3 - Architecture

Sunday, 2 February, 2025 11:29 Manhã

Baseado em Apache Lucene, que é um buscador/indexador de texto mas sem muitos features.
ElasticSearch envelopa isso de um jeito melhor com mais recursos.

DATA IN

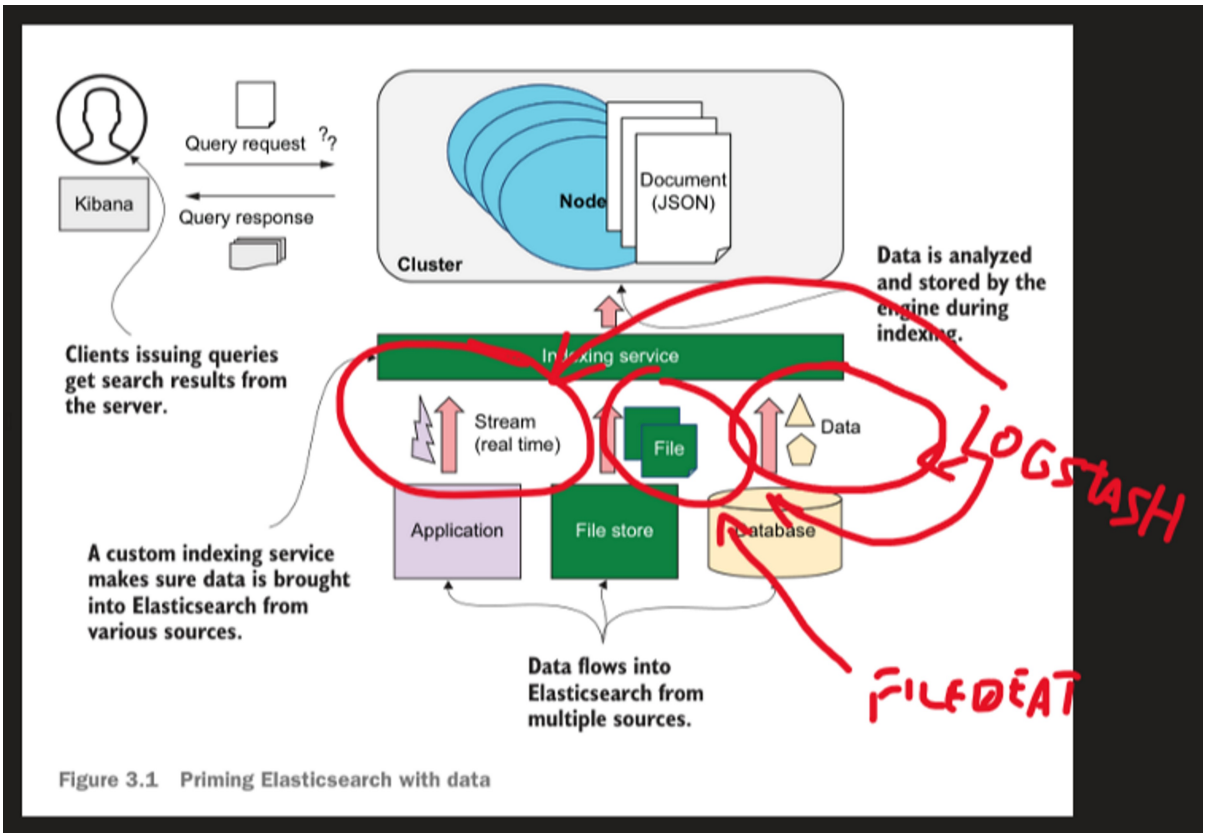


Figure 3.1 Priming Elasticsearch with data

Dado é separado por node e por tipo, no armazenamento.
Index é um bucket, uma coleção lógica. Como uma tabela num banco de dados relacional.
Um *shard* é uma instância do Apache Lucene.

