Elasticsearch

Administración de Sistemas

Unai Lopez Novoa

unai.lopez@ehu.eus

eman ta zabal zazu

Universidad del País Vasco

Euskal Herriko Unibertsitatea

Contenido

1. Introducción
2. Conceptos
3. Operaciones CRUD
4. Búsqueda

Introducción

- Administrar sistemas con muchos componentes es complejo.
  - Entornos diferentes, contenedores, datos distribuidos, ...

- Monitorizar esos sistemas también lo es.
  - Cada componente genera logs/avisos de forma diferente.

Introducción

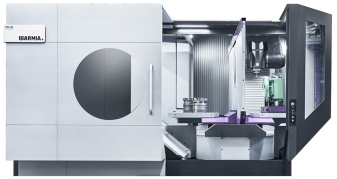
- Escenario de ejemplo: planta industrial
  - Múltiples sistemas gestionan datos.
  - Usuarios de diferentes tipos.

Imagen: https://cfaa.eus/instalaciones-e-investigacion/

Introducción

- Escenario de ejemplo: planta industrial

Elementos de cómputo



Máquina de fabricación

Dispositivos de captura de datos

Servidores de datos

Servidores de cómputo

Usuarios

Analistas de datos

Operarios e ingenieros

Introducción

- La pila ELK es un conjunto de herramientas para monitorización y gestión de logs.
  - Desarrolladas por Elastic: https://www.elastic.co/elastic-stack

