Repositorios de información

Recuperación de información /
Una introducción “gentil” a *Elasticsearch* (1ª parte)

Aviso para navegantes

- *Elasticsearch* (ES) es una herramienta para crear sistemas de recuperación de información escalables y distribuidos basada en la biblioteca Lucene.
- Por lo general, ES se despliega en un clúster con múltiples nodos, cada nodo tendrá múltiples fragmentos de un índice—*shards*—y esos fragmentos, además, pueden tener múltiples réplicas.
- No vamos a ver esas funciones de ES—aunque puedes explorarlas como parte de tus prácticas.
- Vamos a usar ES como una herramienta conveniente que nos permita comprender algunas cuestiones básicas en recuperación de información como los procesos de indexado y consulta, así como la idea de relevancia de los documentos.

Lo primero es lo primero

- Vete a https://www.elastic.co/es/downloads/elasticsearch
- Descarga el ZIP de ***Elasticsearch*** para Windows en tu directorio de usuario.
- Descomprime dicho archivo.
- Deberías tener la siguiente estructura de archivos y directorios:

```
./bin
./config
./jdk
./lib
./logs
./modules
./plugins
LICENSE.txt
NOTICE.txt
README.asciidoc
```

¡Atención! En los ordenadores de la Escuela no utilices la última versión de *ElasticSearch* sino la **versión 8.2.3** disponible **aquí**.

Lo primero es lo primero

- Abre una ventana de órdenes y ejecuta `./bin/elasticsearch.bat`
- Debería producir una salida bastante verbosa incluyendo lo siguiente:

```
Password for the elastic user (reset with
`bin/elasticsearch-reset-password -u elastic`):
u*wmb02yhThyqXjm27fa
```

- **Anota esa password**, la vas a necesitar más tarde…
- Si se te olvidara puedes generarla de nuevo con la siguiente orden

```
./bin/elasticsearch-reset-password -u elastic
```

Presta atención a los mensajes de *ElasticSearch*. Podría ser que no tuvieras espacio suficiente en disco para ejecutarlo. Si fuera el caso libera espacio o "despliega" *Elastic* en un disco duro diferente.

Lo primero es lo primero

- Abre en un navegador la siguiente URL `https://127.0.0.1:9200/`
- Primero, debes aceptar que la conexión no es privada porque aunque usa HTTPS utiliza un certificado autofirmado…
- A continuación debes introducir `elastic` como usuario y la contraseña que anotaste antes…
- Entonces debería aparecer algo como esto…

Lo primero es lo primero

Salida típica de https://127.0.0.1:9200/


```
{
  "name" : "DESKTOP-9SD4DVV",
  "cluster_name" : "elasticsearch",
  "cluster_uuid" : "WMHnf1iHR1CnwPCPbjAU_Q",
  "version" : {
    "number" : "8.4.3",
    "build_flavor" : "default",
    "build_type" : "zip",
    "build_hash" : "42f05b9372a9a4a470db3b52817899b99a76ee73",
    "build_date" : "2022-10-04T07:17:24.662462378Z",
    "build_snapshot" : false,
    "lucene_version" : "9.3.0",
    "minimum_wire_compatibility_version" : "7.17.0",
    "minimum_index_compatibility_version" : "7.0.0"
  },
  "tagline" : "You Know, for Search"
}
```


Por si acaso, esto es JSON.

Órdenes en *Elasticsearch*

- Para invocar órdenes en ES se utiliza HTTP (GET, POST, PUT, DELETE) y parámetros JSON.
- Ventajas:
  - Se puede utilizar ES con cualquier lenguaje de programación que tenga soporte para consumir servicios web REST.
- Desventajas:
  - Usar ES a "bajo nivel" (por ejemplo, con `curl`, sin ningún lenguaje de programación o front-end) es tedioso y poco práctico.
  - ~~Cualquiera con acceso a la URL de un clúster ES puede ejecutar órdenes (por ejemplo, puede modificar o eliminar un índice completo).~~ Ya no, ahora ES está "securizado" por defecto.