- Mapping atribui cada campo do JSON na ingestão a um data type apropriado (e.g. numeric).
- Texto também passa por isso, mas com uma etapa adicional de análise do texto
- Isso tem duas etapas: **tokenização** (é desmontado em tokens) e **normalização**.
- Há várias formas de tokenização, por exemplo, por espaço em branco.
- Tokens são persistidos sem transformações.
- Na normalização, tokens são enriquecidos, e.g. stemming, sinônimos.
- Texto analisado é persistido em um node.

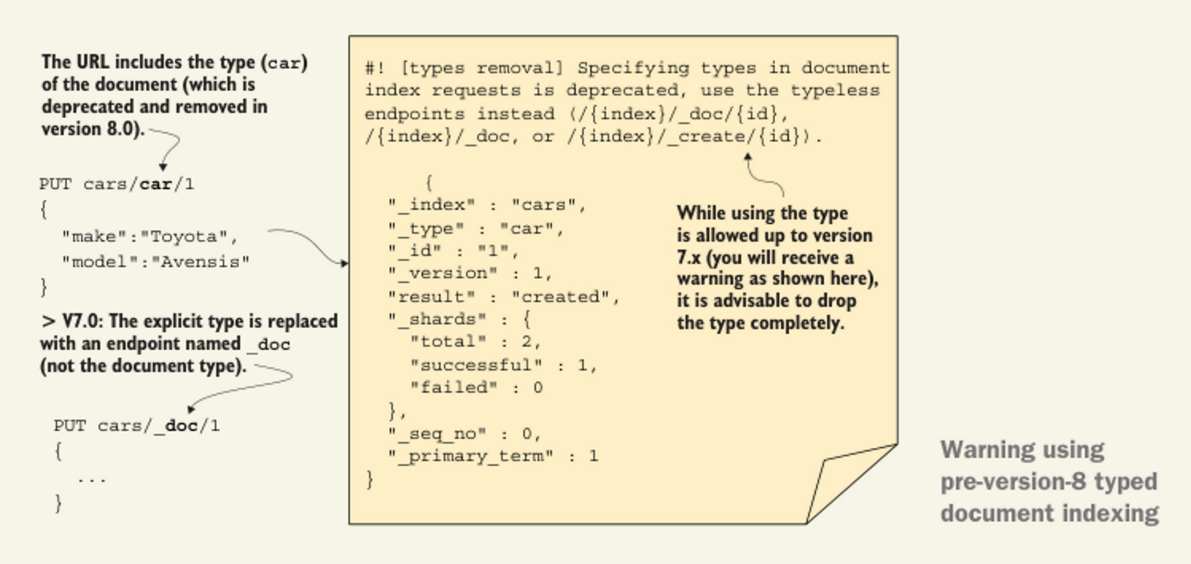
DATA OUT

- Busca e analytics dos dados.
- Se a query for de texto, texto da query é analisado (i.e. tokenizado e normalizado) de forma igual com o processo

na etapa de *data in*.
- Tokens resultantes disso é que são procurados no índice.

- ***Document*** é a unidade de dado armazenada em um index. Formato JSON.
- Esse JSON é lido e cada valor é armazenado por tipo e index específico.
- Permite campos aninhados.
- O documento inteiro é armazenado em um único ID, sem joins.

- Há APIs que operam sobre vários documentos e outras sobre um único documento.
- "*Document type*" foi deprecado na versão 8.x - um único índice podia ter vários tipos diferentes de documentos.
- Foi substituído pelo document type padrão "_doc"



Index

Equivale a um bucket, uma coleção de dados.