Logstash

Recolección y preprocesado de datos

Elasticsearch

BBDD y motor de búsqueda

Kibana

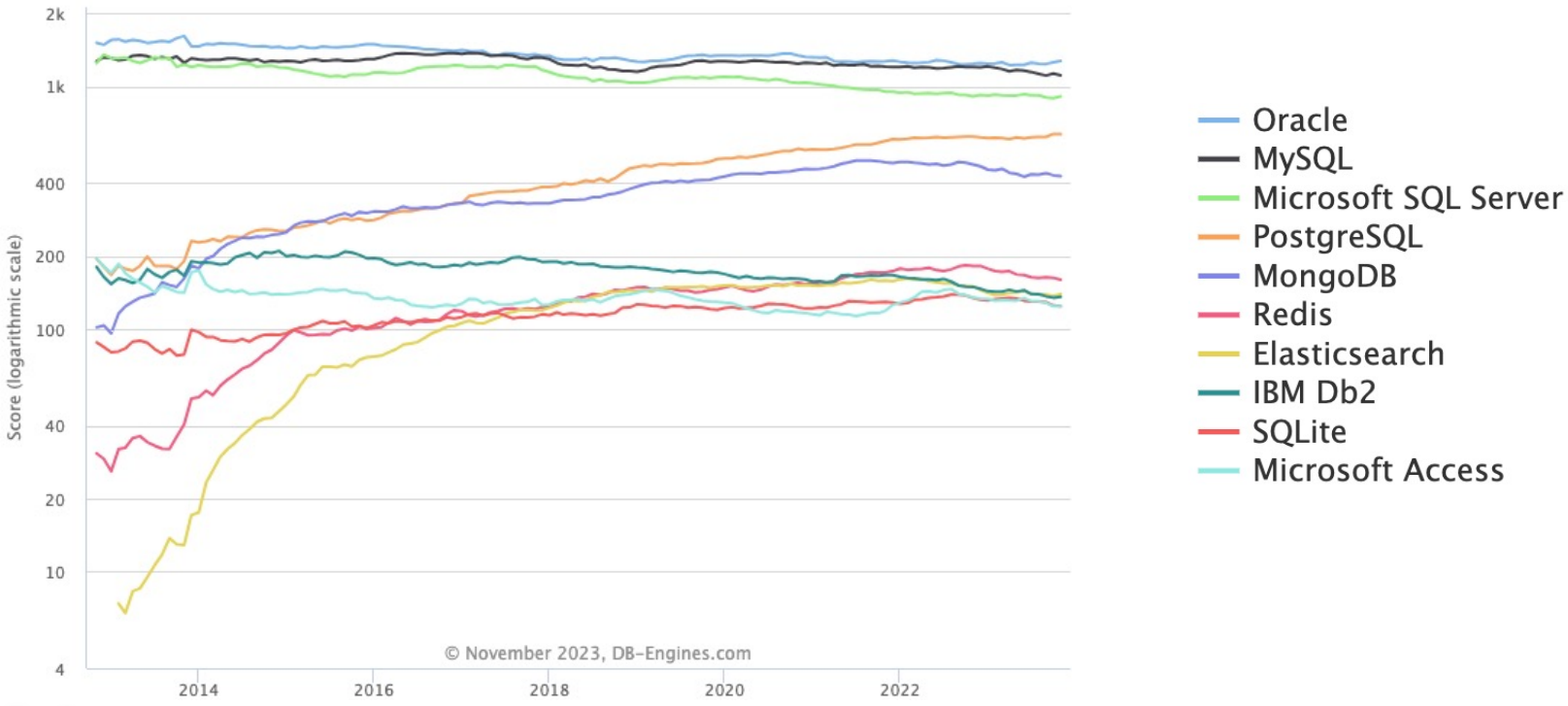
Visualización de datos

Introducción

- Elasticsearch es un motor de búsqueda para texto
  - Distribuido, escalable, Open Source
  - Web: https://www.elastic.co/es/elasticsearch/
- Está basado en Apache Lucene
  - Librería de búsqueda de texto
  - Web: https://lucene.apache.org/
- Internamente, es una base de datos NoSQL.
  - Tiene esquemas flexibles.
  - Los datos se gestionan en formato JSON.

Introducción

- Comparativa de popularidad de BBDDs
  - Fuente: https://db-engines.com/en/ranking_trend
    - Fecha: 24 de noviembre de 2023



Uso

- Instalación con Docker Compose:
  - Despliegue de toda la pila ELK.
  - Fichero YAML en: https://github.com/ulopeznovoa/ELK-8-setup
    - Un servicio por cada componente de ELK.
    - Un volumen para persistencia de datos.

```
services:
  logstash:
    …
  elasticsearch:
    …
  kibana:
    …

volumes:
  data01:
```

Uso

- Requisitos hardware:
  - Tener, al menos, 1 CPU dedicada y 6 GB de RAM.
  - En Compute Engine de GCP, se recomienda crear una instancia, al menos, tipo n1-standard-1 o e2-medium.

- Conectividad:
  - Utiliza los siguientes puertos:
    - 9200: Conexión a la API de Elasticsearch.
    - 80: Interfaz web de Kibana.

- Controles de seguridad deshabilitados.
  - Recomendado activarlos en proyectos reales.

Uso

- Elasticsearch expone una API REST para utilizar sus funcionalidades.
  - Funciones que se exponen en forma de servicios Web.

Operación CRUD	Equivalente REST
Crear (Create)	POST / PUT
Leer (Read)	GET
Actualizar (Update)	PUT / POST / PATCH
Borrar (Delete)	DELETE

  - JSON como formato de intercambio.
  - Se utilizan mediante:
    - Librerías de código
    - Aplicaciones cliente

