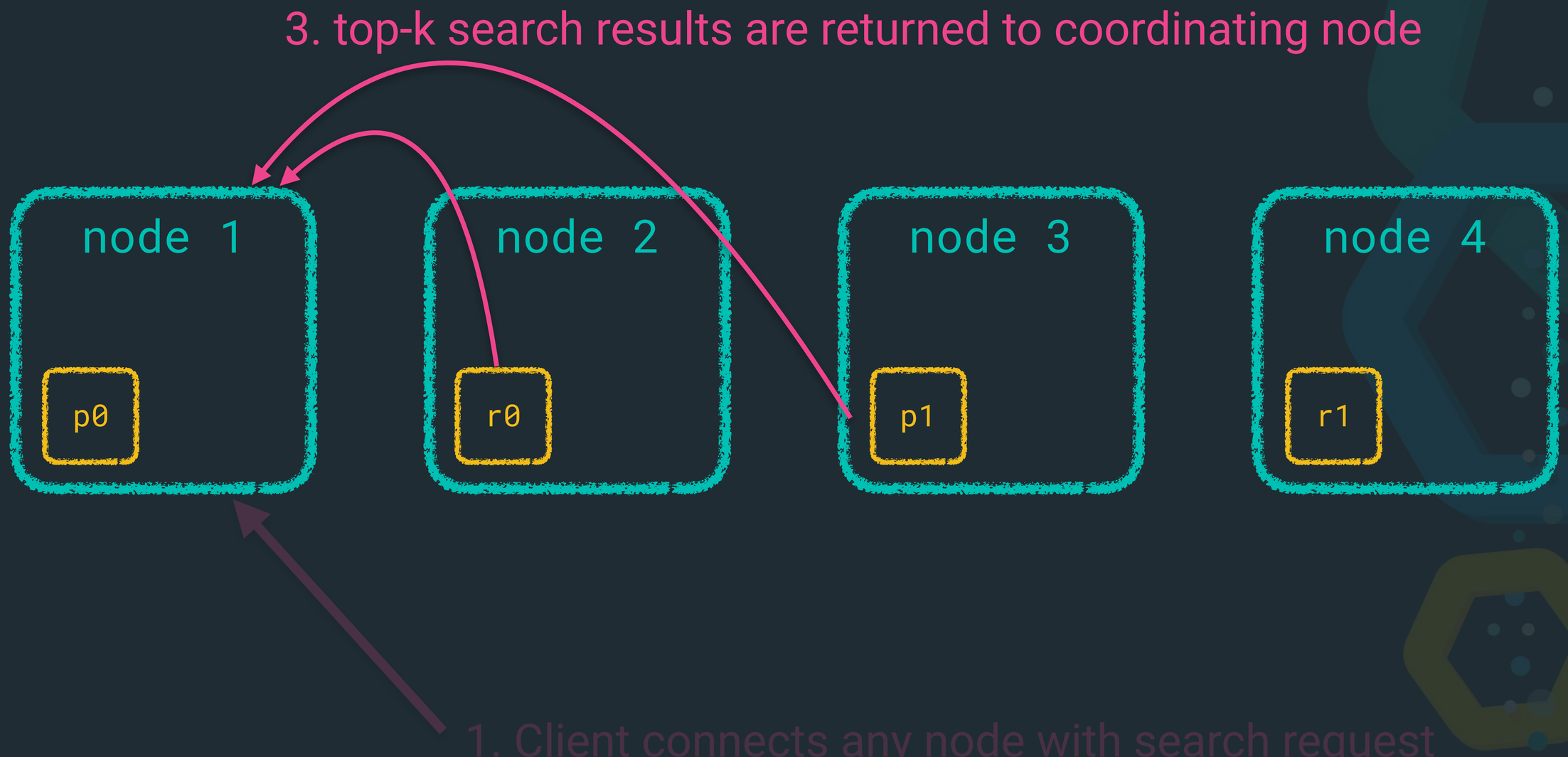


Distributed search in Elasticsearch



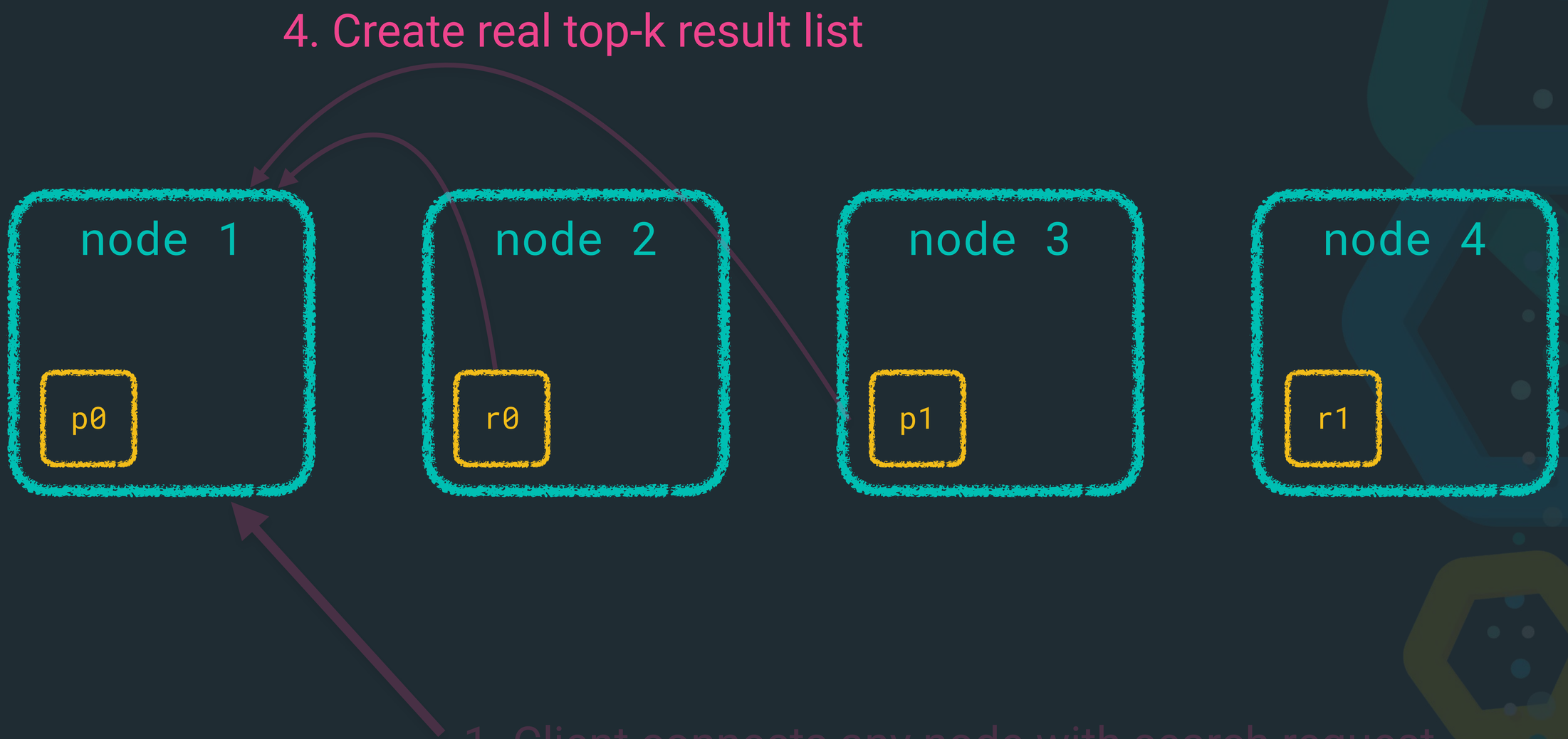
elastic

Distributed search in Elasticsearch



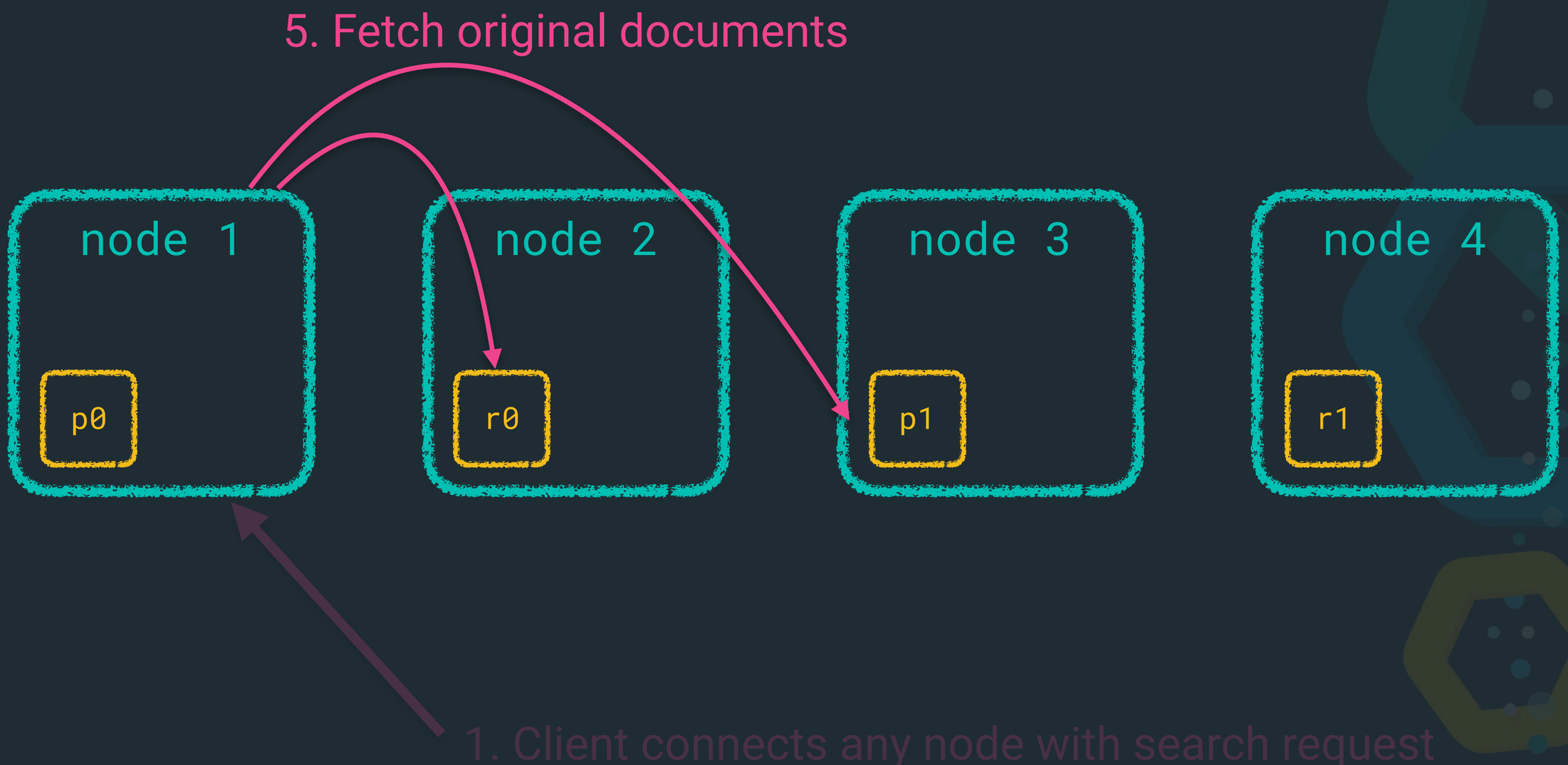
elastic