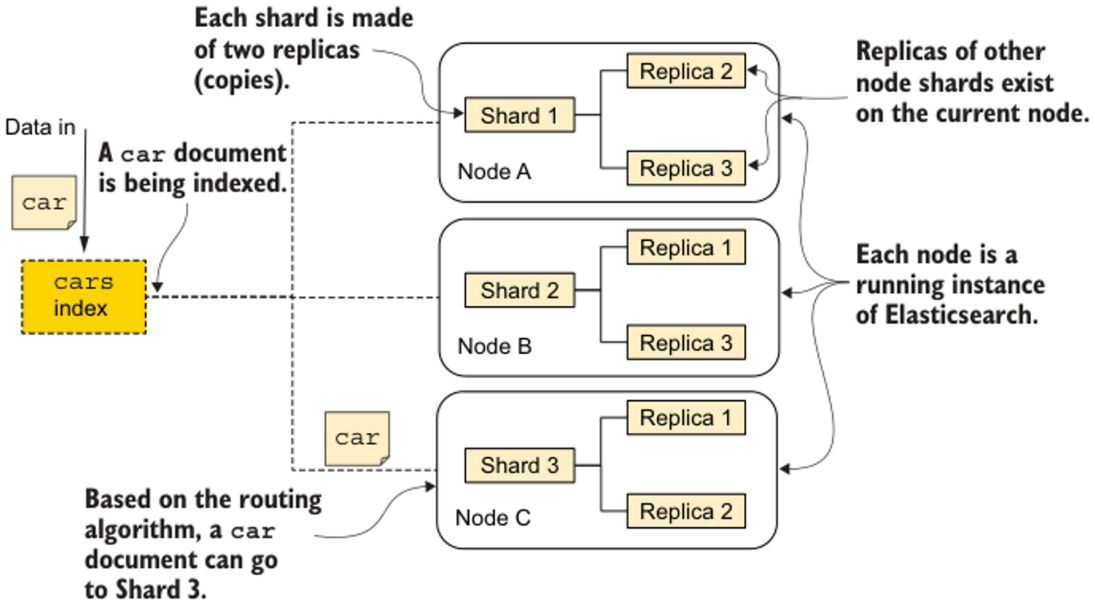
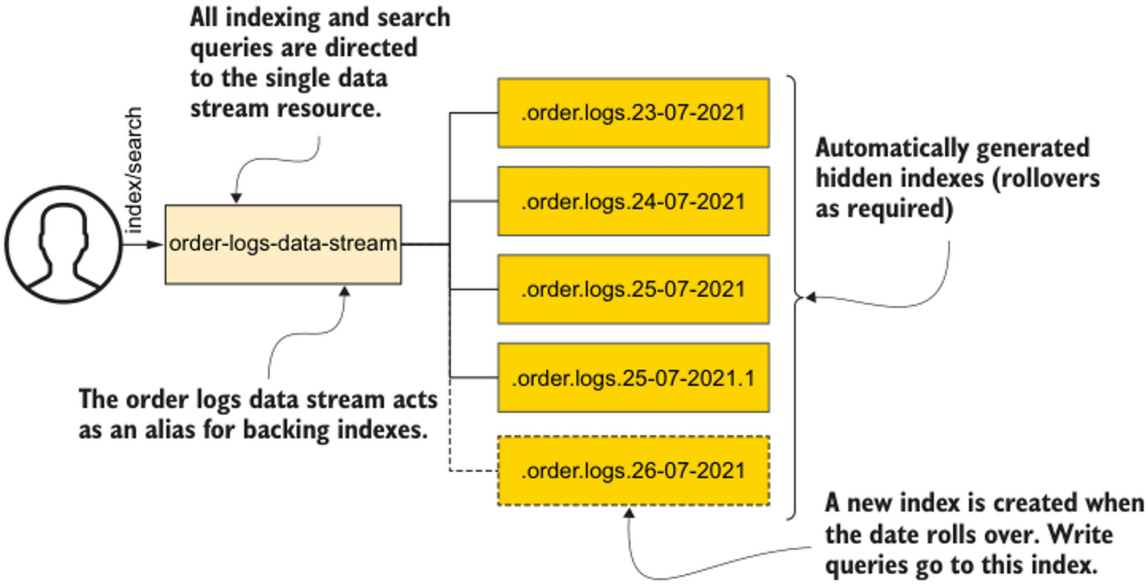


Figure 3.5 An index designed with three shards and two replicas per shard

- Shard é uma instância do Apache Lucene rodando.
- Default a partir da versão 7.x é 1 shard e 1 réplica.
- Shards podem ser *primário* ou *réplicas* - recomendável nunca ter zero réplicas.
- Um index pode existir num node só ou estar distribuído.
- Quantidade de documentos por index é uma variável importante de otimização.
- Alias é um nome alternativo para um índice ou conjunto de índices.
- Número de réplicas pode ser alterado dinamicamente, numa instância operante, mas número de shards não.

