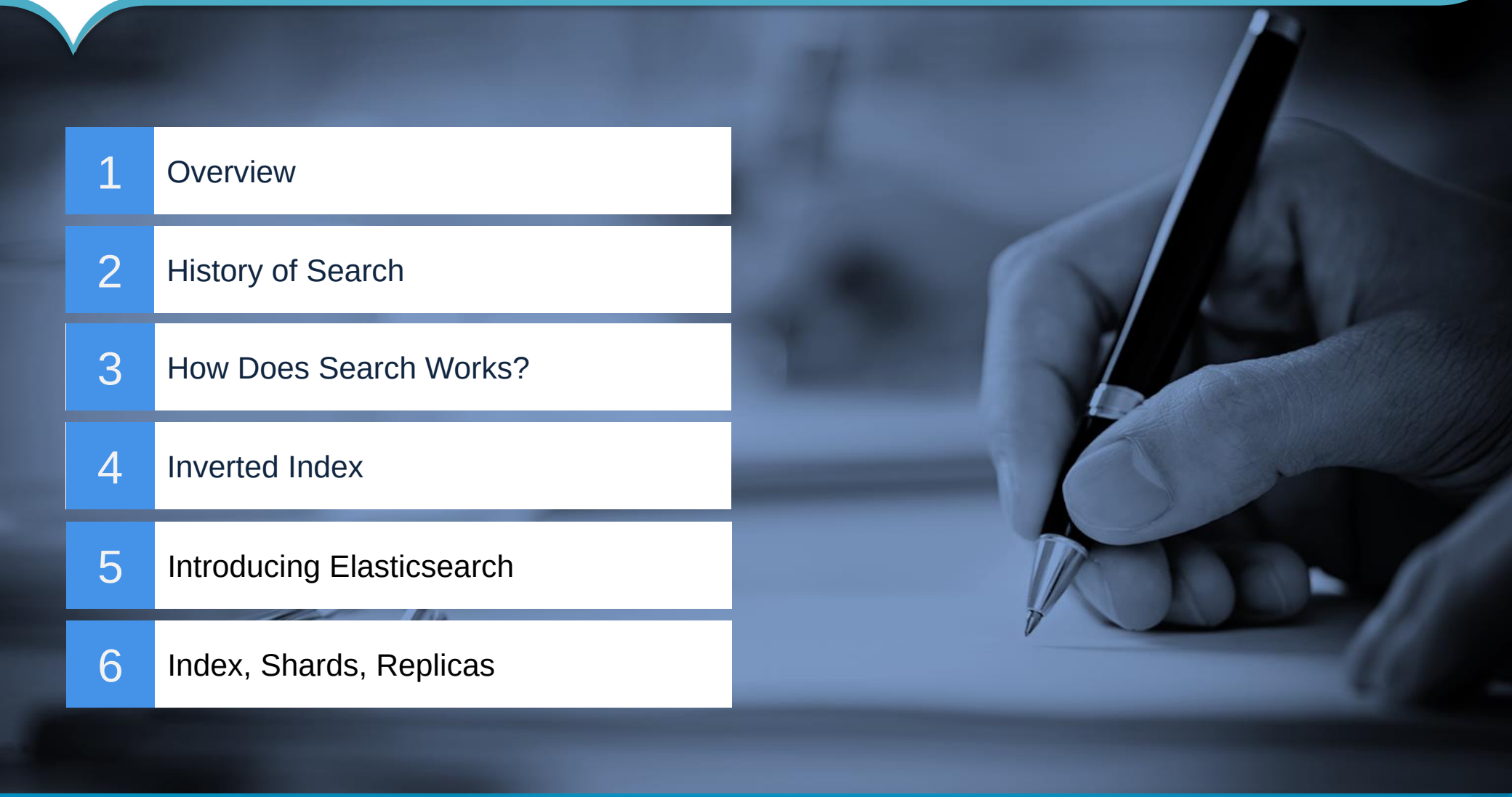
Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

ElasticSearch

May 2017

People matter, results count.

Course Map - Searching and Analyzing Data with Elasticsearch



1. Overview
2. History of Search
3. How Does Search Works?
4. Inverted Index
5. Introducing Elasticsearch
6. Index, Shards, Replicas

Overview

- A little search engine history and the importance of search
- Basics steps involved in indexing and searching documents
- The inverted index, the heart of a search engine
- An introduction to Elasticsearch and its basic building blocks
- Set up and install Elasticsearch on your local machine and check cluster health

Overview

Prerequisites

- Familiarity with the command line on a Mac, Linux or Windows machine
- Familiarity with using RESTful APIs to perform actions
- A very basic understanding of distributed computing

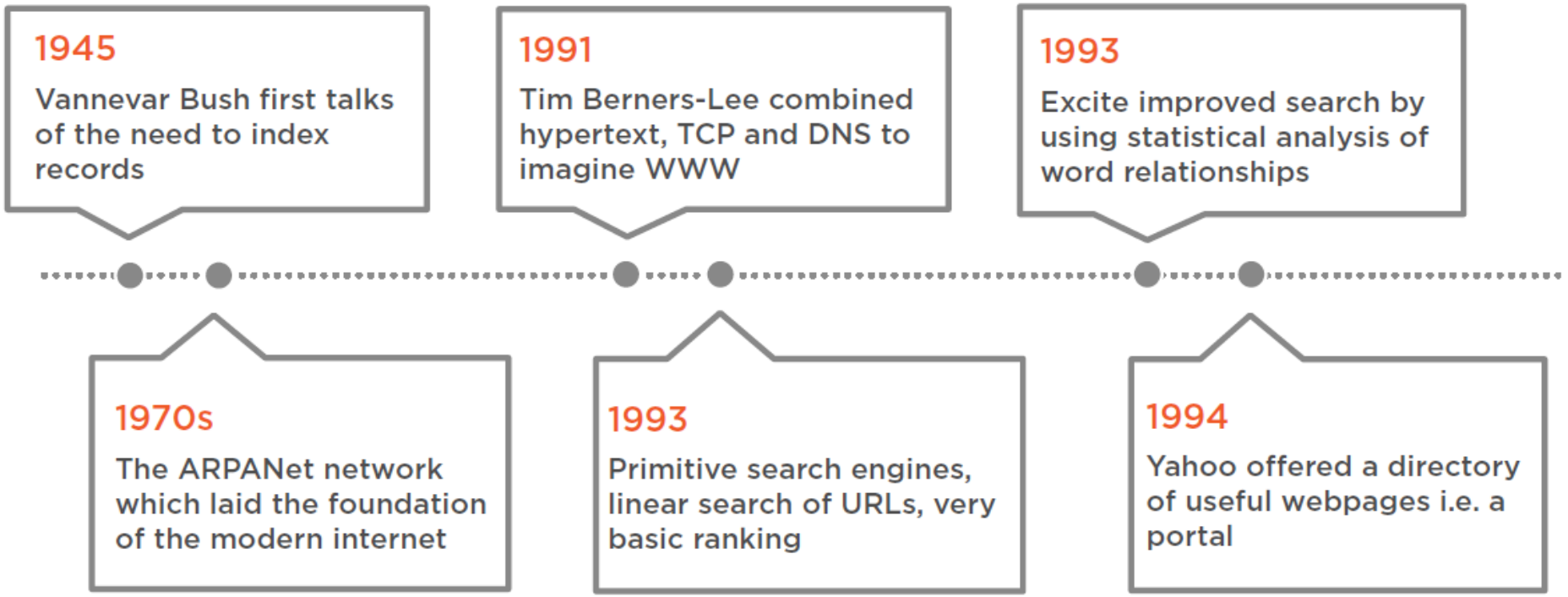
Install and Setup

- The latest version of Elasticsearch, 5.4.0 requires Java version 8
- A Mac, Linux or Windows machine on which Elasticsearch can be installed

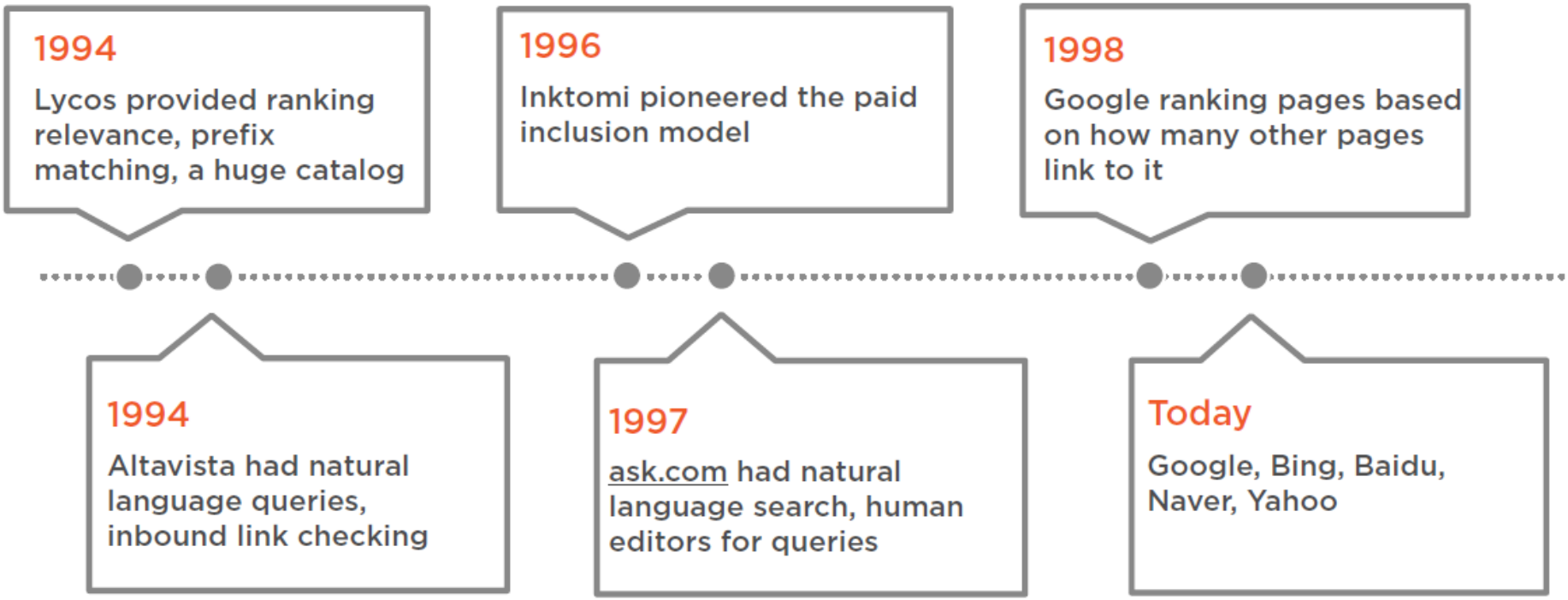
Course Overview

- Introduction to basic concepts in Elasticsearch, download and install
- Building an index, adding documents to it both individually and in bulk
- Search queries on an index using the Query DSL
- Analysis of data on an index using aggregations

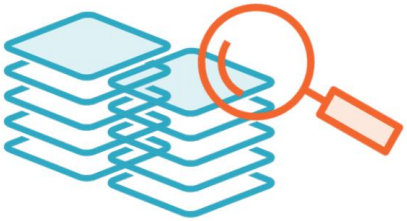
Brief History of Search



Brief History of Search



How Does Search Work?