Distributed search in Elasticsearch



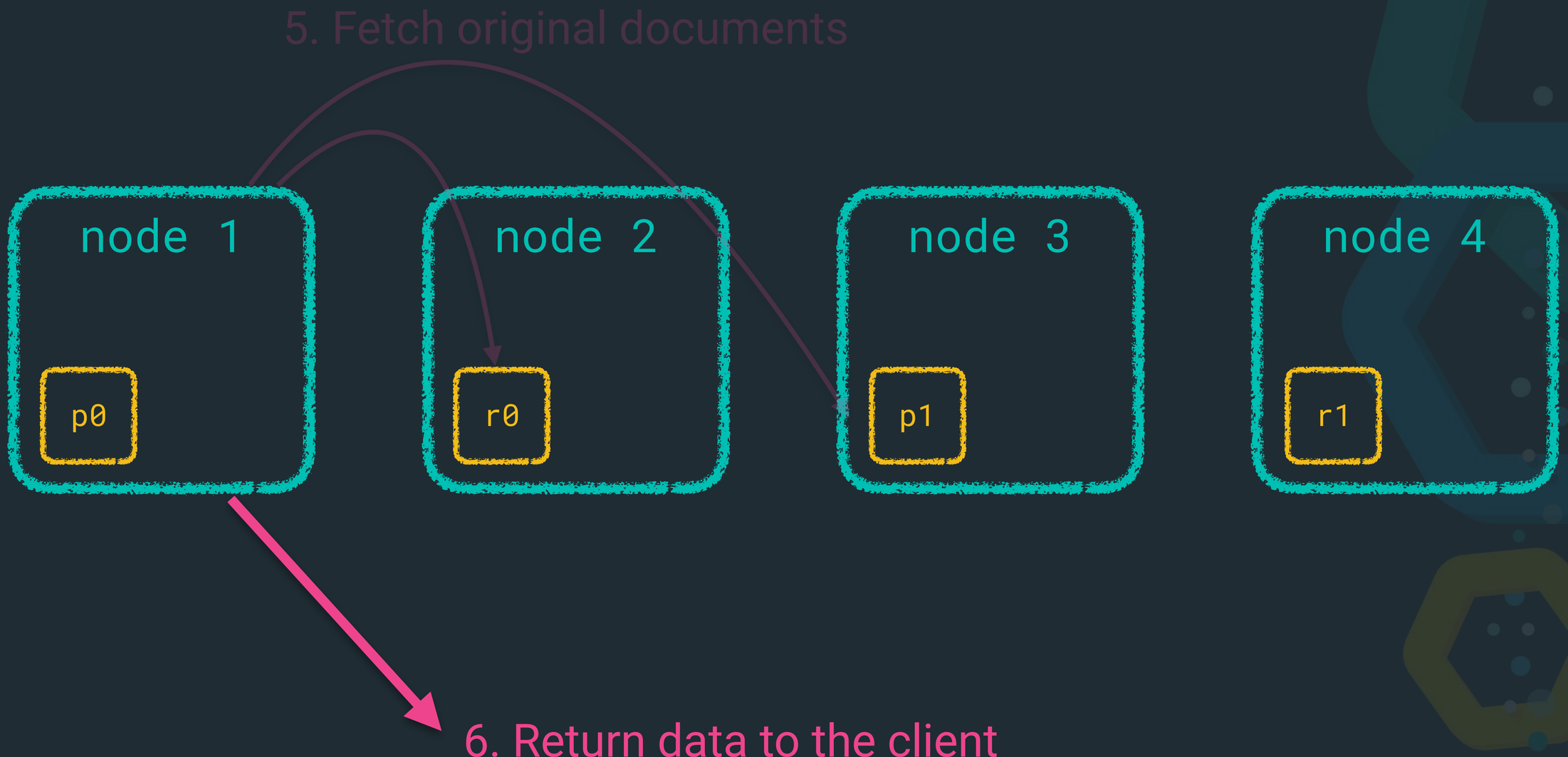
elastic

Distributed search in Elasticsearch



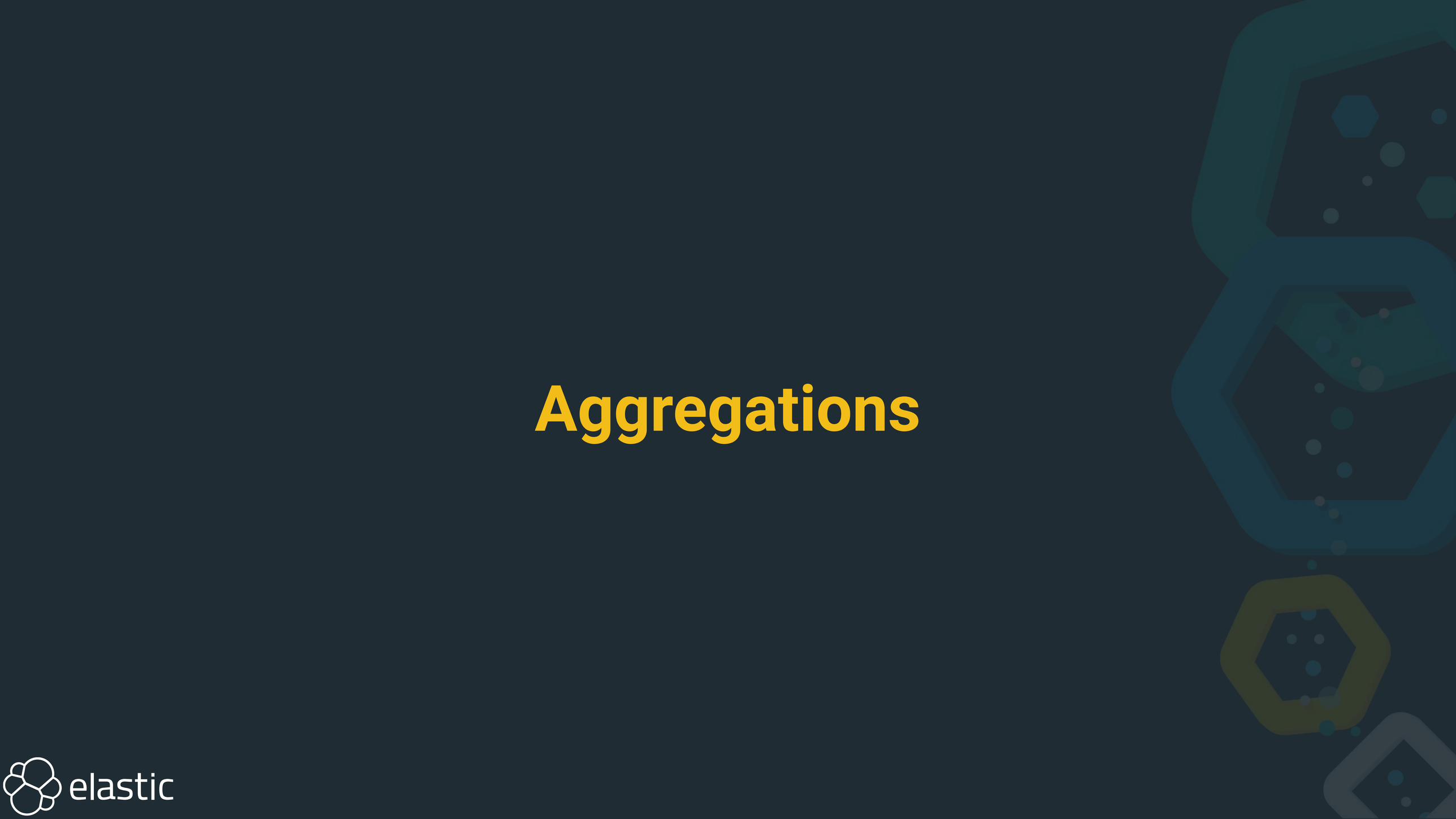
elastic

Distributed search in Elasticsearch



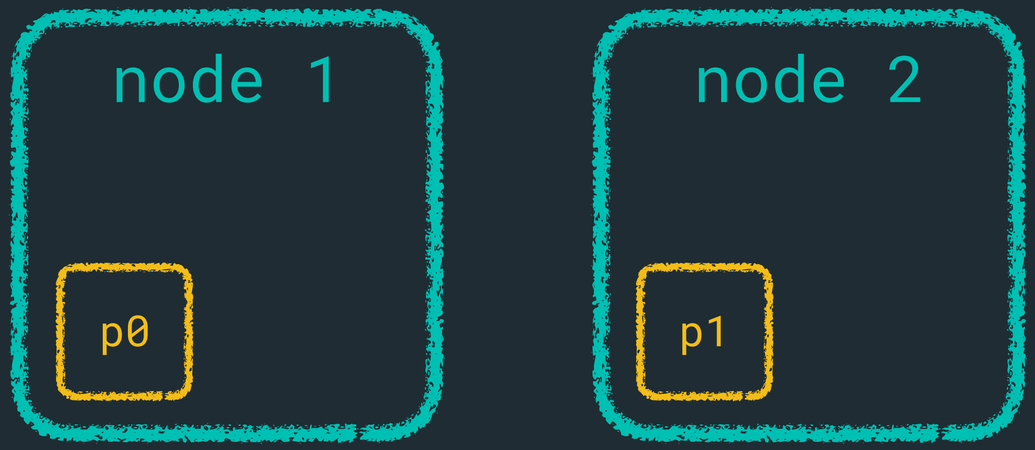