Uso

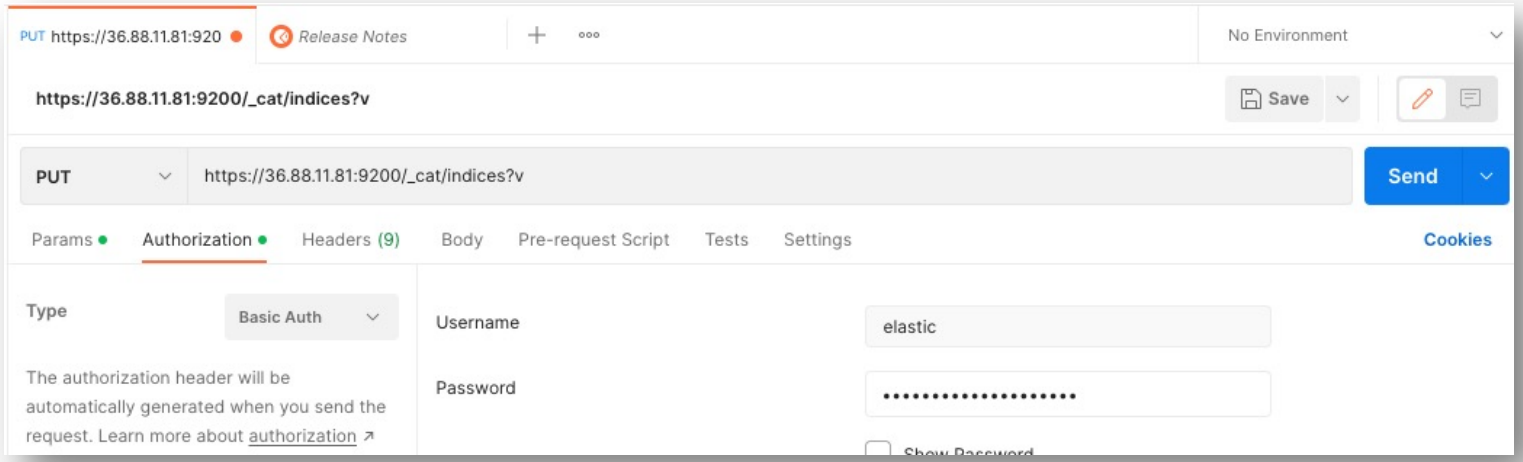
- Recomendación de clientes REST
  - Línea de comandos: curl
    - Web: https://curl.se/
    - Manual: https://curl.se/docs/manpage.html

    - *Ejemplo*: realizar consulta GET al puerto 9200 de la máquina local.

```
curl -XGET 127.0.0.1:9200
```

Uso

- Recomendación de clientes REST
  - Interfaz visual: Postman
    - Web: https://www.postman.com/downloads/



Uso

- Verificar la instalación Elasticsearch

```
curl … -XGET 127.0.0.1:9200
```

  - La respuesta debe ser similar a:

```
{
  "name" : "cfe59530657b",
  "cluster_name" : "docker-cluster",
  "cluster_uuid" : "m4KsyhZmSmCqhOPCxtkhBA",
  "version" : {
    "number" : "8.11.1",
    "build_flavor" : "default",
    "build_type" : "docker",
    "build_hash" : "6f9ff581fbcde658e6f69d6ce03050f060d1fd0c",
    "build_date" : "2023-11-11T10:05:59.421038163Z",
    "build_snapshot" : false,
    "lucene_version" : "9.8.0",
    "minimum_wire_compatibility_version" : "7.17.0",
    "minimum_index_compatibility_version" : "7.0.0"
  },
  "tagline" : "You Know, for Search"
}
```

Ejercicio 1

- Desplegar la pila ELK con Docker Compose.
- Verificar la instalación con un cliente REST.

Conceptos

- Documentos
  - Elementos que se buscan en la BBDD
  - No se refiere sólo al texto
    - También a la estructura en formato JSON
  - Cada documento tiene un ID único y un tipo
  - Ejemplo:

```
{
  "_index": "libros",
  "_id": "1",
  "_version": 1,
  "found": true,
  "_source": {
    "titulo": "Don Quijote de la Mancha",
    "autor": "Miguel de Cervantes",
    "generos": ["novela", "caballerias"]
  }
}
```

Ejemplo adaptado de: https://www.elastic.co/es/blog/a-practical-introduction-to-elasticsearch

Conceptos

- Índices
  - Permiten buscar documentos de un tipo determinado.
    - Sólo se permite 1 tipo de documento en un mismo índice.
  - Contienen el esquema de los datos (documentos).