- **What Is the Objective of Search?**
  - Find the most relevant documents with your search terms

Most Relevant Document for Search Terms:

Know of the document's existence

Index the document for lookup

Know how relevant the document is

Retrieve ranked by relevance

How Does Search Work?

Most Relevant Document for Search Terms

Web crawler

Inverted index

Scoring

Search

How Does Search Work?

Search is not restricted to the Web

Sites Have Their Own Search

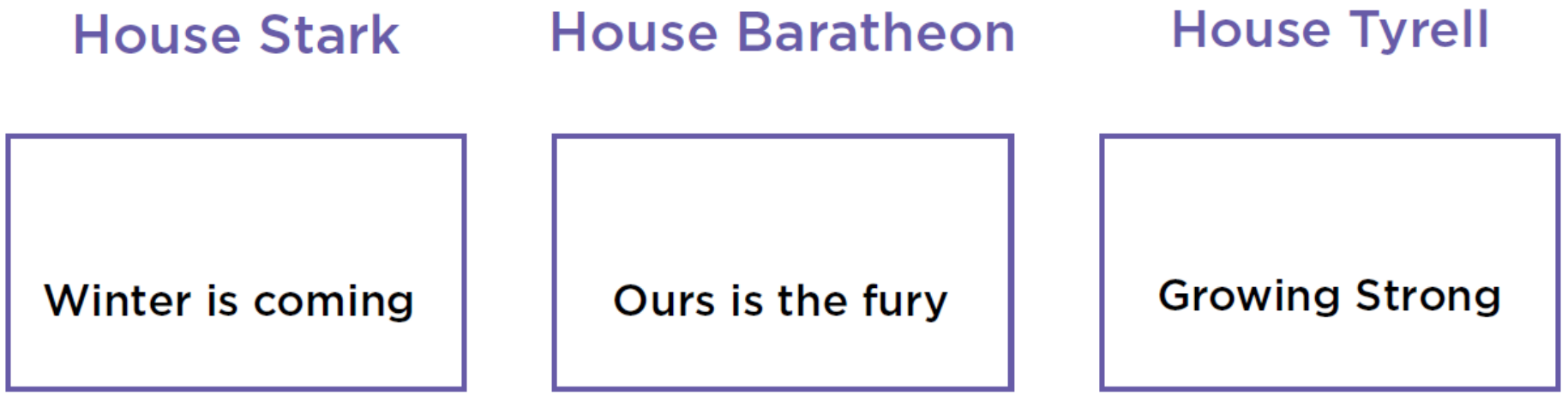
E-commerce

Video

E-learning

Inverted Index

Documents Have Content



Inverted Index

Tokenize Text into Words

winter
is
coming
ours
the
fury
growing
strong

split words

lowercased

removed punctuation

Inverted Index

Tokenize Text into Words

winter	1
is	2
coming	1
ours	1
the	1
fury	1
growing	1
strong	1

Inverted Index

Tokenize Text into Words

winter	1
is	2
coming	1
ours	1
the	1
fury	1
growing	1
strong	1

Inverted Index

Tokenize Text into Words

winter	1	Stark
is	2	Stark, Baratheon
coming	1	Stark
ours	1	Baratheon
the	1	Baratheon
fury	1	Baratheon
growing	1	Tyrell
strong	1	Tyrell

Inverted Index

Tokenize Text into Words

winter	1	Stark
is	2	Stark, Baratheon
coming	1	Stark
ours	1	Baratheon
the	1	Baratheon
fury	1	Baratheon
growing	1	Tyrell
strong	1	Tyrell

Inverted Index

Dictionary

sorted so lookup is easy

coming	1	Stark
fury	1	Baratheon
growing	1	Tyrell
is	2	Stark, Baratheon
ours	1	Baratheon
strong	1	Tyrell
the	1	Baratheon
winter	1	Stark

Inverted Index

Postings

coming	1	Stark
fury	1	Baratheon
growing	1	Tyrell
is	2	Stark, Baratheon
ours	1	Baratheon
strong	1	Tyrell
the	1	Baratheon
winter	1	Stark

Inverted Index

Search

coming	1	Stark
fury	1	Baratheon
growing	1	Tyrell
is	2	Stark, Baratheon
ours	1	Baratheon
strong	1	Tyrell
the	1	Baratheon
winter	1	Stark

winter

Inverted Index

Search

coming	1	Stark
fury	1	Baratheon
growing	1	Tyrell
is	2	Stark, Baratheon
ours	1	Baratheon
strong	1	Tyrell
the	1	Baratheon
winter	1	Stark

fury

Inverted Index

Search

coming	1	Stark
fury	1	Baratheon
growing	1	Tyrell
is	2	Stark, Baratheon
ours	1	Baratheon
strong	1	Tyrell
the	1	Baratheon
winter	1	Stark

is

Inverted Index

Search

coming	1	Stark
fury	1	Baratheon
growing	1	Tyrell
is	2	Stark, Baratheon
ours	1	Baratheon
strong	1	Tyrell
the	1	Baratheon
winter	1	Stark

coming OR strong

Inverted Index

Search

coming	1	Stark
fury	1	Baratheon
growing	1	Tyrell
is	2	Stark, Baratheon
ours	1	Baratheon
strong	1	Tyrell
the	1	Baratheon
winter	1	Stark

fury and growing

Inverted Index

Search

Searches Using Inverted Indices

- Find all words ending with “ong”

  strong ⟶ gnorts

- Search for all words starting with “gno”

Inverted Index

Search

Searches Using Inverted Indices

- Split words into n-grams for substring search
- yours ⟶ yo, you, our, ours, urs
- Match substrings with n-grams

Inverted Index

Searches Using Inverted Indices

- Geo-hashes for geographical search
- Algorithms such as Metaphone for phonetic matching
- “Did you mean?” searches use a Levenshtein automaton

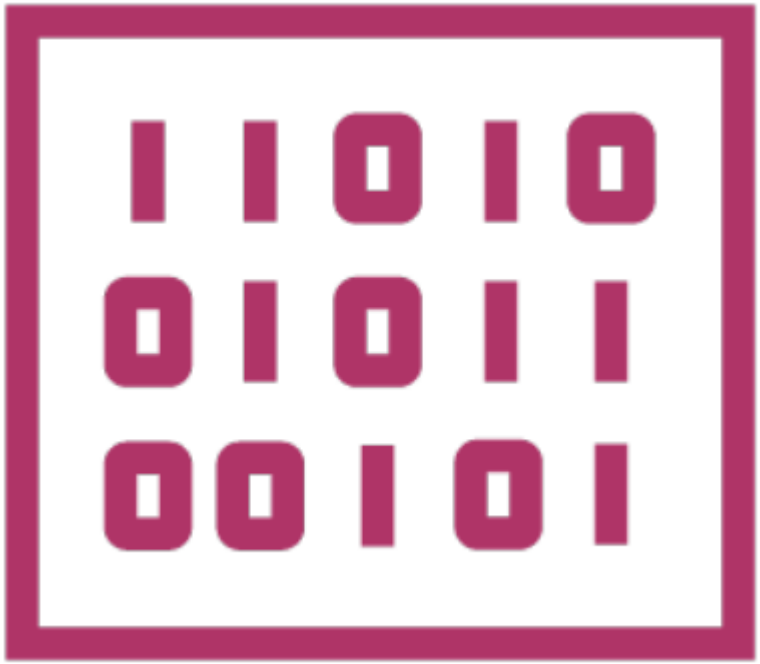
Inverted Index

- An inverted index is at the heart of a search engine

Implementing Search - Apache Lucene

Apache Lucene

The indexing and search library for a high performance, full-text search engine.

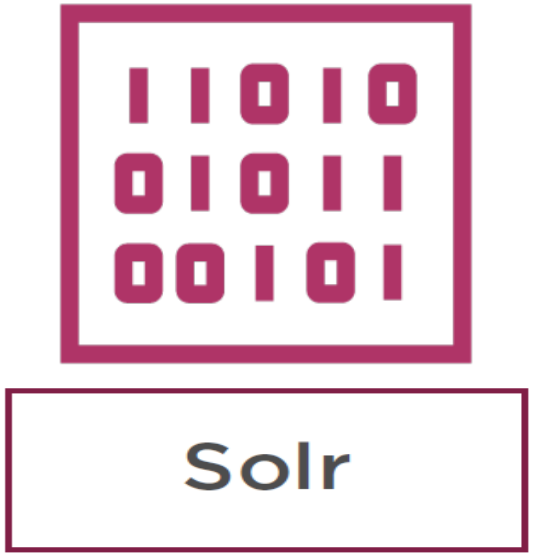


Open source, free to use written in Java, ported to other languages.

Just like Hadoop in the distributed computing world, Lucene is the nucleus of several technologies built around it.