elastic

Aggregations



elastic

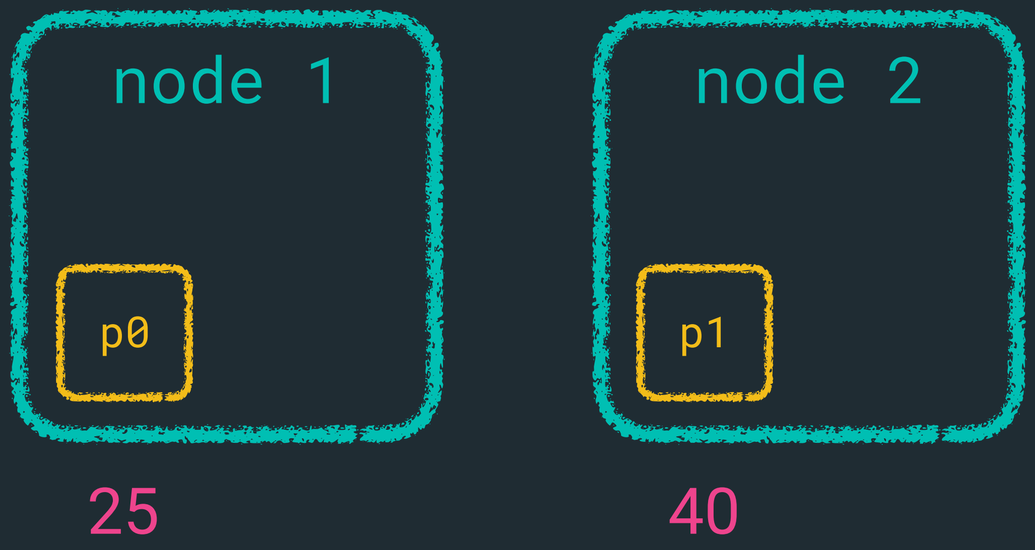
Aggregations - cardinality




```
POST /sales/_search?size=0
{
  "aggs" : {
    "type_count" : {
      "cardinality" : {
        "field" : "type"
      }
    }