- Equivalencias **aproximadas**:

Esquema relacional	MongoDB	Elasticsearch
Tabla	Colección	Índice
Fila (de una tabla)	Documento	Documento

Conceptos

- Elasticsearch utiliza índices invertidos.
  - Comparativa de índices:

Índice directo

Índice	Contenido
1	que es esto
2	esto es un texto
3	este texto es otro texto

Índice invertido

Índice	Contenido
que	1
es	1, 2, 3
esto	1, 2
un	2
texto	2, 3
este	3
otro	3

ID de los documentos que contienen la palabra

Conceptos

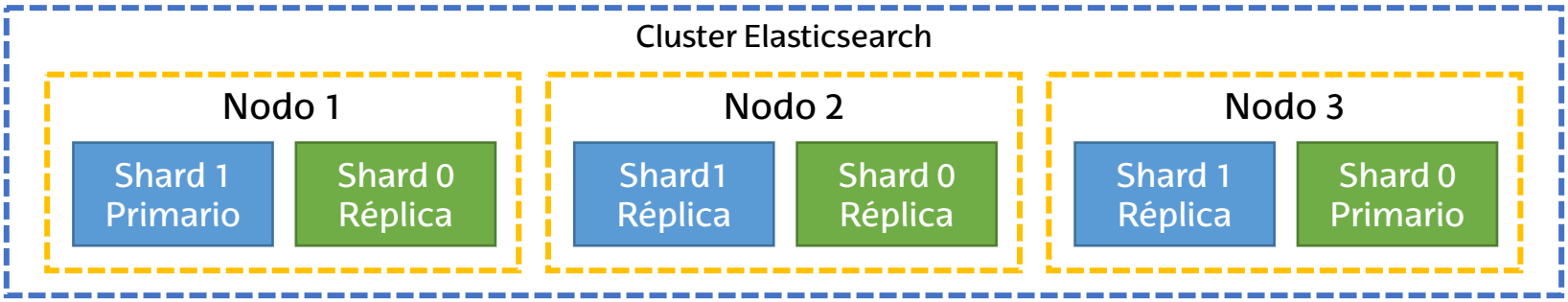
- Índices invertidos:
  - Ventajas:
    - Velocidad de búsqueda.
    - Facilidad para aplicar algoritmos de relevancia.
    - Facilidad para aplicar analizadores de texto.
  - Inconvenientes:
    - Velocidad de indexación.
    - Algunas búsquedas son costosas (p.e. Not lógico).

Conceptos

- Shard
  - Es una instalación de Elasticsearch en un nodo.
    - P.e., en una máquina virtual o contenedor.
  - Varias instalaciones (shards) forman un cluster Elasticsearch.
    - Objetivo: mayor rendimiento y tolerancia a fallos.
  - Un Shard es una instancia completa de un índice Lucene
    - Es un motor de búsqueda en sí mismo.
    - Indexa y administra las consultas para un subconjunto de los datos.

Conceptos

- Shard
  - Cada índice puede tener asignados Shards primarios y réplicas.
    - *Las consultas se redirigen al Shard apropiado:*
    - Las escrituras se envían a Shards primarios.
    - Las lecturas se envían a primarios o a cualquier réplica.
  - Ejemplo
    - 2 Shards primarios y 2 réplicas (por cada Shard):



Conceptos

- Mapping
  - Es la definición de un esquema para un índice
    - Indica el formato de los datos a indexar.
  - Elasticsearch puede inferir tipos automáticamente
    - Aunque a veces podemos querer definirlos a mano.
  - Ejemplo:
    - Fichero mapping-libros.json:
      - El nombre del índice se define en otra parte.

```
{
  "mappings" : {
    "properties" : {
      "titulo" : {"type": "text" },
      "autor" : {"type": "text" },
      "isbn" : {"type": "integer" }
    }
  }
}
```

Conceptos

- Mapping
  - Para cada campo se puede definir:
    - El tipo de datos: text, byte, short, integer, long, float, double, boolean, date

```
{ "properties" : {
    "precio" : {"type": ”float" }
}}
```

    - Activar/bloquear indexación como texto: analyzed, not_analyzed, no

```
{ "properties" : {
    "precio" : {"index": ”not_analyzed" }
}}
```