Implementing Search - Apache Lucene

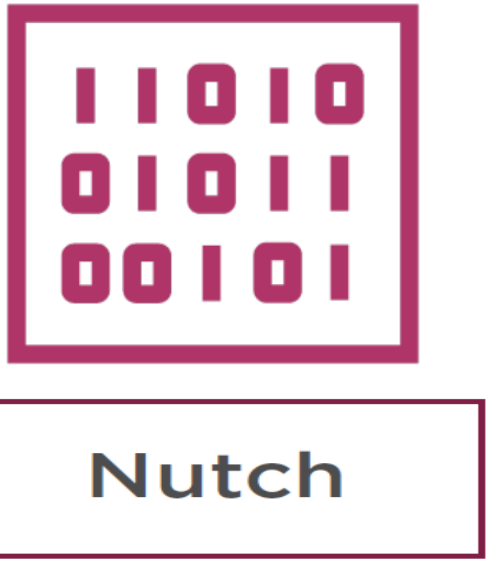
Apache Lucene



Web crawling and index parsing

Implementing Search - Apache Lucene

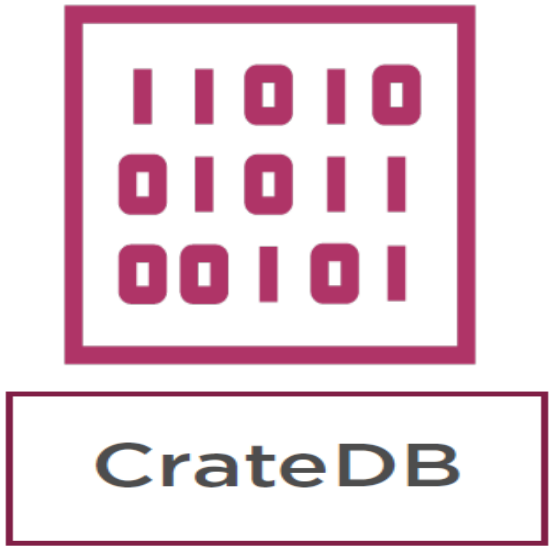
Apache Lucene



Open source, free to use written in Java, ported to other languages

Implementing Search - Apache Lucene

Apache Lucene

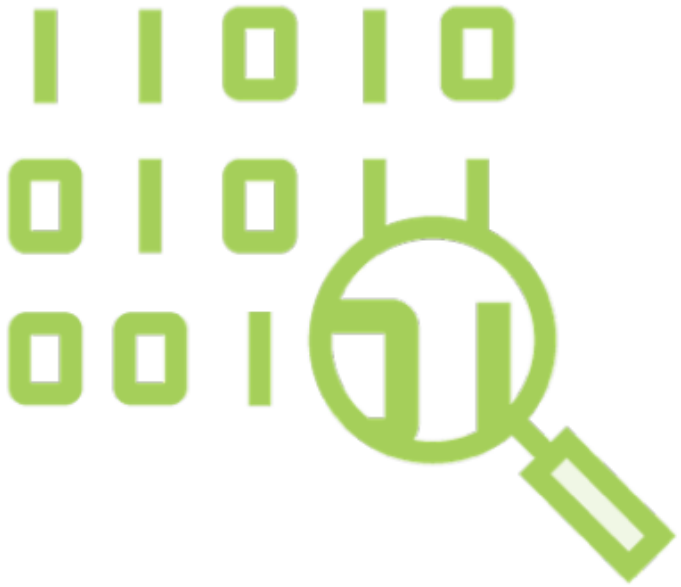


Open source, SQL distributed database

Elasticsearch

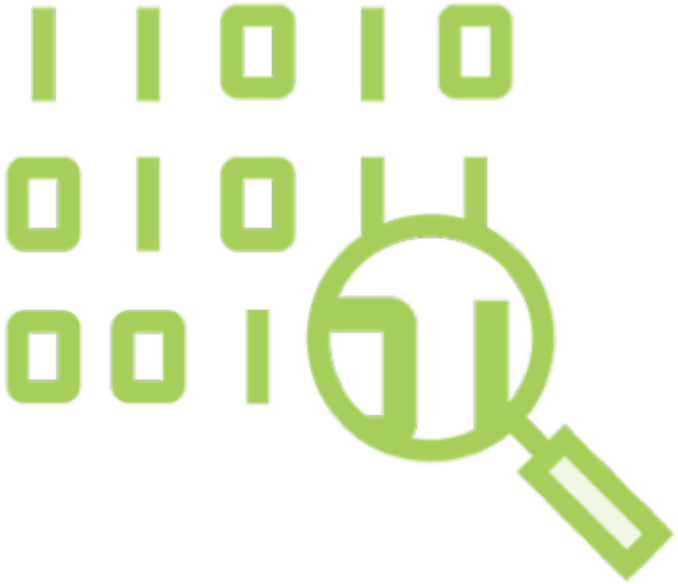
Elasticsearch is a distributed search and analytics engine which runs on Lucene

Introducing Elasticsearch



- An open source, search and analytics engine, written in Java built on Apache Lucene

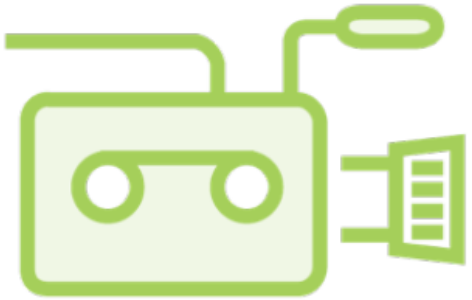
Introducing Elasticsearch



- Distributed: Scales to thousands of nodes
- High availability: Multiple copies of data
- RESTful API: CRUD, monitoring and other operation via simple JSON-based HTTP calls
- Powerful Query DSL: Express complex queries simply
- Schemaless: Index data without an explicit schema

Elasticsearch

Product catalog
Inventory
Autocomplete



Video clips
Categories
Tags

Courses
Authors
Topics

Elasticsearch

Mining log data for insights

Price alerting platform

Business analytics and intelligence

Working with Elasticsearch

As a service in the cloud

On your local machine

https://www.elastic.co/cloud/as-a-service

Basic Concepts of Elasticsearch

Near Realtime Search

Very low latency, ~1 second from the time a document is indexed until it becomes searchable