  }
}
```

elastic

Aggregations - cardinality



How many distinct elements are in my index?

What is the total? 40? 65?

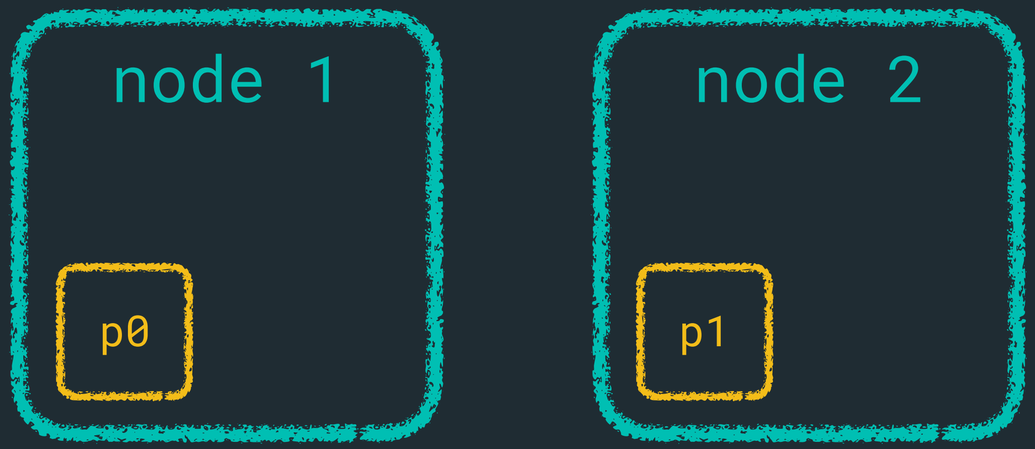
Naive solution: merge data to single dataset and count. Doesn't scale!