- Adicionar nós pode aliviar excesso de dados, ao distribuir mais os shards.
- E.g. para coletar dados separados em vários dias (série temporal), cada coleção-dia um em um índice, podemos fazer uma query num único índice-mãe, criando um alias de agrega todos os índices desejados. *Data stream*.



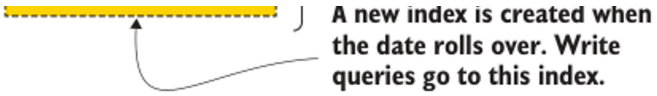


Figure 3.7 A data stream consists of automatically generated hidden indexes.

Shards e Replicas:

Mecanismo sofisticado de salvamento quando indexamos um documento.

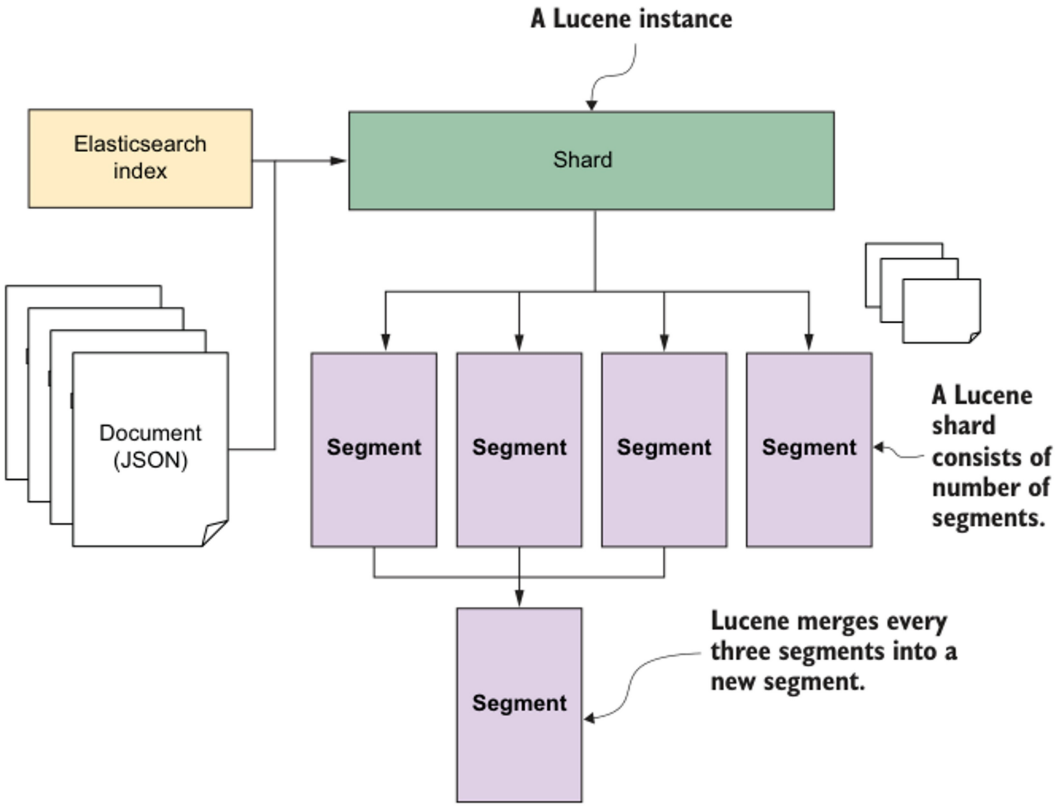
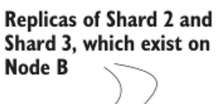