Basic Concepts of Elasticsearch

Single server

Performs indexing

Allows search

Has a unique id

and name

Node

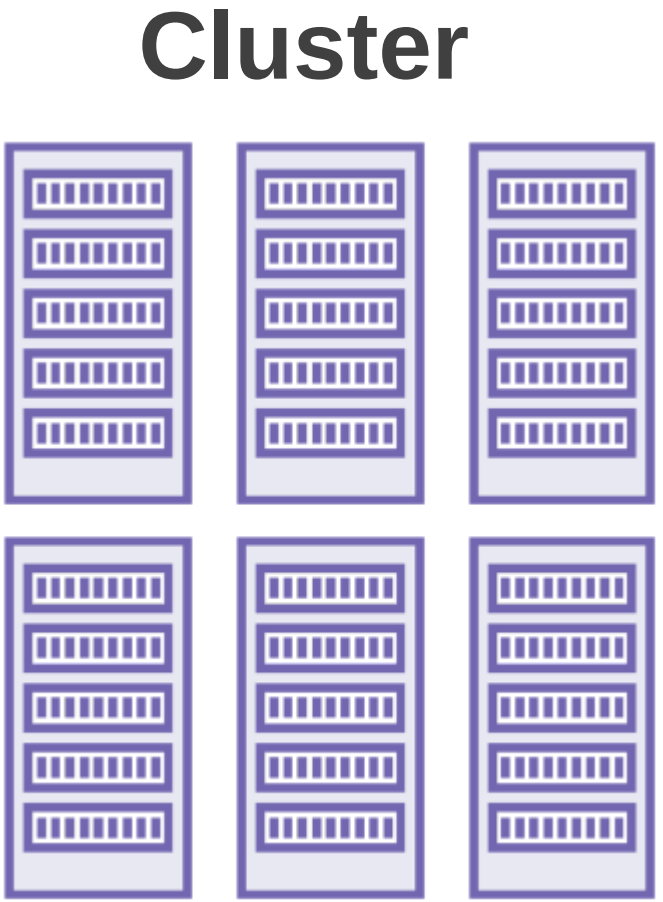
Basic Concepts of Elasticsearch

Collection of nodes

Holds the entire indexed data

Has a unique name

Nodes join a cluster using the cluster name



Basic Concepts of Elasticsearch

Document

A whole bunch of documents that need to be indexed so they can be searched

Basic Concepts of Elasticsearch

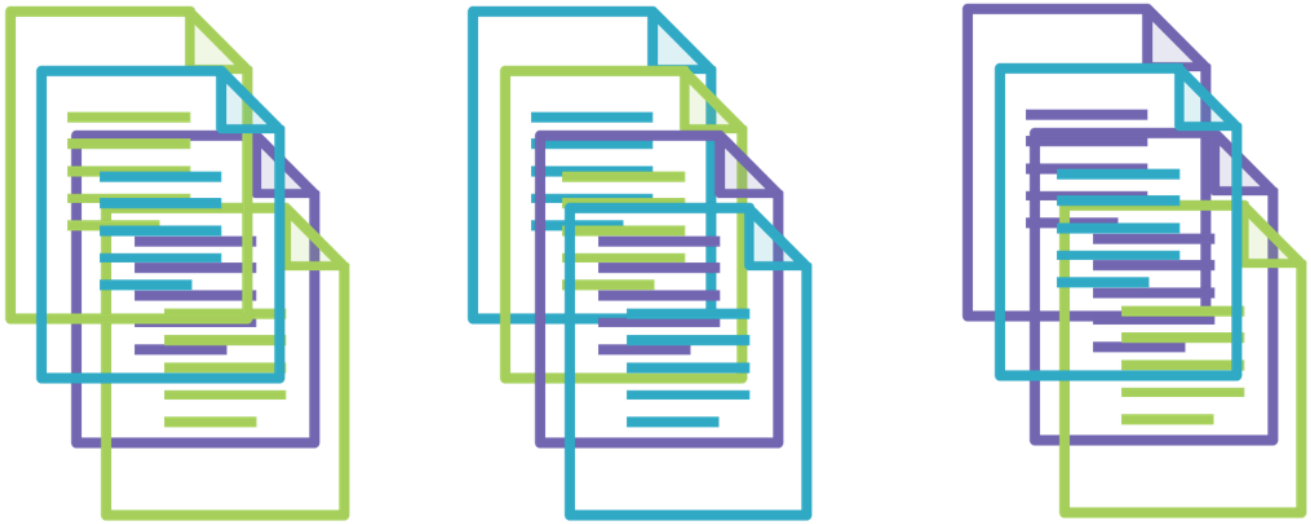
Document

catalog , reviews

- titles, description, comments

Basic Concepts of Elasticsearch

Type



Documents are divided into categories or types

Basic Concepts of Elasticsearch

Index

All of these types of documents make up an index

Basic Concepts of Elasticsearch

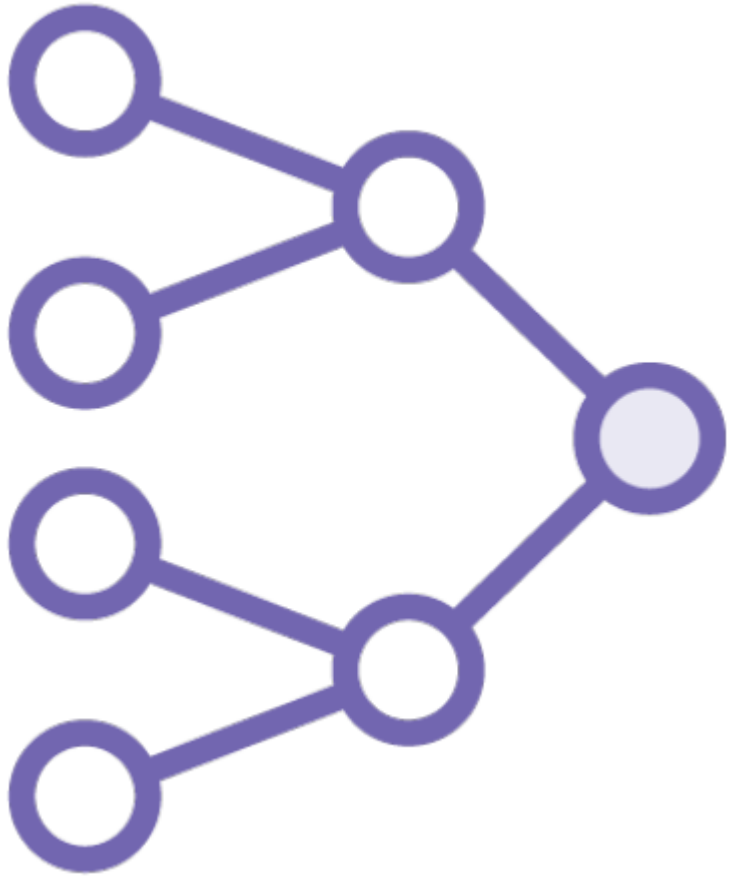
Index

Collection of similar documents

Identified by name

Any number of indices in a cluster

Different indices for different logical groupings



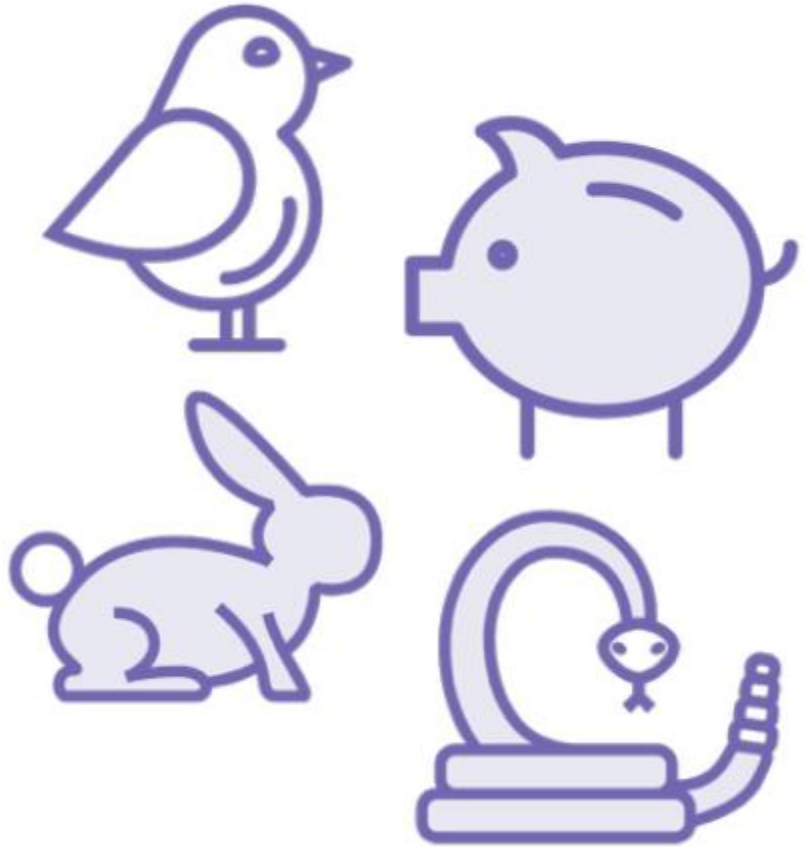
Basic Concepts of Elasticsearch

Type

Logical partitioning of documents

User defined grouping semantics

Documents with the same fields belong to one type



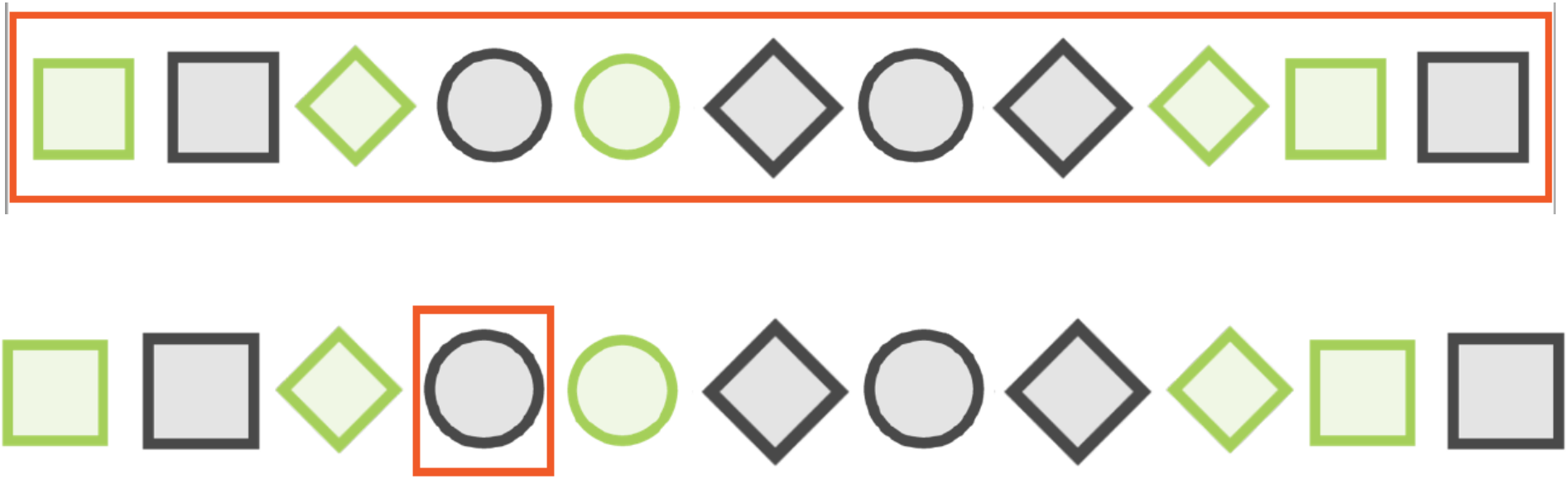
Basic Concepts of Elasticsearch

Document

- Basic unit of information to be indexed
- Expressed in JSON
- Resides within an index
- Assigned to a type within an index