Solution: Use HyperLogLog++

HyperLogLog++

- Hash based counting
- Trades in memory for accuracy
- Fixed memory usage, based on configurable precision
- Result: Small mergeable data structure, can easily be sent over the network

Aggregations - percentile




```
GET latency/_search
{
  "size": 0,
  "aggs" : {
    "load_time_outlier" : {
      "percentiles" : {
        "field" : "load_time"
      }
    }
  }
}
```

elastic

T-Digest

- Extreme percentiles are more accurate than the Median
- Percentiles are divided into buckets
- When buckets grow over a boundary, approximation kicks in, saving memory in the process
- The exact level of inaccuracy is difficult to generalize
- Alternative: HDR histograms

Probabilistic data structures

- bloom/cuckoo/quotient filters (membership check)
- HyperLogLog++ (cardinality)
- T-Digest, DDSketch, HDR histogram (percentile)
- Count-Min sketch (frequency, top-k)
- Hashing (similarity)

elastic

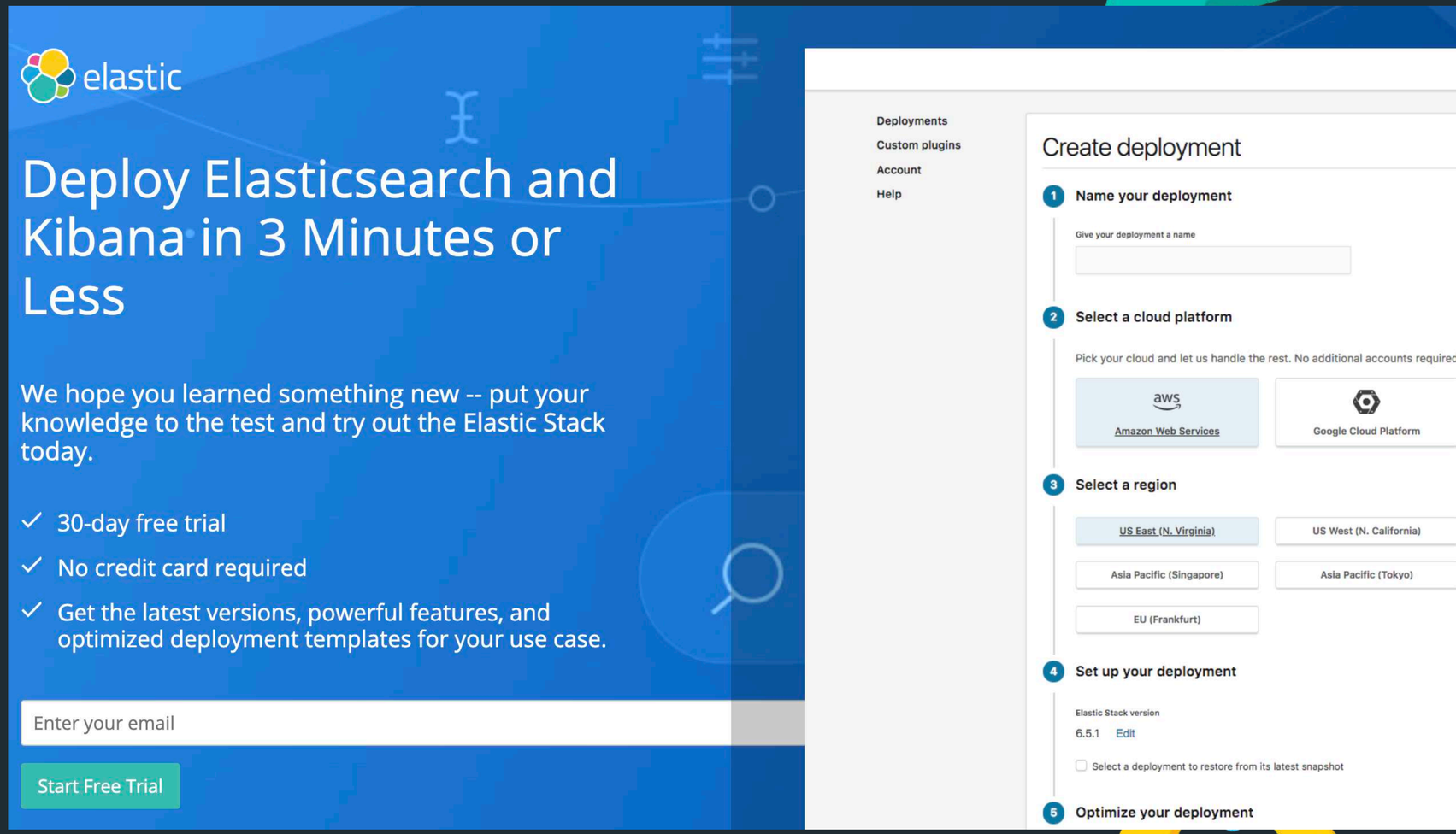
Demo

Try it out yourself!

https://ela.st/jade-hochschule-samples

elastic

Elastic Cloud
Free 30 day trial



https://ela.st/university-wilhelmshaven

elastic

Upcoming trends & summary

... or why you should take a closer look at search

Search is not just google...