Front-ends para *ElasticSearch*

- Necesitamos un front-end gráfico para trabajar con ES. La verdad es que ha habido muchos pero pocos sobreviven o funcionan con las últimas versiones de ES.
- *Kibana* es la interfaz oficial para ES, **pero** usarla para estas prácticas y los entregables sería **matar moscas a cañonazos**.
- *Kaizen* y ***Elasticvue*** son front-ends mucho más sencillos. La primera es una aplicación de escritorio y la segunda es una aplicación web. Nosotros usaremos ***Elasticvue***.
- Para usarlo haz lo siguiente:
  - Detén *Elasticsearch*.
  - Abre en un editor de texto el archivo `config/elasticsearch.yml`.
  - Al final de ese archivo añade las líneas que aparecen a la izquierda de esta diapositiva.
  - Graba el archivo.
  - Reinicia *Elasticsearch*.
  - Vete a `https://app.elasticvue.com/` y trata de conectarte a tu "cluster". Recuerda usar HTTPS **y** `127.0.0.1` en vez de `localhost`.

```
# allow CORS requests from
https://app.elasticvue.com

http.cors.enabled: true

http.cors.allow-origin:
"https://app.elasticvue.com"


# and if your cluster uses
authorization:

http.cors.allow-headers:
X-Requested-With,Content-Type,Co
ntent-Length,Authorization
```

Hora de crear un índice

- El índice es lo que en las clases de teoría denominamos el **“fichero invertido”**.
- Obviamente, para crear uno necesitamos una **colección de documentos.**
- ES puede indexar cualquier tipo de documento pero al final siempre se transforman en JSON (podría ser un documento JSON con un solo campo cuyo contenido es el texto plano del documento original).
- En aras de la simplicidad, vamos a descargar una colección que ya está formada por documentos JSON.

La colección de documentos

- Vete a https://bit.ly/3o5BLSe
- **Descarga** y **descomprime** el archivo `tweets-20090624-20090626-en_es-10percent.ndjson.gz` (547 MB y 2,7 millones de tuits).
- El archivo contiene el 10% de todos los tuits publicados en inglés y español los días 24, 25 y 26 de junio de 2009.
- La estructura de los documentos se muestra en la siguiente diapositiva.

Un archivo *ndjson* es un archivo de texto que no es JSON válido pero que contiene un documento JSON por línea.

Si quisieras todos los tuits en inglés y español correspondientes a ese período vete a la URL https://bit.ly/3BN0qzK, descarga y descomprime el archivo `tweets-20090624-20090626-en_es.ndjson.gz`.

Ocupará 5,34 GB y contendrá 26,5 millones de tuits; indexarlo debería llevar unos 25 minutos.

Estructura de los documentos

```
{
  "created_at":"Wed Jun 24 00:25:42 +0000 2009",
  "id_str":"2302882806",
  "lang":"en",
  "text":"best dinner evar + playing outside with the frog",
  "user_id_str":"15622439"
}
```

Estos objetos contienen tan solo el mínimo de componentes de un tuit, la estructura real es mucho más extensa (véase).

Este ejemplo no contiene los campos `in_reply_to_user_id_str` ni `in_reply_to_status_id_str` que nos permitirían crear un grafo con las respuestas a tuits de otros usuarios.

Antes de indexar la colección

- Para indexar la colección necesitamos invocar las órdenes necesarias en ES.
- Podríamos tratar de hacerlo con el front-end pero es mucho más razonable escribir un script en algún lenguaje de programación.
- El lenguaje en cuestión no es muy importante pero vamos a usar *Python*, ¡sí, Python!
    - Para ello usaremos el software que se usa habitualmente en "Fundamentos de Informática": *Python* y *PyScripter* (en tu propio ordenador descarga las últimas versiones de cada uno).
- Toda la información sobre ES y Python está disponible aquí:
    - https://www.elastic.co/guide/en/elasticsearch/client/python-api/current/index.html
    - https://elasticsearch-py.readthedocs.io/en/v8.4.3/index.html