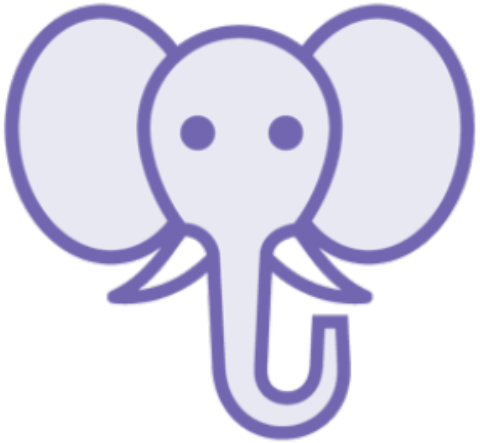
Index, Shards, Replicas

Document in an Index

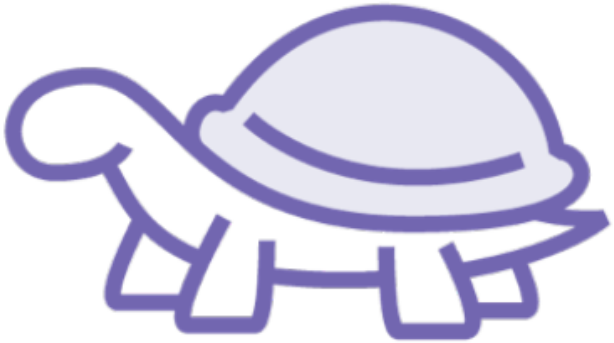


Index, Shards, Replicas

Document in an Index



Too **large** to fit in the hard disk of one node



Too **slow** to serve all search requests from one node

Index, Shards, Replicas

Shards

Split the index across multiple nodes in the cluster

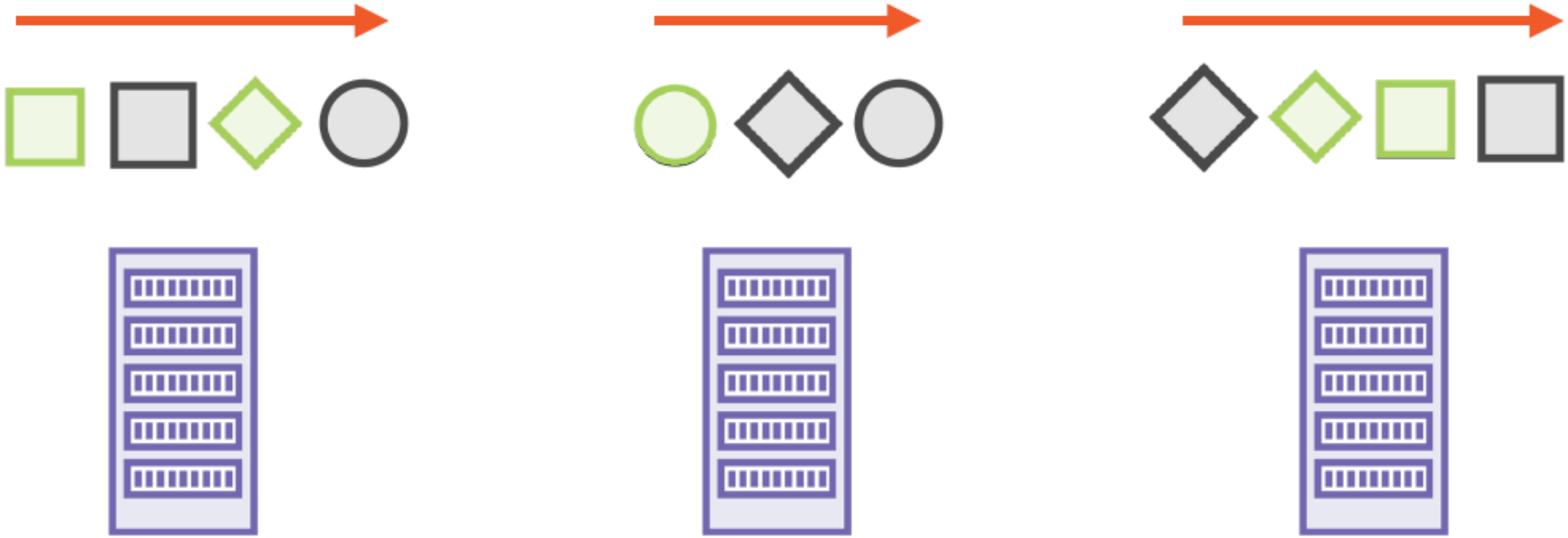
Index, Shards, Replicas

Shards

Sharding an index

Index, Shards, Replicas

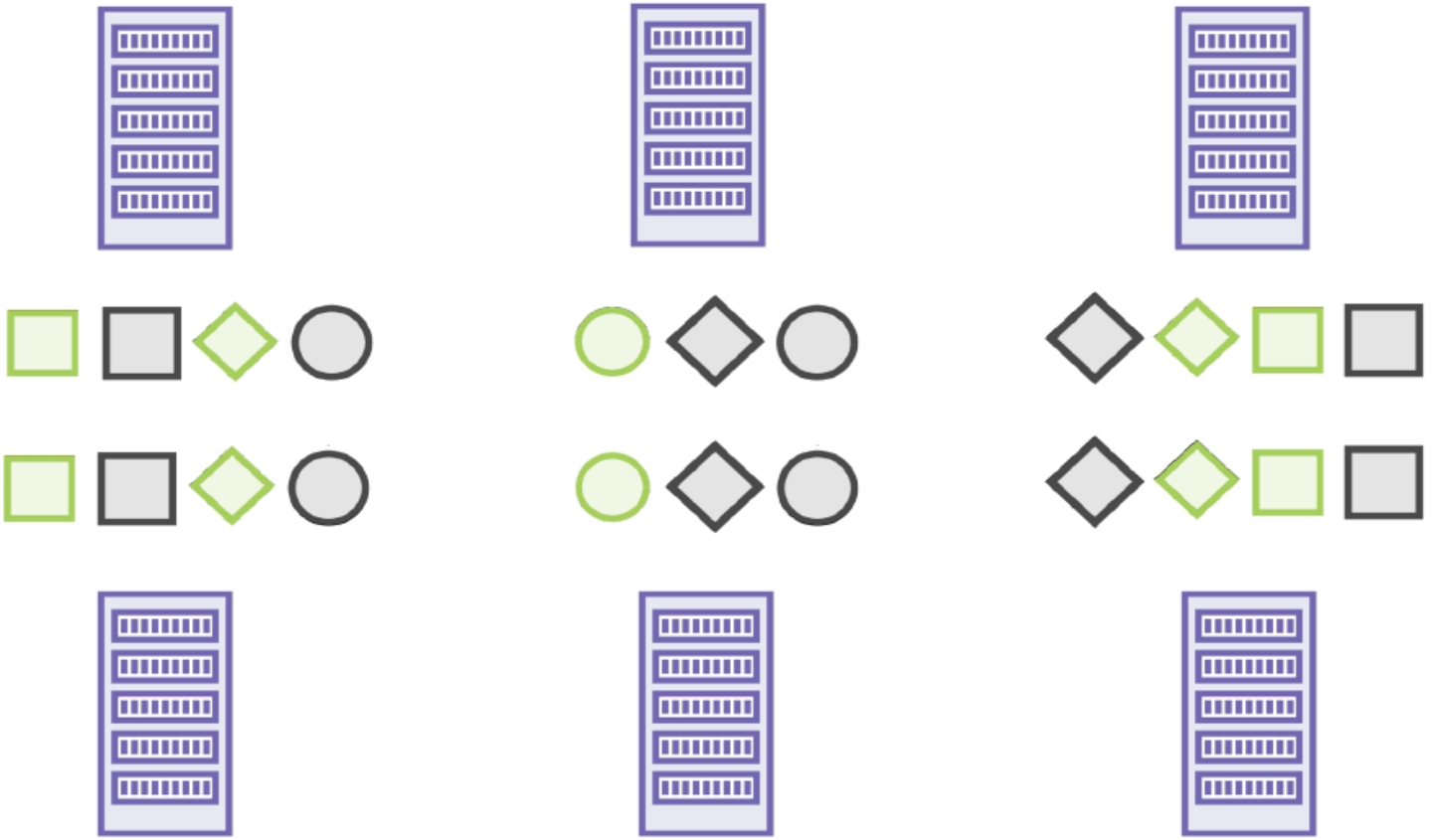
Shards



Search in parallel on multiple nodes

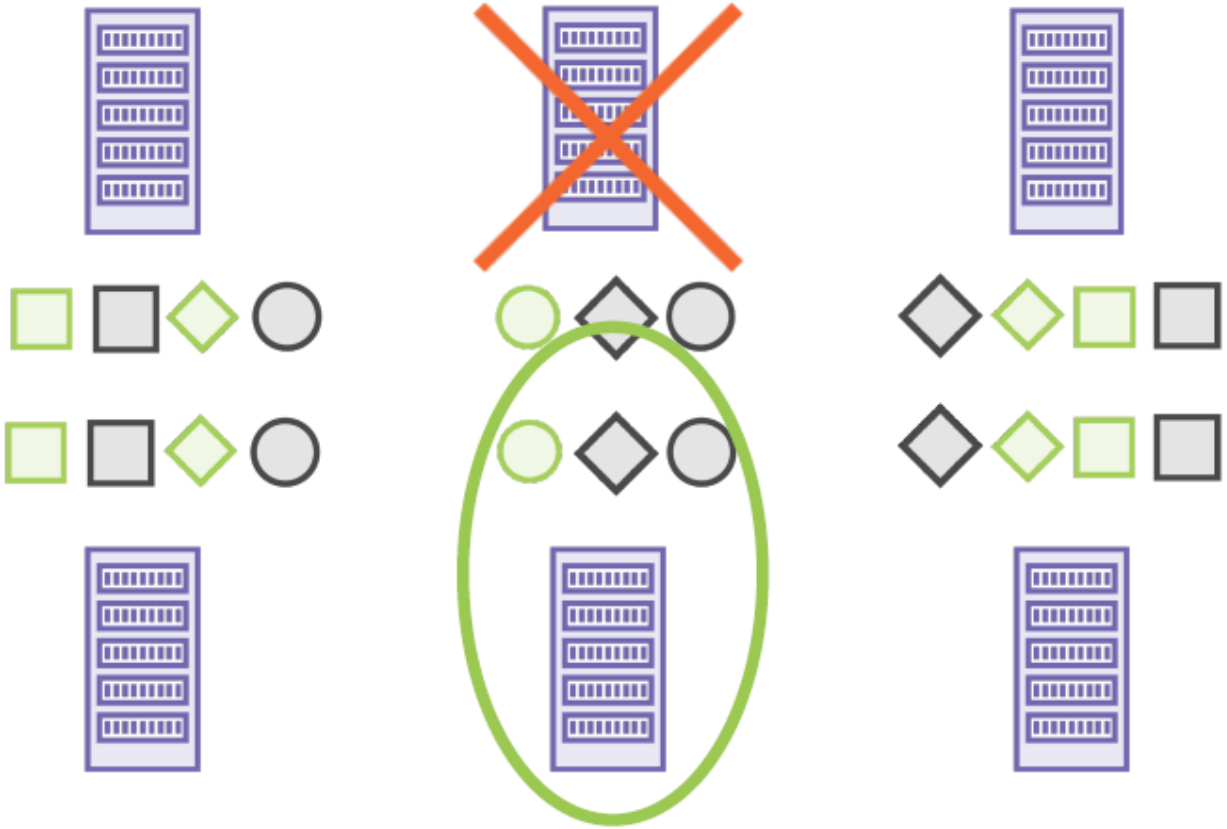
Index, Shards, Replicas

Replicas



Index, Shards, Replicas

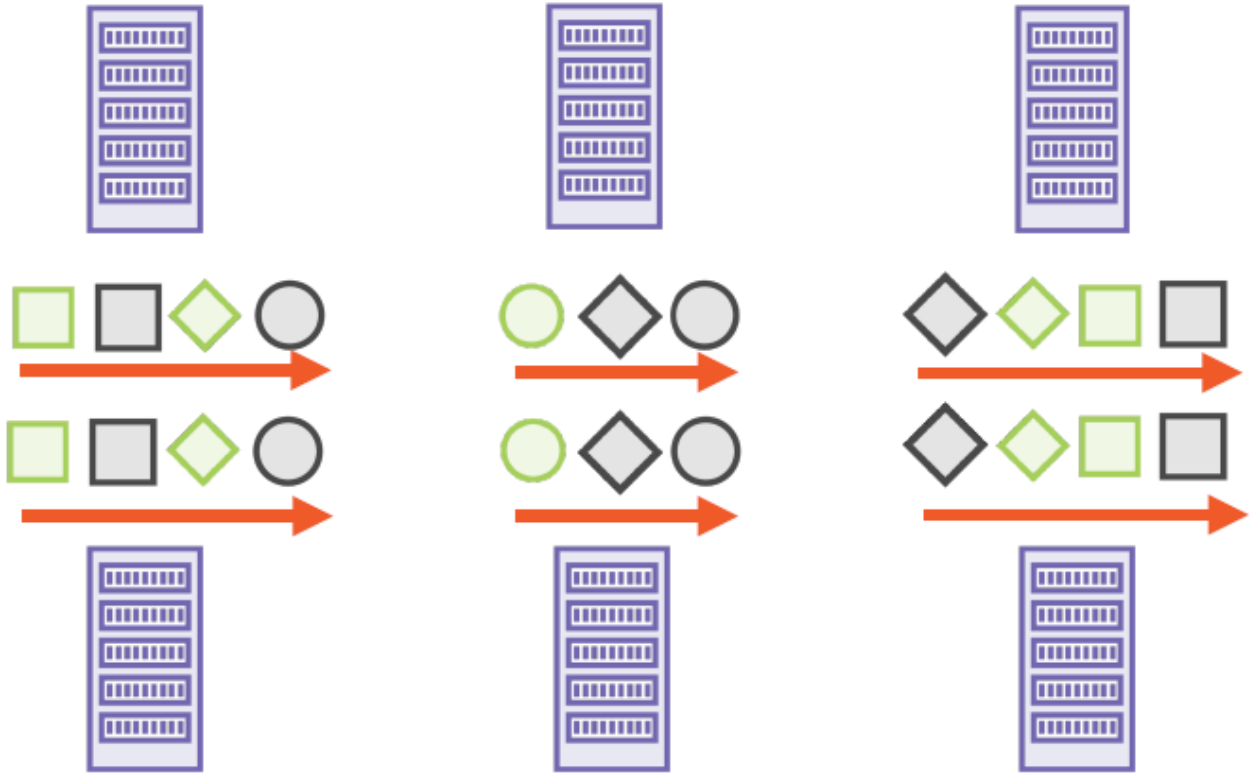
Replicas



High availability in case a node fails

Index, Shards, Replicas

Replicas

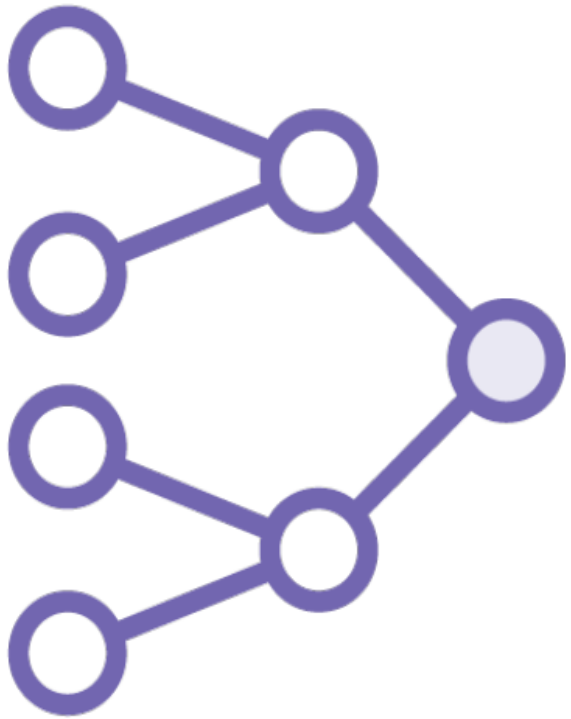


Scale search volume/throughput by searching multiple replicas

Index, Shards, Replicas

Shards and Replicas

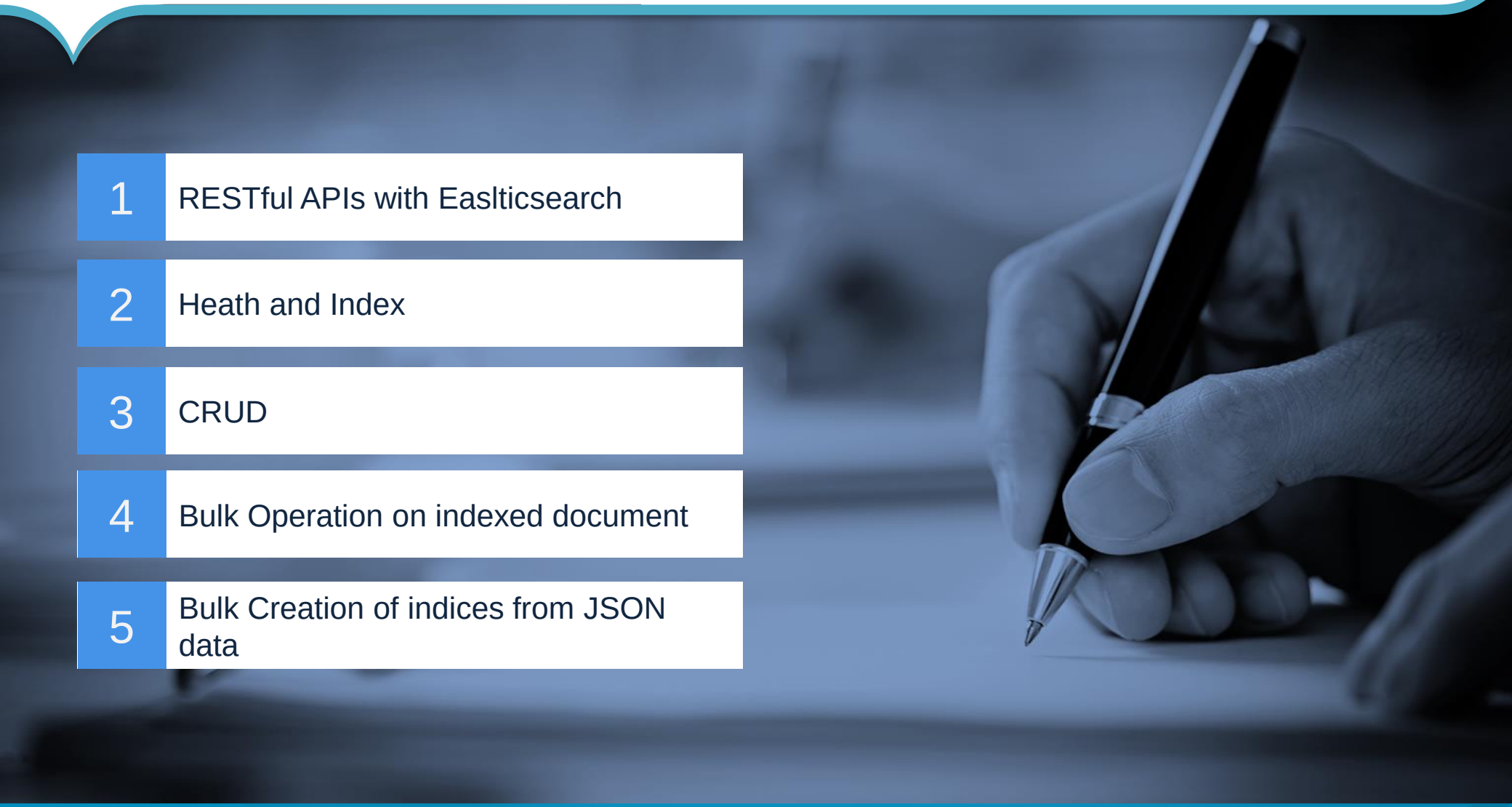
- An index can be split into multiple shards
- A shard can be replicated zero or more times
- An index in Elasticsearch has 5 shards and 1 replica by default