- "Just google it" does not cut it
- Enterprise search: Intranet/G-Drive/Dropbox
- Ecommerce search
- SIEM
- Observability: Logging, APM & Metrics

Search is not 'done'

- Constant improvement
- Data structures & algorithms (BKD tree for geo shapes)
- Academic research moves to industry thanks to Apache Lucene

Search is still tough

- Language specific analysis
- Smart query parsing (`nike red hoodie xl`)
- Geo based search
- Anomaly detection
- Incoporating feedback loops

Upcoming trends

- Learning-to-Rank
- Deep Learning
- Feedback loop

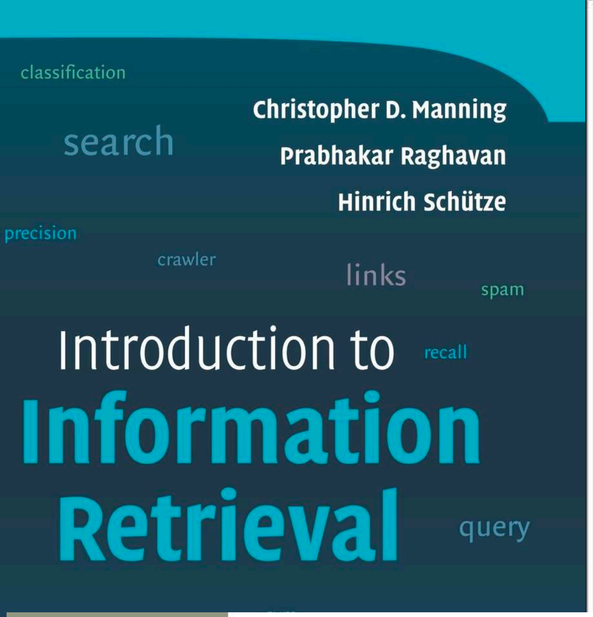
Summary

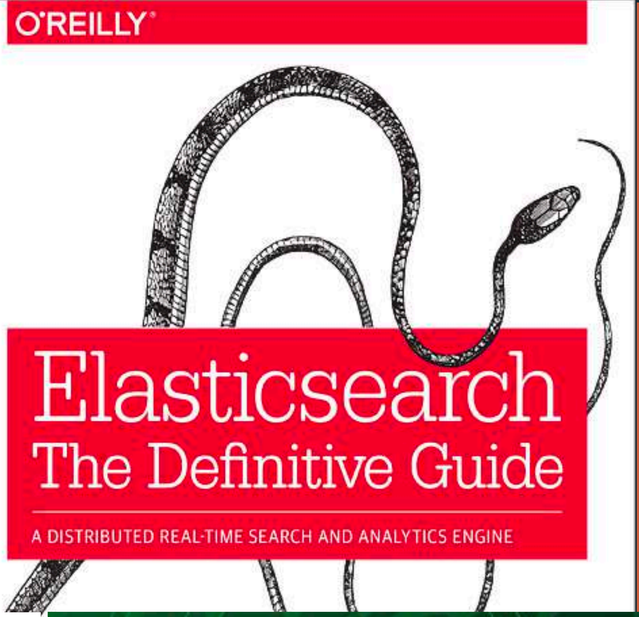
- Everything is a search problem!
- Search is hard... and interesting
- Distributed systems are hard... and interesting
- Domain knowledge required
- Data keeps exploding, good job chances!