Usando ES con *Python*

- En una ventana de órdenes ejecuta lo siguiente (el parámetro `--user` solo es necesario si hubiera más usuarios de Python en la máquina):

```
pip install --user elasticsearch
```

- Ejecuta *PyScripter* y crea un nuevo script (`File|New|New Python module`)
- Importa *ElasticSearch*:

```
from elasticsearch import Elasticsearch
```

- Ejecuta el script (debería funcionar sin problemas).
- **¡Atención!** Si tuvieras diferentes versiones de *Python* instaladas en la máquina asegúrate de que el módulo para ES se instala en el entorno de la versión que vayas a usar con *PyScripter*.

Indexando la colección

Descarga el script **`indexer.py`**
(por favor, asegúrate de que está en la misma carpeta que el archivo *ndjson*),
revísalo pero **no lo ejecutes.**

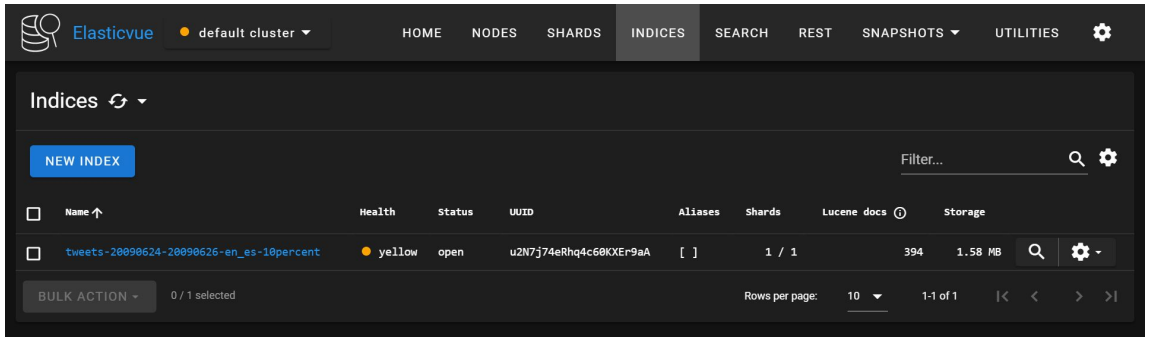
Indexando la colección

- Ahora ES está “securizado” por defecto así que tenemos que lidiar con ello—es una idea terrible tener instancias de ES inseguras.
- El enfoque más sencillo, aunque no el mejor, es autenticarse usando el usuario `elastic` y la contraseña proporcionada por ES. Puedes consultar otras opciones aquí: *Connecting | Elasticsearch Python Client [8.4]*.
- Hay otro problema: estamos usando un certificado autofirmado—que tampoco es una buena idea—así que necesitamos una forma de decirle a Python que se “fíe” del mismo:
  - El certificado CA raíz proporcionado por ES, `http_ca.crt`, está en el directorio `config/certs` dentro del directorio de ES.
  - Copia ese archivo, `http_ca.crt`, a la carpeta donde has descargado `indexer.py`.
  - **Recuerda modificar `indexer.py` para que use la password de tu instancia ES.**

Este es un ejemplo de “juguete”. No es habitual que el cliente se ejecute en la misma máquina donde se ejecuta ES. Por eso no estamos accediendo desde el script al archivo `http_ca.crt` que está en el directorio de ES. Además, no te conectarías a un nodo concreto sino a un cluster…