Figure 3.8 Lucene's mechanism for indexing documents

- Réplicas permitem distribuir as requisições de leitura
- Réplicas existem em nodes diferentes, por definição de redundância.
- Um node é uma instância do ElasticSearch
- Um cluster é uma coleção de Nodes

- Cluster tem 3 estados, obtidos por "GET _cluster/health" ou "GET localhost:9200/_cat/health" :
  - Red (algum shard está indisponível)
  - Yellow (Todos os shards estão disponíveis, mas algumas réplicas não)
  - Green (Todos os shards e réplicas em ordem)



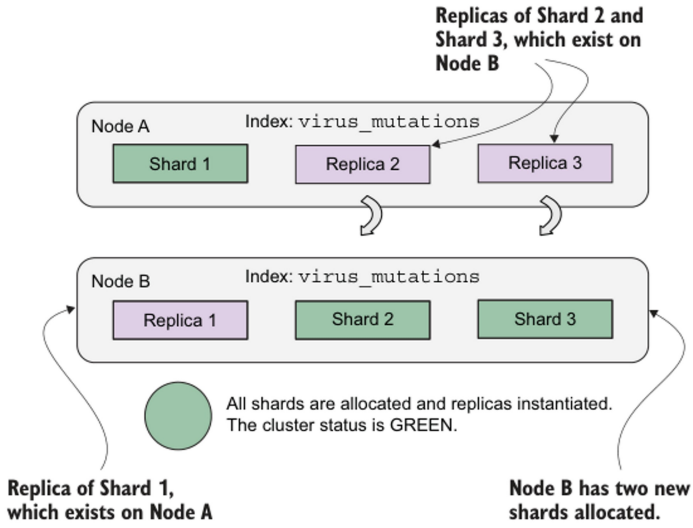


Figure 3.12 Happy days! All shards and replicas are allocated.

- Réplica pode ser promovida a shard primário se algum node crashar

- Uma questão importante é tamanho de shard, não existe receita de bolo.
- Por padrão, não se recomenda passar de 50GB, embora haja exemplos com mais do que isso.
- Se passar, deve ser distribuído em vários shards.
- Em relação à memória, recomenda-se hospedar até 20 shards por GB de heap memory.
- Default de memory de uma instância ES é 1GB, que é customizável.
- Número de shards não é alterável em um instância operante, corromperia os documentos.
- Solução para manipular shards em operação é **reindexação**

Nodes e Clusters:

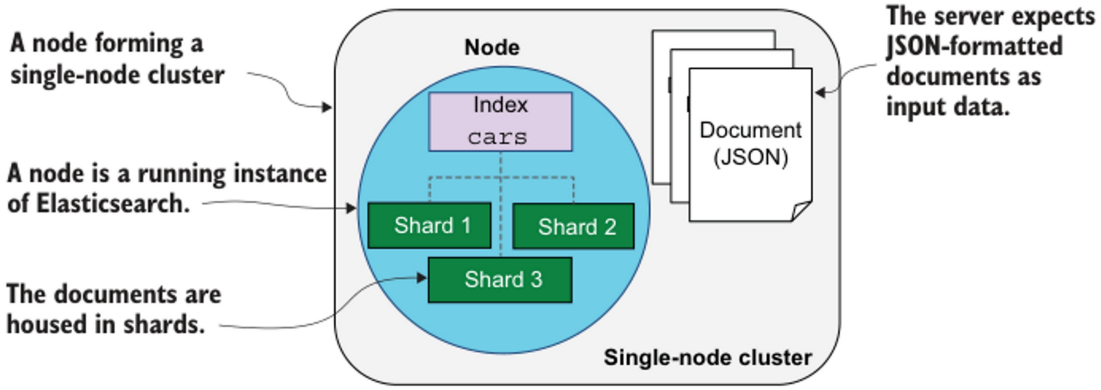


Figure 3.14 A single-node Elasticsearch cluster

- Não se recomenda ter vários nodes na mesma máquina, pois não haveria réplicas, portanto sem backup, YELLOW.
- Arquivo *elasticsearch.yml* define as configurações do cluster, como o nome do cluster onde os nodes seguintes serão agregados.
- Mais nodes <--> mais backups <--> mais desempenho.
- Mais nodes não altera o número de shards, que é fixo, apenas réplicas (até haver reindexação)
- Também é possível criar um **multicluster**. Recomendável separar em clusters diferentes para organizar e assegurar dados.
- A cada node é atribuído um conjunto de responsabilidades (e.g. gerenciar comunicação internodes, indexar

dados, machine learning etc)

- Um node pode mudar de responsabilidades se algum outro crashar
- Data node pode ter variantes como *data_hot*, *data_cold* etc

Table 3.2 Node roles and responsibilities

Role	Description
Master node	Its primary responsibility is cluster management.
Data node	Responsible for document persistence and retrieval.
Ingest node	Responsible for the transformation of data via pipeline ingestion before indexing.
Machine learning node	Handles machine learning jobs and requests.
Transform node	Handles transformation requests.
Coordination node	This is the default role. It takes care of incoming client requests.

- Role de cada node pode ser definido no .yml

Inverted Index:

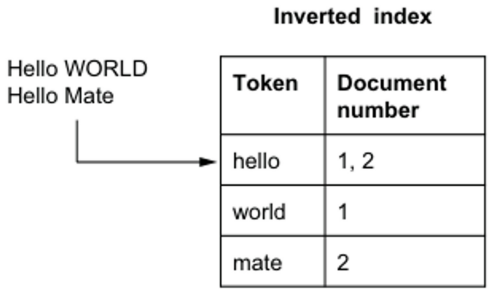


Figure 3.17 An inverted index data structure

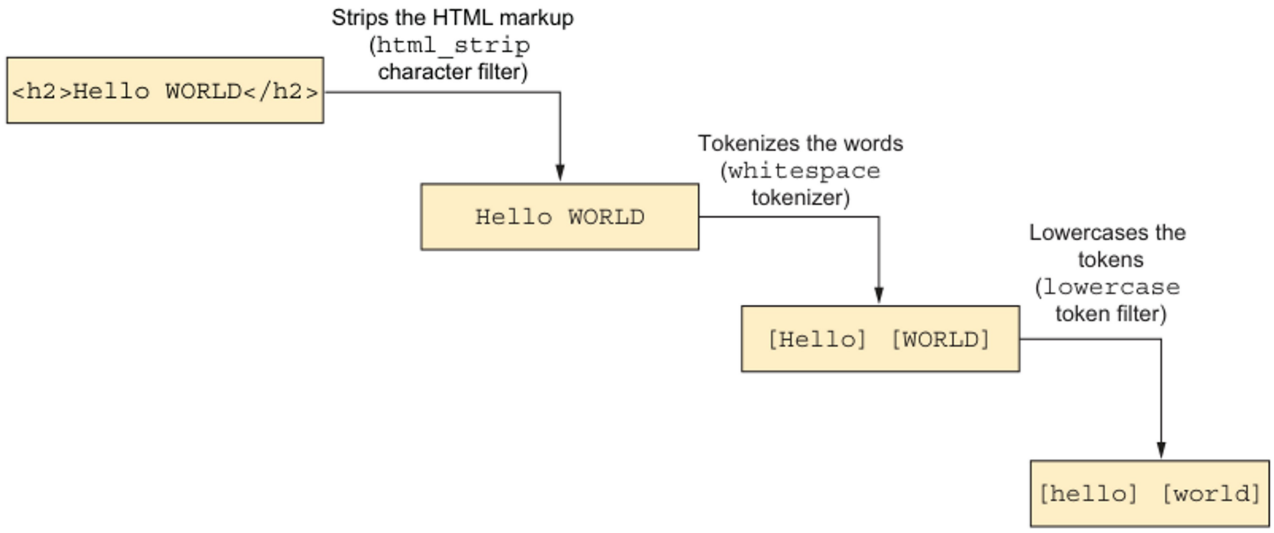


Figure 3.18 Text analysis procedure where Elasticsearch processes text

- Inverted Index também tem um campo "Frequency" que diz quantas vezes um token aparece, afeta scoring.