    - Analizador de texto a utilizar: standard, whitespace, simple, …

```
{ "properties" : {
    "descripcion" : {"analyzer": ”english" }
}}
```

Operaciones CRUD

- Crear un índice con un mapping:
  - Utilizando curl para enviar la petición

```
curl -H "Content-Type: application/json" -XPUT
<IP-servidor>:<puerto>/<nombre-indice> <definición-mapping>
```

    - Dos formas de indicar ‹definición-mapping›:
      - --data-binary @‹fichero› Con un fichero adjunto
      - -d '‹mapping›' Escribirlo como parte del comando
        - Ejemplo: -d '{"mappings": ... }'

  - Ejemplo:

```
curl -H "Content-Type: application/json" -XPUT
127.0.0.1:9200/libros --data-binary @mapping-libros.json
```

"libros" como nombre del índice

    - Devuelve un JSON indicando que se ha creado correctamente:

```
{"acknowledged":true, "shards_acknowledged":true, "index":"libros"}
```

Operaciones CRUD

- Carga de un documento
  - Utilizando curl:

```
curl -H "Content-Type: application/json" -XPOST
<IP>:<puerto>/<índice>/_doc/<ID-documento> --data-binary @<fichero>
```

Opcional. Si no lo incluimos, se genera uno aleatorio

  - Ejemplo:
    - Fichero cuento-de-navidad.json

```
{
  "titulo": "Cuento de Navidad",
  "autor": "Charles Dickens",
  "isbn": "798-..."
}
```

    - Ejemplo:

```
curl -H "Content-Type: application/json" -XPOST
127.0.0.1:9200/libros/_doc/5 --data-binary @cuento-de-navidad.json
```

Elegimos 5 como ID para el nuevo libro

Operaciones CRUD

- Operaciones de consulta:
  - *Ejemplos con 127.0.0.1:9200 como IP:Puerto y* libros *como índice.*
  - Recuperar índices existentes:

```
curl 127.0.0.1:9200/_cat/indices?v
```

  - Leer el mapping de un índice:

```
curl -XGET 127.0.0.1:9200/libros/_mapping
```

  - Recuperar número de documentos en un índice:

```
curl -XGET 127.0.0.1:9200/libros/_count?pretty
```

  - Recuperar los documentos de un índice:

```
curl -XGET 127.0.0.1:9200/libros/_search?pretty
```

Operaciones CRUD

- Cargar múltiples documentos a la vez (carga "en bruto"):

```
curl -H "Content-Type: application/json" -XPUT <IP>:<puerto>/<índice>/_bulk
--data-binary @<fichero>
```

  - Ejemplo:

```
curl -H "Content-Type: application/json" -XPUT '127.0.0.1:9200/libros/_bulk'
--data-binary @muchos-libros.json
```

  - Cada nuevo elemento a cargar debe tener 2 objetos JSON:
    - Uno con el ID para el índice
    - Uno con los datos y el ID recién creado
      - Ejemplo con 2 documentos nuevos:

```
{"create" : {"_index": "libros", "_id" : "10" } }
{"id" : "10", "titulo" : "David Copperfield", "autor" : "Charles Dickens"}
{"create" : {"_index": "libros", "_id" : "11" } }
{"id" : "11", "titulo" : "Pinocho", "autor" : "Carlo Collodi"}
…
```

Operaciones CRUD

- Modificar documentos
  - En Elasticsearch no es posible modificar un documento ya cargado.
    - Un documento ya indexado es inmutable.
  - Cada documento tiene un campo "_version"
  - Para modificar un documento existente, se crea un nuevo documento:
    - Se copia del original con las modificaciones solicitadas
    - Se actualiza el campo "_version"
    - Se marca el documento original para ser eliminado.