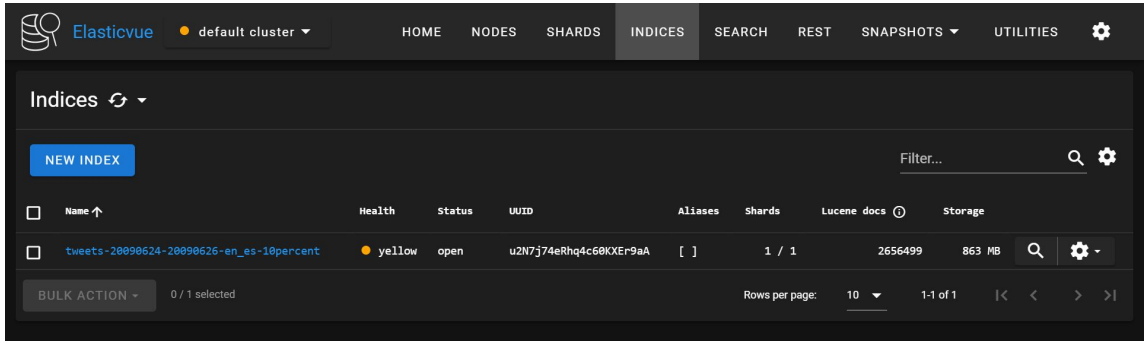
Indexando la colección

- **Ahora ya puedes ejecutar `indexer.py`**
- Vete a la vista `INDICES` en ***Elasticvue***, debería aparecer algo similar a esta imagen—tal vez tengas que refrescar el índice, la opción para ello está en el menú en la rueda dentada ⚙️correspondiente al índice.
- ¿Qué problema tiene este script?



Indexando la colección

- Detén el script.
- Descarga el script [`bulk-indexer.py`], revísalo, **modifica la password para el usuario `elastic`** y ejecútalo.
- Vete a la vista `INDICES` en ***Elasticvue***, debería aparecer algo parecido a esta imagen en unos pocos minutos (<5).



Problemas de espacio en disco

- El script podría detenerse y dar distintos errores, por ejemplo:
  - `TransportError: TransportError(429, 'cluster_block_exception', 'index [tweets-20090624-20090626-en_es-10percent] blocked by: [TOO_MANY_REQUESTS/12/disk usage exceeded flood-stage watermark, index has read-only-allow-delete block];')`
  - `elasticsearch.helpers.BulkIndexError: 192354 document(s) failed to index.`
  - `elasticsearch.ApiError: ApiError(429, 'circuit_breaking_exception', '[parent] Data too large, data for [<http_request>] would be [2775344040/2.5gb], which is larger than the limit of [171547033/163.5mb], real usage: [2775344040/2.5gb], new bytes reserved: [0/0b], usages [fielddata=0/0b, request=0/0b, inflight_requests=0/0b, model_inference=0/0b, eql_sequence=0/0b]')`

Problemas de espacio en disco

- Para “resolver” esos problemas haz lo siguiente:
  - En *Elasticvue*, borra el índice y ve a la vista REST.
  - Escoge PUT como HTTP `Method`.
  - Escribe `_cluster/settings` en el `Path`.
  - Usa la siguiente carga JSON:

```
{
  "persistent": {
    "indices.breaker.fielddata.limit": "100%",
    "indices.breaker.request.limit": "100%",
    "network.breaker.inflight_requests.limit": "100%",
    "indices.breaker.total.limit": "100%",
    "cluster.routing.allocation.disk.watermark.low": "95%",
    "cluster.routing.allocation.disk.watermark.high": "95%",
    "cluster.routing.allocation.disk.watermark.flood_stage": "95%"
  }
}
```

  - Pulsa el botón `SEND REQUEST`.
  - Ejecuta el script (`bulk-indexer.py`) de nuevo.

Si aún así no tienes espacio suficiente vas a tener que liberar espacio o tener *ElasticSearch* en otro disco con espacio suficiente.

Lanzando consultas con *Elasticvue*

- Vete a la vista REST en *Elasticvue*.
- Algunas consultas GET:
  - `tweets-20090624-20090626-en_es-10percent/_search?q=cristiano ronaldo`

    214 hits eq
  - `tweets-20090624-20090626-en_es-10percent/_search?q=iran`

    10000 hits gte ???
  - `tweets-20090624-20090626-en_es-10percent/_search?q=text:iran AND lang:en`

    10000 hits gte ???
  - `tweets-20090624-20090626-en_es-10percent/_search?q=text:iran AND lang:es`

    98 hits eq
  - `tweets-20090624-20090626-en_es-10percent/_search?q=user_id_str:2467791`

    7 hits eq

¡Atención! `text`, `lang` y `user_id_str` son campos de los documentos, en este caso de los tuits; no son parámetros *Elasticsearch*.