Relevância:

- ES usa BM25 para atribuir score de relevância por default, que é uma versão melhorada do TF/IDF.
- Podemos customizar uma função de *similarity* pra cada campo

Table 3.5 Elasticsearch's similarity algorithms

Similarity algorithm	Type	Description
Okapi BM25 (default)	`BM25`	An enhanced TF/IDF algorithm that considers field length in addition to term and document frequencies
Divergence from Randomness (DFR)	`DFR`	Uses the DFR framework developed by its authors, Amati and Rijsbergen, which aims to improve search relevance by measuring the divergence between the actual term and an expected random distribution. Terms that occur more often in relevant documents than in a random distribution are assigned higher weights when ranking search results.
Divergence from Independence (DFI)	`DFI`	A specific model of the DFR family that measures the divergence of the actual term frequency distribution from an independent distribution. DFI aims to assign higher scores to documents by comparing the observed term frequencies with those expected in a random, uncorrelated term frequency.
LM Dirichlet	`LMDirichlet`	Calculates the relevance of documents based on the probability of generating the query terms from the document's language model
LM Jelinek-Mercer	`LMJelinek-Mercer`	Provides improved search result relevance compared to models that do not account for data sparsity
Manua		Creates a manual script
Boolean similarity	`boolean`	Does not consider ranking factors unless the query criteria are satisfied

- No Okapi BM25, frequência no documento do termo procurado aumenta a pontuação. Também premia palavras menos comuns no index.
- Normaliza score para o tamanho do documento, evita viés em favor de documentos longos.
- Podemos settar stopwords para serem ignoradas na busca.

Routing Algorithm:

- Atribui um documento a dado shard, pela seguinte função:

- ```
  shard_number = hash(id) % number_of_shards
  ```
- Por isso não podemos mudar número de shards, bagunçaria esta função, documentos seriam inacháveis.

- Escalagem vertical (aumenta a máquina) requer desligar o cluster, se não tivermos modo de evitar o downtime
- Escalagem horizontal (colocar mais máquinas) apenas agrega mais nodes, mais prático e recomendável.

Summary

- Elasticsearch expects data to be brought in to be indexed. Data sources can include simple files, databases, live streams, Twitter, and so on.
- In the indexing process, data undergoes a rigorous analysis phase, during which advanced data structures like inverted indexes are created.
- Data is retrieved or searched via the search APIs (along with the document APIs for single document retrieval).
- Incoming data must be wrapped up in a JSON document. Because the JSON document is the fundamental data-holding entity, it is persisted to shards and replicas.
- Shards and replicas are Apache Lucene instances whose responsibility is to persist, retrieve, and distribute documents.
- When we start up the Elasticsearch application, it boots up as a one-node, single-cluster application. Adding nodes expands the cluster, making it a multi-node cluster.
- For faster information retrieval and data persistence, Elasticsearch provides advanced data structures like inverted indexes for structural data (such as textual information) and BKD trees for nonstructural data (such as dates and numbers).
- Relevancy is a positive floating-point score attached to retrieved document results. It defines how well the document matches the search criteria.
  ```

- Elasticsearch uses the Okapi Best Match (BM) 25 relevancy or similarity algorithm, an enhanced version of the term frequency/inverse document frequency similarity algorithm.
- You can scale Elasticsearch up or out, based on your requirements and available resources. Scaling up beefs up existing machines (adding additional memory, CPU, RAM, etc.), while scaling out spins up more virtual machines (VMs), allowing them to join the cluster and share the load.