Summary

- Learnt a little search engine history, ubiquitous nature of search
- Understood the basics steps involved in indexing and searching documents
- Learnt how the inverted index data structure works
- Got a brief introduction to Elasticsearch and its building blocks
- Set up and installed Elasticsearch on your local machine

Course Map – CRUD operations using the Elasticsearch APIs



1	RESTful APIs with Easlticsearch
2	Heath and Index
3	CRUD
4	Bulk Operation on indexed document
5	Bulk Creation of indices from JSON data

RESTful APIs with Easlticsearch

RESTful APIs

- Elasticsearch uses REST APIs to administer the cluster, perform CRUD operations, search etc.
- Data is sent to and received from the server in JSON form

RESTful APIs with Easlticsearch

Cluster Health Status

```
curl "localhost:9200/_cat/health?v&pretty"
```

- **Green:**
  - All shards and replicas are available for requests, cluster fully functional
- Yellow:
  - Some replicas may not be available, cluster is still functional.
- Red:
  - Some shards not available, cluster NOT fully functional

RESTful APIs with Easlticsearch

cURL for Requests to REST APIs

- cURL is a tool which allows you to transfer data from and to a server using a variety of protocols
- HTTP, FTP, GOPHER, IMAP, LDAP etc.

CRUD

Demo

- Update documents by id:
  - whole documents
  - partial documents
- Delete documents in an index
- Delete the entire index

Bulk Operation on indexed document

Demo

- Bulk operations on documents:
  - retrieve multiple documents
  - index multiple documents
  - multiple operations in one command

Bulk Creation of indices from JSON data

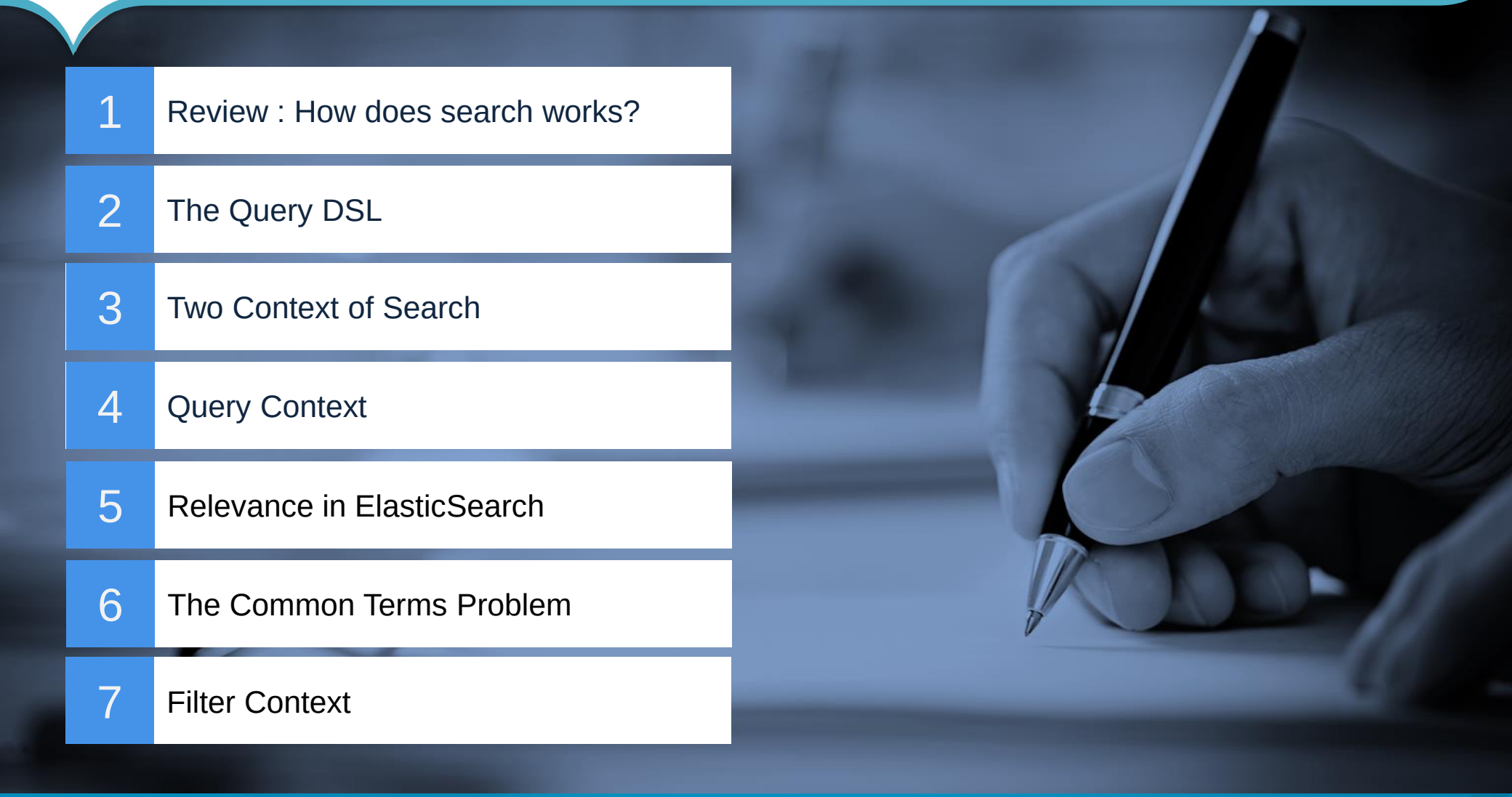
Demo

- Bulk index documents from a JSON file

Summary

- Performed CRUD operations on indexes holding documents
- Implemented bulk operations on indexed documents
- Created indices in bulk from JSON data in a file

Course Map - Executing Search Requests Using Elasticsearch Query DSL



1. Review : How does search works?
2. The Query DSL
3. Two Context of Search
4. Query Context
5. Relevance in ElasticSearch
6. The Common Terms Problem
7. Filter Context

Review : How does search works?

Know of the document's existence

Index the document for lookup

Know how relevant the document is

Retrieve ranked by relevance

Web crawler

Inverted index

Scoring

Search

Review : How does search works?

ElasticSearch:

Specify documents to index

Elasticsearch creates the inverted index behind the scenes

Applies a scoring algorithm for each document for each term

Retrieves documents using a Query DSL

The Query DSL

Query DSL

A flexible, expressive search language that Elasticsearch uses to expose most of the power of Lucene through a simple JSON interface. It is what you should be using to write your queries in production. It makes your queries more flexible, more precise, easier to read, and easier to debug.

https://www.elastic.co/guide/en/elasticsearch/guide/current/query-dsl-intro.html

Two Context of Search

Query Context:

How well does this document match this query?

Filter Context:

Does this document match this query clause?

Two Context of Search

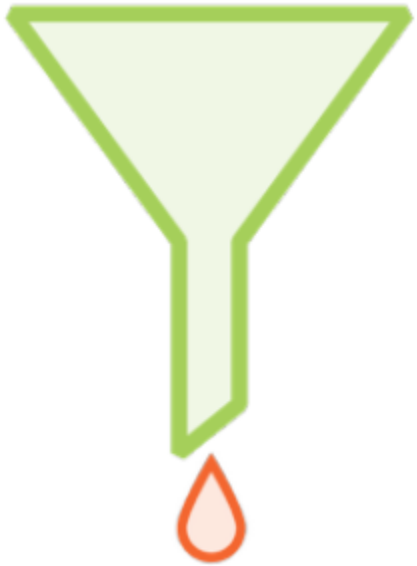
Query Context:

- **Included or not:** Determine whether the document should be part of the result
- **Relevance score:** Calculated for every search term the document maps to
- **High score, more relevant:** More relevant documents, higher in the search rankings

Two Context of Search

Filter Context:

- **Included or not:** Yes/no determines whether included in the result
- **No scoring:** No additional relevance ranking in the search results
- **Structured data:** Exact matches, range queries
- **Faster:** Only determine inclusion in results, no scoring to consider



Query Context

Query Terms Specification

- Search terms as URL query parameters

- Search terms within the URL request body

Note:

- *Stateless Search – Elastic search maintain no open cursor or session for your search.*
- *Search Multiple Indices -*
  - *curl -XGET "localhost:9200/**customers,products**/_search?pretty“*
  - *curl -XGET "localhost:9200/**products/mobiles,laptops/**_search?pretty"*

Query Context

Query Params vs. Request Boby

Query params options are a subset of options available in the request body.

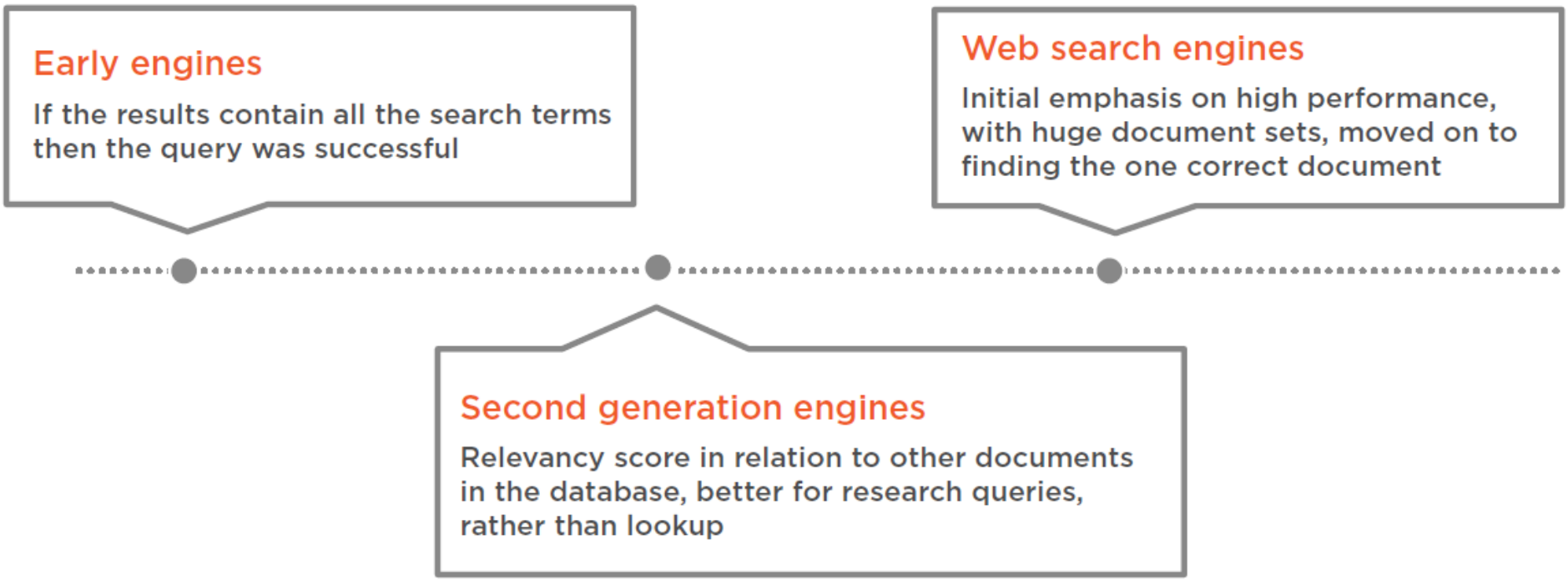
Relevance in Elasticsearch

Meaning of Relevance:

- The search results answered your question or solved your problem
- The user understands easily why the search engine retrieved these results

Relevance in Elasticsearch

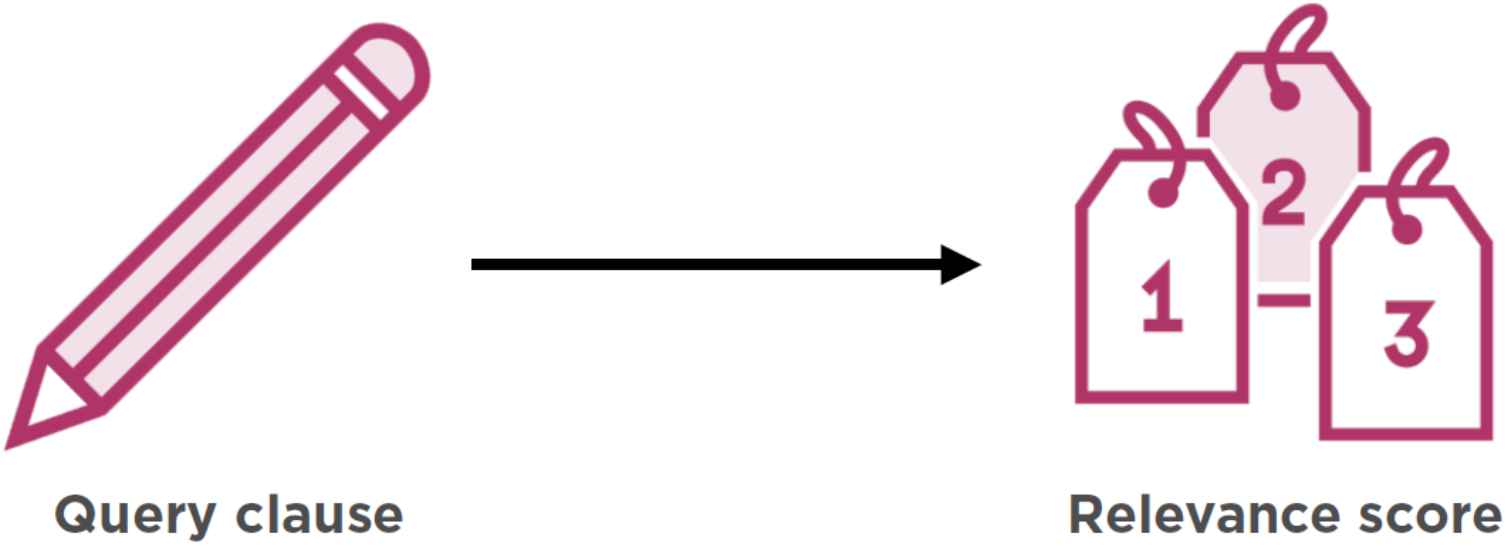
Meaning of Relevance:



Relevance in Elasticsearch

- **Represented by the _score field in every search result**
- **Higher the value of _score more relevant the document**

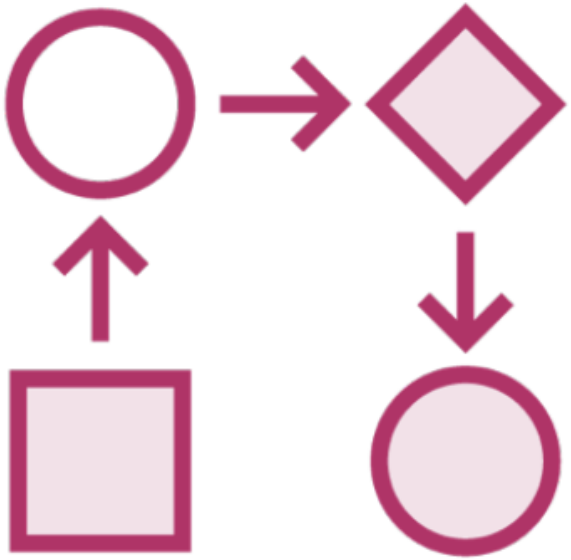
Relevance in Elasticsearch



- Each document has a different relevance score based on the query clause
- Fuzzy searches might look at how similar the search term is to the word present in the document
- Term searches might look at the percentage of search terms that were found in the document

Relevance in Elasticsearch

Elasticsearch Relevance Algorithm



TF/IDF

- **T**erm **F**requency/**I**nverse **D**ocument **F**requency

Relevance in Elasticsearch

TF/IDF Relevance Algorithm

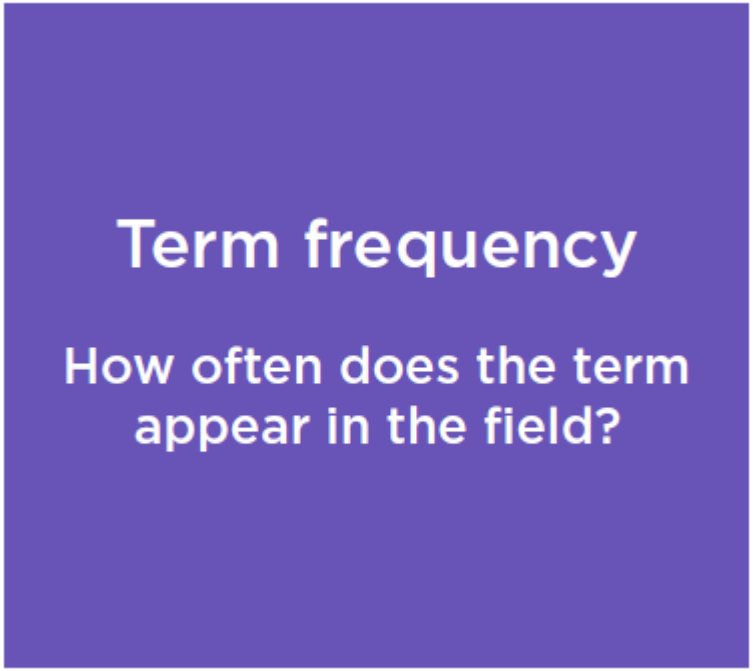
- **Term frequency:** How often does the term appear in the field?
- **Inverse document frequency:** How often does the term appear in the index?
- **Field-length norm** How long is the field which was searched?

Relevance in Elasticsearch

Term Frequency:

More often, more relevant

A field containing 4 mentions of a term is more relevant than one which has just one mention



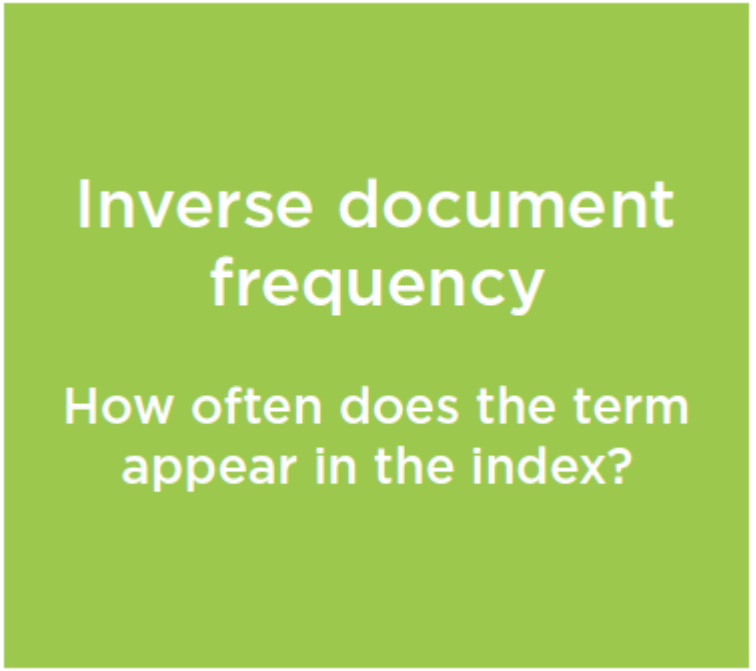
Relevance in Elasticsearch

Inverse Document Frequency

More often, less relevant

If the term is really common across documents in the index, its relevance for a particular document is low

e.g. stopwords such as “the”, “this”

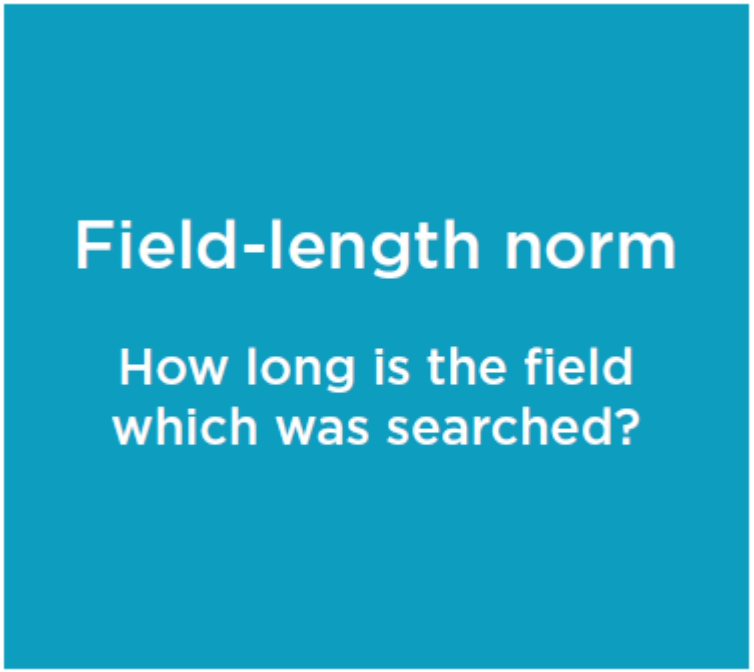


Relevance in Elasticsearch

Field-length Norm

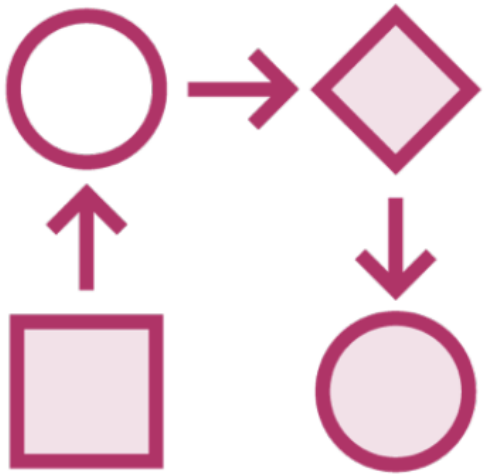
Longer fields, less relevant

A term appearing in a longer field is one of a larger set, so less relevant e.g. words in the title of a book are more relevant than words in the contents