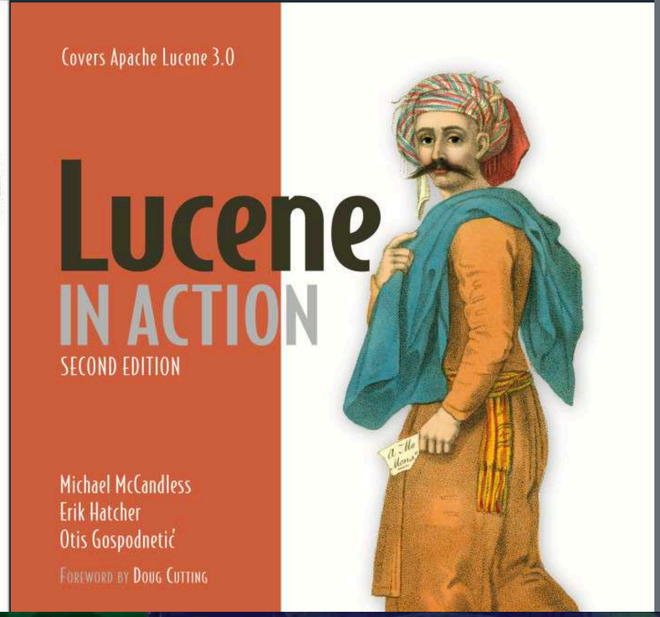
elastic

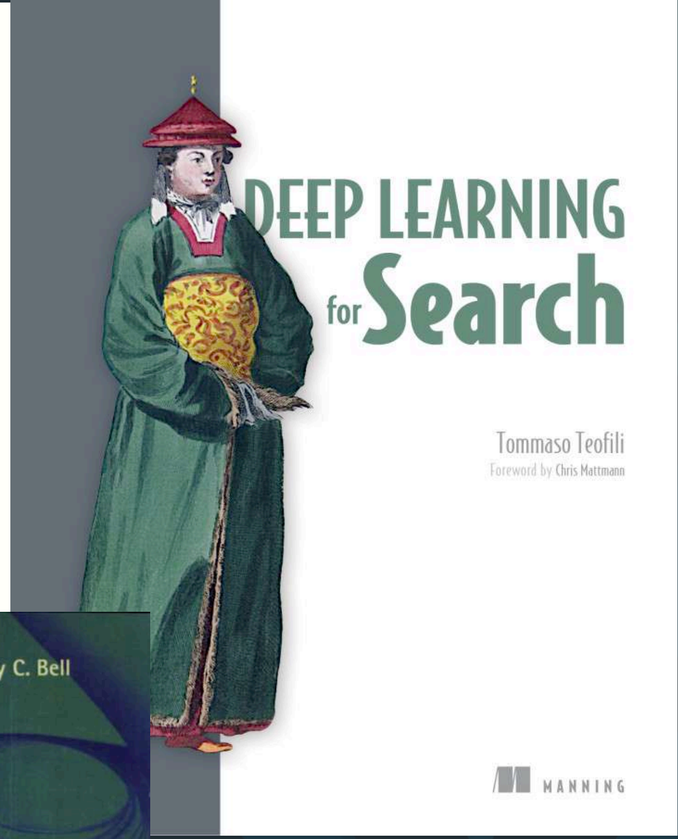
Literature

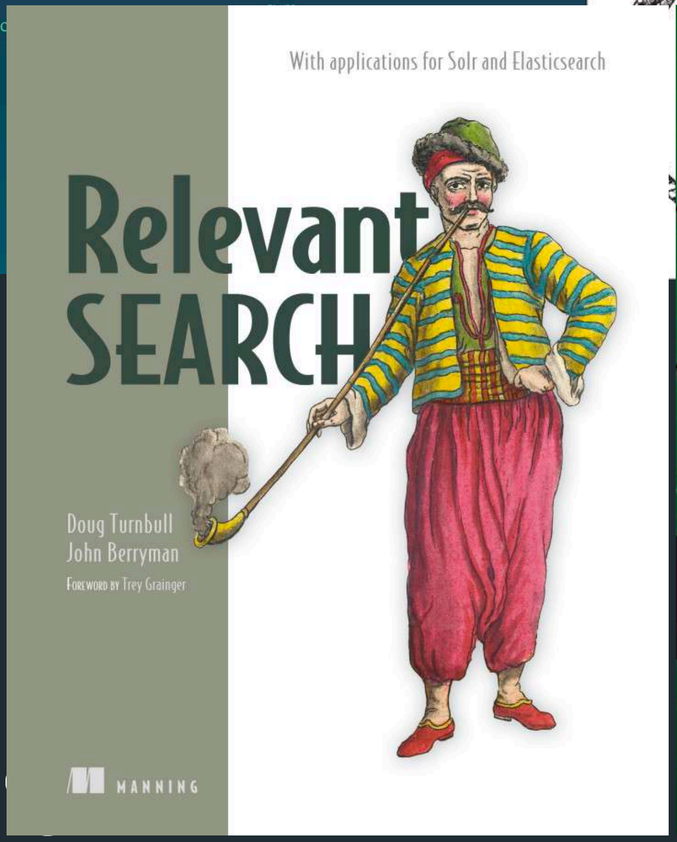
Books, books, books

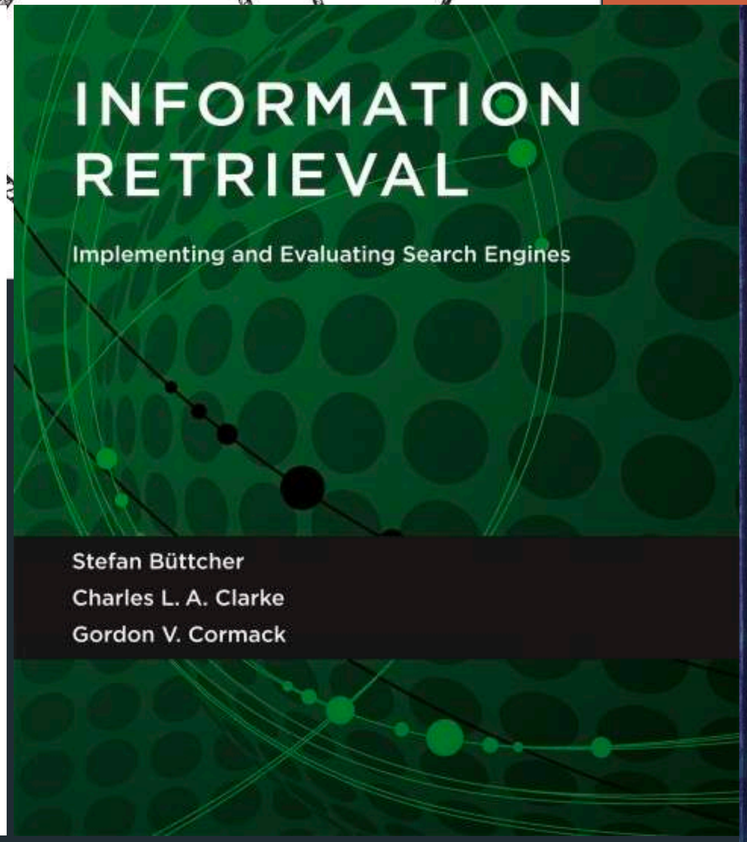

classification
search
Christopher D. Manning
Prabhakar Raghavan
Hinrich Schütze
precision
crawler
links
spam
Introduction to
recall
Information
Retrieval
query

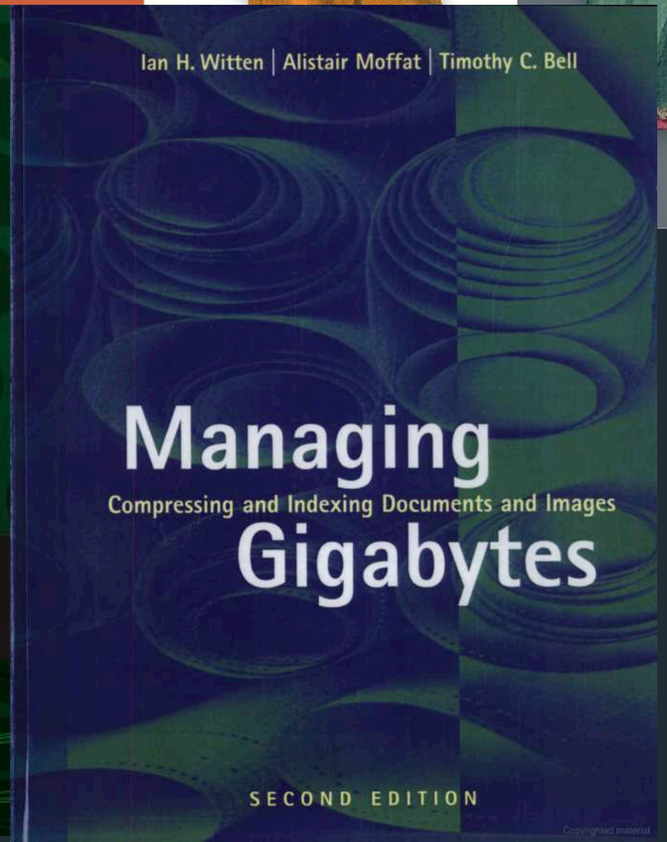

O'REILLY
Elasticsearch
The Definitive Guide
A DISTRIBUTED REAL-TIME SEARCH AND ANALYTICS ENGINE