Operaciones CRUD

- Modificar documentos
  1) Definir un objeto JSON con los campos a modificar.
  2) Utilizar Curl para realizar la modificación.
     - Debemos indicar el ID del documento a modificar.

```
curl -H "Content-Type: application/json" -XPOST
<IP>:<puerto>/<índice>/_update/<ID> --data-binary @<fichero>
```

- Ejemplo:
  - Fichero modificar-libro.json

```
{
  "doc" : {
    "titulo" : "Las aventuras de Pinocho" }
}
```

  - Comando:

```
curl -H "Content-Type: application/json" -XPOST
127.0.0.1:9200/libros/_update/11 --data-binary @modificar-libro.json
```

El índice se indica en la consulta

Operaciones CRUD

- Borrar documentos:
  - Utilizando Curl:

```
curl -XDELETE <IP>:<puerto>/<índice>/_doc/<ID>
```

  - Ejemplo:
    - Borrar el documento con ID 5 del índice “libros”:

```
curl -XDELETE 127.0.0.1:9200/libros/_doc/5
```

- Borrar índice:
  - Utilizando Curl:

```
curl -XDELETE <IP>:<puerto>/<índice>
```

Operaciones CRUD

- Resumen de comandos con ejemplos
  - https://documenter.getpostman.com/view/5283544/S1LyUnSc

Ejercicio 2

- Crear un índice “peliculas” con el siguiente mapping:
  - Campo “titulo”, tipo text.
  - Campo “director”, tipo text.
  - Campo “año”, tipo integer.

- Cargar los siguientes datos al índice:

```
{"create" : {"_index": "peliculas", "_id" : "1" } }
{"id" : "1", "titulo" : "El padrino", "director" : "Francis Ford Coppola", "año": 1972}
{"create" : {"_index": "peliculas", "_id" : "2" } }
{"id" : "2", "titulo" : "Gladiator", "director" : "Ridley Scott", "año": 2030}
{"create" : {"_index": "peliculas", "_id" : "3" } }
{"id" : "3", "titulo" : "Inception", "director" : "Christopher Nolan", "año": 2010}
```

- *Continua en la siguiente diapositiva*

Ejercicio 2

- Modificar el campo “año” de la película “Gladiator”.
  - Debería ser 2000 en lugar de 2030.
- Borrar el documento correspondiente a la película “Inception”.
- Verificar que los cambios realizados al índice son correctos.

Búsquedas

- Elasticsearch está optimizado para búsquedas de texto.
  - Permite recuperar documentos cuyos campos coincidan con unos criterios de búsqueda.
    - P.e. que el campo “titulo” contenga “libro de la selva”.

- Hay 2 formas de limitar los resultados de búsqueda.
  - Diferentes tipos de consulta
    - Devuelven los resultados en orden de relevancia.
  - Filtros
    - Incluyen o eliminan documentos de los resultados.

Búsquedas

- Se utiliza el endpoint *_search*.
  - Los términos y configuración de búsqueda se escriben en JSON.

- Ejemplo:
  - Buscar documentos en el índice “libros” que contengan la palabra “roja” en el campo “titulo”.

```
curl -H "Content-Type: application/json" -XGET
'127.0.0.1:9200/libros/_search?pretty' -d '
{ "query":
  { "match":
    { "titulo": "roja" }
  }
}'
```

Formatea los resultados de forma legible en la terminal

Búsquedas

- Tipos de consultas
  - Devuelve todos los documentos
    - Se suele usar junto con filtros (más adelante).

```
{"match_all": {}}
```

  - Coincidencia
    - Devuelve campos que contengan los términos buscados.

```
{"match": {"titulo": "roja"}}
```

  - Coincidencia múltiple
    - Buscar el mismo término en diferentes campos

```
{"multi_match": {"query": "roja", "fields": ["titulo", "descripcion"]}
```

Búsquedas

- Tipos de consultas
  - Coincidencia de frase
    - Busca los términos para el campo indicado en el orden indicado.

```
{"match_phrase": {"descripcion": "la casa era verde"}}
```

    - Se puede añadir el parámetro “slop”: indica un umbral variación en el orden de los términos escritos.

    - Ejemplo:
      - Encontrar documentos con “casa era verde” o “casa verde era”

```
{"match_phrase": {
   "descripcion": {"query": "la casa era verde", "slop" : 1}}
}
```

      - Si slop fuese 2, encontraría p.e. “la verde casa”.