Lanzando consultas con *Elasticvue*

- En los ejemplos anteriores tenemos consultas que retornan un número de resultados anotado con **`eq`** y otras que retornan un número sospechosamente redondo de 10.000 resultados anotado con **`gte`**.
- En este segundo caso ES no está retornando el número real de documentos en el índice que satisfacen la consulta sino que solo nos indica que son más de 10.000.
- Tenemos dos opciones para conocer el número real de resultados...

Lanzando consultas con *Elasticvue*

- Usando el API `_count`:
  - `tweets-20090624-20090626-en_es-10percent/_count?q=iran`

    16900 hits
  - `tweets-20090624-20090626-en_es-10percent/_count?q=text:iran AND lang:en`

    16802 hits

Lanzando consultas con *Elasticvue*

- Con el API `_search` y una petición JSON; es decir, usando el método P0ST:

```
tweets-20090624-20090626-en_es-10percent/_search
{
      "size" : 0,
      "track_total_hits" : true,
      "query" : {
          "match" : {
               "text": "iran"
          }
      }
}
```

16900 hits

Lanzando consultas con *Elasticvue*

- Con el API `_search` y una petición JSON; es decir, usando el método P0ST:

```
tweets-20090624-20090626-en_es-10percent/_search
{
      "size" : 0,
      "track_total_hits" : true,
      "query" : {
          "query_string" : {
                "query": "text:iran AND lang:en"
          }
      }
}
```

16802 hits

El lenguaje de consulta de *Elastic*

- *Elasticsearch* proporciona un DSL (*Domain Specific Language*) para describir nuestras consultas (véase).
- Sin embargo, puesto que estamos usando ES en un contexto de recuperación de información nos interesan únicamente las consultas de texto completo, en particular `query_string` o, aún más sencillas, `simple_query_string`.
- Algunos ejemplos:

Operador *and* (+ cuando se use `simple_query_string`)

```
{
      "track_total_hits" : true,
      "query" : {
          "simple_query_string" : {
            "query": "china + google"
          }
      }
}
```

Recuerda usar la URL `tweets-20090624-20090626-en_es-10percent/_search`

El lenguaje de consulta de *Elastic*

- *Elasticsearch* proporciona un DSL (*Domain Specific Language*) para describir nuestras consultas (véase).
- Sin embargo, puesto que estamos usando ES en un contexto de recuperación de información nos interesan únicamente las consultas de texto completo, en particular `query_string` o, aún más sencillas, `simple_query_string`.
- Algunos ejemplos:

Búsqueda por frase exacta

```
{
      "track_total_hits" : true,
      "query" : {
          "simple_query_string" : {
            "query": "\"liu xiaobo\""
          }
      }
}
```

Recuerda usar la URL `tweets-20090624-20090626-en_es-10percent/_search`

El lenguaje de consulta de *Elastic*

- *Elasticsearch* proporciona un DSL (*Domain Specific Language*) para describir nuestras consultas (véase).
- Sin embargo, puesto que estamos usando ES en un contexto de recuperación de información nos interesan únicamente las consultas de texto completo, en particular `query_string` o, aún más sencillas, `simple_query_string`.
- Algunos ejemplos:

  Operador *or* (`|` cuando se use `simple_query_string`) y frase exacta

```
{
      "track_total_hits" : true,
      "query" : {
          "simple_query_string" : {
            "query": "\"south waziristan\" | pakistan"
          }
      }
}
```

Recuerda usar la URL `tweets-20090624-20090626-en_es-10percent/_search`

El lenguaje de consulta de *Elastic*

- *Elasticsearch* proporciona un DSL (*Domain Specific Language*) para describir nuestras consultas (véase).
- Sin embargo, puesto que estamos usando ES en un contexto de recuperación de información nos interesan únicamente las consultas de texto completo, en particular `query_string` o, aún más sencillas, `simple_query_string`.
- Algunos ejemplos:

  Operador *or* (`|` cuando se use `simple_query_string`) y frase exacta

```
{
      "track_total_hits" : true,
      "query" : {
          "simple_query_string" : {
            "query": "\"michael jackson\" | \"king of pop\""
          }
      }
}
```

Recuerda usar la URL `tweets-20090624-20090626-en_es-10percent/_search`

Highlighting

- Los buscadores proporcionan ***snippets***—breves resúmenes—en los resultados.
- Además de eso, se puede pedir a ES que **"destaque" aquellos tokens del *snippet* que han hecho *"match"* con la consulta** de tal modo que sea más sencillo para la persona usuaria determinar si el resultado es relevante o no para su necesidad de información.

```
tweets-20090624-20090626-en_es-10percent/_search
{
    "size":50,
    "query": {
        "simple_query_string": {
            "query":"\"michael jackson\" | \"king of pop\""
        }
    },
    "highlight": {
        "fields": {
            "text": {}
        }
    }
}
```

Obteniendo sugerencias


```
"options": [
        {
          "text": "pregnant",
          "score": 0.875,
          "freq": 857
        },
        {
          "text": "preganant",
          "score": 0.7777778,
          "freq": 1
        },
        {
          "text": "pergnant",
          "score": 0.75,
          "freq": 1
        },
        {
          "text": "pregnanc",
          "score": 0.75,
          "freq": 1
        },
        {
          "text": "pregnaut",
          "score": 0.75,
          "freq": 1
        }
      ]
```


- Durante las clases de teoría vimos que **entre el 10% y el 15% de las consultas enviadas a los buscadores contienen faltas de ortografía y errores tipográficos**. ES ofrece una forma sencilla de solicitar **sugerencias** que explotan los contenidos de nuestro índice.
- Veamos cómo podemos obtener sugerencias para la palabra `prregnant` (sic).

```
tweets-20090624-20090626-en_es-10percent/_search
{
  "suggest" : {
    "text" : "prregnant",
    "Suggestion" : {
      "term" : {
        "field" : "text"
      }
    }
  }
}
```

Enviando consultas desde *Python*

- El cliente Python para ES permite especificar los parámetros de las consultas usando la misma estructura que el API REST pero usando diccionarios en lugar de JSON.
- Descarga y revisa el script **`query.py`**—este script no tiene por qué estar en la misma carpeta que el archivo *ndjson*.
- **A tener en cuenta:** es una pésima idea usar el superusuario `elastic` solo para consultar un índice. Además, si el contenido del índice no va a cambiar debería cambiarse a solo-lectura.
- Para crear un usuario solo para realizar consultas se podría hacer lo siguiente `bin/elasticsearch-users useradd lectura -p abretesesamo -r viewer`
- Juega un poco con las consultas de ejemplo para comprobar qué resultados retorna ES. Comprueba también que este usuario no puede crear índices ni añadir documentos a los mismos.
- Por favor, lee la documentación.

Otros parámetros útiles: `default_operator`, `docvalue_fields`, ...

Tarea: Paginación de resultados

- La forma más sencilla de paginar los resultados de búsqueda es usando los parámetros `from` (valor por defecto: 0) y `size` (valor por defecto: 10).
- El mayor problema de esto es que resulta ineficiente y, además, solo se pueden obtener un máximo de 10.000 resultados.

  ```
  Result window is too large, from + size must be less than or equal to: [10000] but was [???]. See the scroll api for a more efficient way to request large data sets. This limit can be set by changing the [index.max_result_window] index level setting.
  ```

- Por favor, revisa la siguiente documentación:
  - Paginate search results.
  - Scroll search results.
  - Search after.
  - Scan.
- Tarea: escribe un script para volcar todos los documentos que satisfagan una consulta con muchos más de 10.000 resultados (p.ej., `/_search?q=michael jackson`).
- **Pistas:** usa **`helpers.scan`**, ten en cuenta que puede haber caracteres Unicode.

Repositorios de información

Recuperación de información /
Una introducción “gentil” a *Elasticsearch* (Fin de la 1ª parte)