Relevance in Elasticsearch

Elasticsearch Relevance Algorithm



- The TF/IDF score can be combined with other factors based on the query clause

Relevance in Elasticsearch is calculated using TF/IDF in combination with other factors

The Common Terms Problem

Common Terms

- Search for **“The quick brown fox”**

The Common Terms Problem

Common Terms

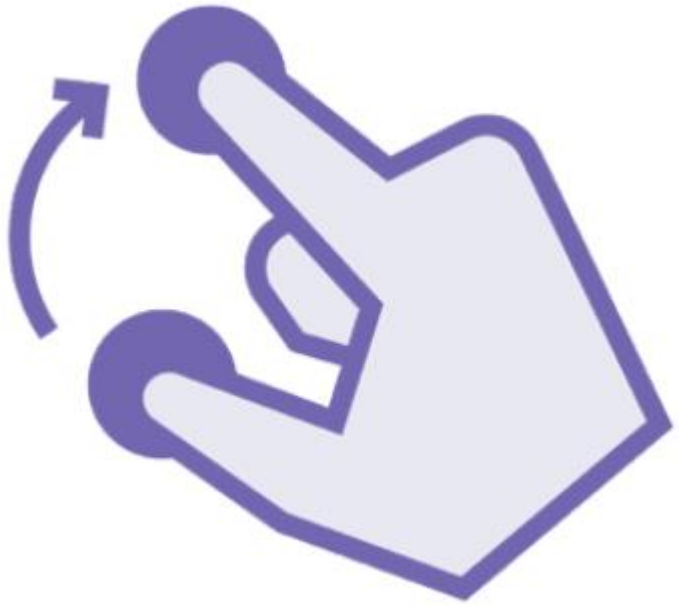
- The word “the” is likely to match a huge number of documents
- With low relevance to the actual search

- Leaving out stopwords can have unexpected impact
- Unable to distinguish between “great” and “not great”

The Common Terms Problem

Split Terms: Low and High Frequency

- Low: “quick brown fox”
- High: “the”



Self Adjusting Algorithm

The Common Terms Problem

Split Terms: Low and High Frequency

Low: “quick brown fox”

- Search for documents which have the rarer terms first

The Common Terms Problem

Split Terms: Low and High Frequency

- Look for the high frequency terms in the document subset which match the low frequency terms

High : “the”

Improved relevance, good performance

Cutoff_frequency: terms with frequency of >0.1% are common terms

Compound Queries: The Boolean Query

Boolean Query

- Matches documents by combining multiple queries using Boolean operators such as AND, OR

Compound Queries: The Boolean Query

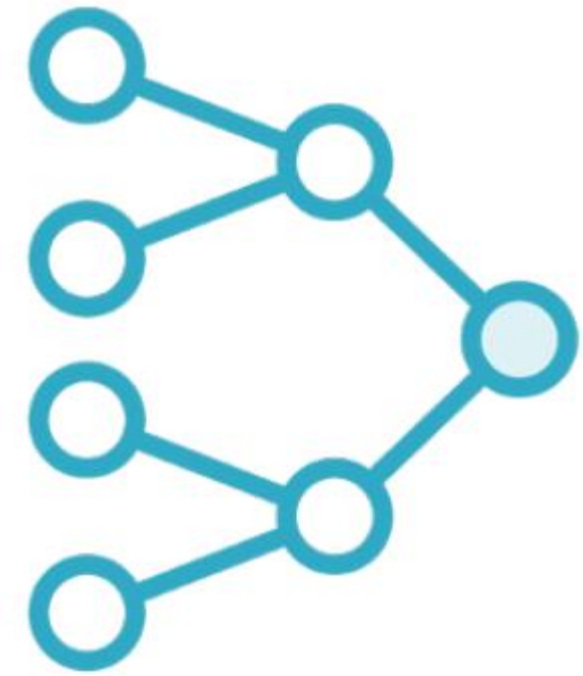
Boolean Query

- **must**
  - The clause must appear in matching documents
- **should**
  - The clause may appear in matching documents but may not sometimes
- **must_not**
  - The clause must not appear in the document results
- **filter**
  - The clause must appear in results but results are not scored

Compound Queries: The Boolean Query

Term Query

- The exact term needs to be found in the inverted index for indexed documents
- The terms found in the index may vary based on how you analyze them



Filter Context

Filters

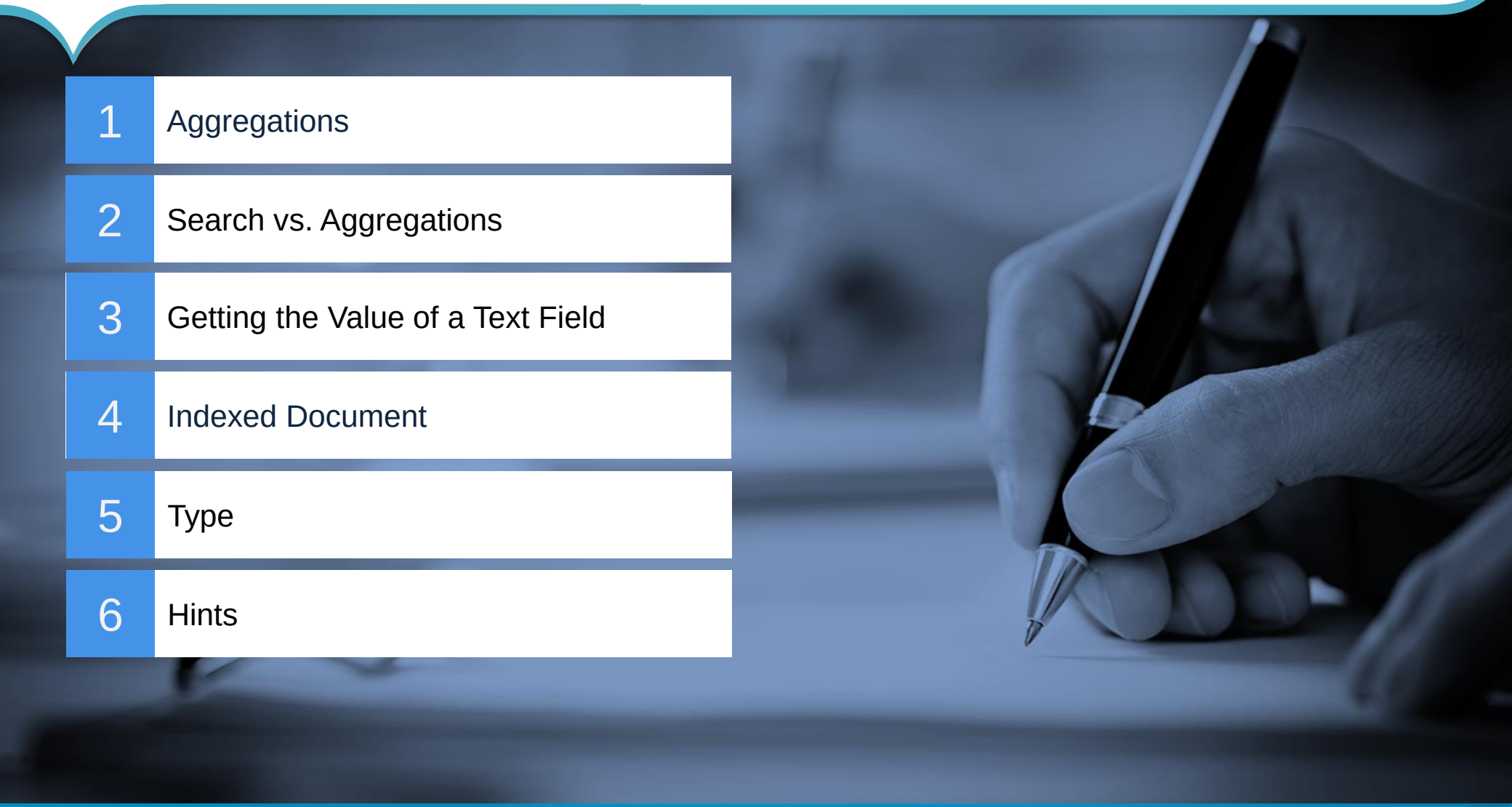
- The documents in the result are not scored

- Each document responds yes/no to whether it should be included in the result

Summary

- Understood the Query DSL that Elasticsearch uses for search queries
- Worked with searches which need relevance and filters where relevance is not required
- Worked with full text searches, term searchers, compound searches, filters
- Understood the basics of the TF/IDF algorithm for relevance
- Implemented all queries using the Elasticsearch REST API

Course Map - Executing Analytical Queries Through Aggregations



1	Aggregations
2	Search vs. Aggregations
3	Getting the Value of a Text Field
4	Indexed Document
5	Type
6	Hints

Aggregations

Four Types of Aggregations

- Metric
- Bucketing
- Matrix
- Pipeline

Aggregations

Metric

- Aggregations over a set of documents
  - All documents in a search result
  - Documents within a logical group

Aggregations

Bucketing

- Logically group documents based on search query
- A document falls into a bucket if the criteria matches
- Each bucket associated with a key

Aggregations

Matrix

- Operates on multiple fields and produces a matrix result
- Experimental and may change in future releases
- *Not covered*

Aggregations

Pipeline

- Aggregations that work on the output of other aggregations
- Experimental and may change in future releases
- *Not covered*

Search vs. Aggregations

Search

Inverted index of the terms present in documents

The terms themselves can be hashed and stored in the index

“Which documents contain this term?”

Aggregations

Actual value of fields present in documents

Actual values of the terms are needed, hash values do not suffice

“What is the value of this field for this document?”

Getting the Value of a Text Field

- Text field values are stored in an **in-memory** data structure called fielddata
- fielddata is built **on demand** when a field is used for aggregations, sorting etc

Indexed Document

Logically group these documents into buckets

Types

Each bucket satisfies some criterion

Hints

- Cardinality of a field
  - The number of **unique** values present in the field across documents.
- To enable fielddata use mapping API
- sum_other_doc_count:
  - The sum of document counts that are not part of the response.
- doc_count_error_upper_bound:
  - Calculating aggregations across index shards can result in imperfections

Hints

- Bucketing, Metric Aggregations:
  - Find the average age of male and female customers
- 3- Level Nested Aggregations
  - Find the average age of male and female customers with in range.
- Filter Aggregations
  - The average age of customers in the state of Texas

Summary

- Perform analytics queries on Elasticsearch
- Metric, bucketing, matrix and pipeline aggregations
- Implemented queries for metrics and bucketing aggregations
- Worked with multi-level nesting of aggregations

Capgemini

CONSULTING.TECHNOLOGY.OUTSOURCING

People matter, results count.

About Capgemini

With almost 180,000 people in over 40 countries, Capgemini is one of the world's foremost providers of consulting, technology and outsourcing services. The Group reported 2014 global revenues of EUR 10.573 billion.

Together with its clients, Capgemini creates and delivers business and technology solutions that fit their needs and drive the results they want. A deeply multicultural organization, Capgemini has developed its own way of working, the Collaborative Business Experience™, and draws on Rightshore®, its worldwide delivery model.

www.capgemini.com