Búsquedas

- Tipos de consultas
  - Combinación booleana
    - Permite combinar diferentes tipos de consulta utilizando los términos “must”, “must_not” y“should”
      - “must” equivale al AND lógico, “should” y al OR lógico.
    - Ejemplo:

```
curl … - d ’
{
  "query": {
    "bool": {
      "must": [ { "match": { "title": "Verde"}}, { "match": { "anyo": "1928"}} ],
      "must_not": { "match": { "autor": "Jose"}}
    }
  }
}’
```

Búsquedas

- Filtros:
  - Por valor exacto:

```
{ "term": { "anyo": 2014 }}
```

  - Por uno de los valores de un listado:

```
{ "term": { "genero": ["novela","crimen"] }}
```

  - Números/fechas en un rango concreto:
    - gt (mayor), gte (mayor o igual), lt (menor), lte (menor o igual)

```
{ "range": { "anyo": {"gte": 2010 }} }
```

  - Si el campo existe en el documento:

```
{ "exists": {"field": "editorial"} }
```

Búsquedas

- Filtros:
  - Si el campo no existe en el documento:

```
{ "missing": {"field": "editorial"} }
```

  - Combinaciones booleanas.
    - Funcionan igual que para las consultas
  - Ejemplo:

```
curl … /search - d'
{
  "query": {
    "bool": {
      "must": { "match": { "title": "roja"}},
      "filter": { "range" : { "anyo" : { "gte" : 1950 }}}
}}}
```

Búsquedas

- Se devuelve un JSON con un listado de “hits”.
  - Documentos que han encajado con los términos de búsqueda.
  - Se ordenan mediante el campo “_score”.
    - Indica la relevancia de cada documento para la búsqueda[1].
  - Ejemplo:
    - Fragmento del resultado de la búsqueda { “title” : “roja” }.

```
"hits" : [
  {
    "_index" : "libros",
    …
    "_score" : 13.889601,
    "_source" : {
      …
      "titulo" : "Caperucita roja"
…
```

[1] “Understanding Elasticsearch Scoring and the Explain API”: https://opster.com/guides/elasticsearch/search-apis/elasticsearch-scoring-understanding-the-explain-api/

Ejercicio 3

- Descargar el dataset "Obras de Shakespeare" de eGela:
  - Es un fragmento del dataset de muestra oficial "Obras completas"[1]:
  - Esquema:

```
{
  "speaker" : {"type": "keyword" },
  "play_name" : {"type": "keyword" },
  "line_id" : { "type" : "integer" },
  "speech_number" : { "type" : "integer" }
}
```

  - Más información:
    - https://www.elastic.co/guide/en/kibana/5.5/tutorial-load-dataset.html

- *Ver siguiente diapositiva*

[1]Dataset "Obras completas de Shakespeare": https://download.elastic.co/demos/kibana/gettingstarted/shakespeare.json

Ejercicio 3

- Crear un índice llamado “shakespeare” siguiendo el esquema de la diapositiva anterior.
- Cargar el dataset de las obras de Shakespeare en el índice “shakespeare”.
- Encontrar:
  - A qué obra de Shakespeare pertenece la frase “To be or not to be” y en qué línea de la obra se encuentra.
  - Cuántas frases tiene el personaje "OCTAVIUS CAESAR" en la obra "Antony and Cleopatra”.

Bibliografía

- Frank Kane. “Elasticsearch 7 and the Elastic Stack: In Depth and Hands On”, Udemy, 2020[1]:
  - https://www.udemy.com/course/elasticsearch-7-and-elastic-stack
- Alejandro Marqués, “Elastic Search - Capítulo I”, Paradigma Digital, 2019[1]:
  - https://youtu.be/UIN2NeMb7xc
- Elasticsearch v8.5 Guide, 2022[2]:
  - https://www.elastic.co/guide/en/elasticsearch/reference/current/index.html
- Consultados en noviembre 2020[1] y noviembre 2022[2].