Covers Apache Lucene 3.0
Lucene
IN ACTION
SECOND EDITION
Michael McCandless
Erik Hatcher
Otis Gospodnetić
Foreword by Doug Cutting


DEEP LEARNING
for Search
Tommaso Teofili
Foreword by Chris Mattmann
MANNING


With applications for Solr and Elasticsearch
Relevant
SEARCH
Doug Turnbull
John Berryman
Foreword by Trey Grainger
MANNING


INFORMATION
RETRIEVAL
Implementing and Evaluating Search Engines
Stefan Büttcher
Charles L. A. Clarke
Gordon V. Cormack


Ian H. Witten | Alistair Moffat | Timothy C. Bell
Managing
Compressing and Indexing Documents and Images
Gigabytes
SECOND EDITION

elastic

Resources

Links, links, links

Links

https://lucene.apache.org/core/8_2_0/core/org/apache/lucene/search/similarities/TFIDFSimilarity.html

https://www.elastic.co/blog/whats-new-in-lucene-8

https://www.elastic.co/blog/faster-retrieval-of-top-hits-in-elasticsearch-with-block-max-wand

https://speakerdeck.com/elastic/amusing-algorithms-and-data-structures

https://www.elastic.co/blog/index-sorting-elasticsearch-6-0

https://raft.github.io/

https://github.com/elastic/elasticsearch-formal-models

https://gist.github.com/spinscale/b62c8b357fae7db3f14b7d3127758951

Links - probabilistic data structures

https://github.com/addthis/stream-lib

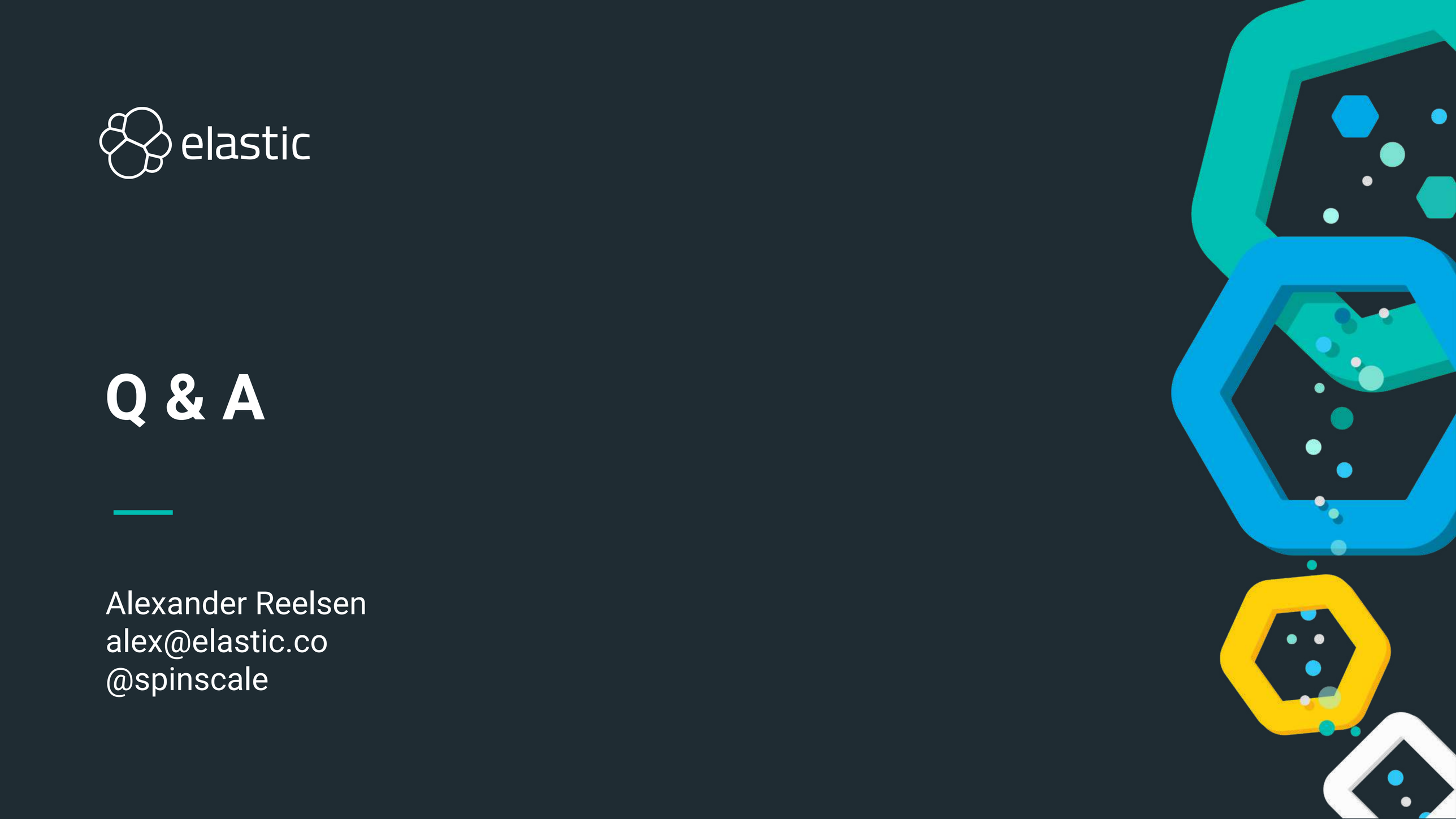
https://github.com/DataDog/sketches-java

https://github.com/HdrHistogram/HdrHistogram

https://github.com/JohnStarich/java-skip-list

https://github.com/addthis/stream-lib

https://static.googleusercontent.com/media/research.google.com/fr/pubs/archive/40671.pdf



Q & A

Alexander Reelsen
alex@elastic.co
@spinscale