ELK: elasticsearch (API y DSLQuery)

1.3_api_dsl_query.pdf

- API = Application Programming Interface. Capa abstracción funciones ofrecidas por la apliación

Cluster APIs

- **Health**: `GET _cluster/health?pretty`
  - status
    - green: todos los shard bien
    - yellow: no están bien las réplicas
    - red: shard primario no indexado
- **Stats**: `GET _cluster/stats?human&pretty`
  - estadísticas del cluster (número de shards, almacenamiento, uso de memoria)
  - información sobre los nodos (cantidad, roles, os, JVM, memoria, cpu, plugins)
- **Info**: `GET _nodes?pretty` → `curl -XGET <IP>/_nodes`
  - Información de todos los nodos del cluster.
  - `GET _nodes/nodeId1`
    - información concreta de uno de los nodos
  - nombre de host, puerto http, IP, OS, procesos, plugins...

Documents APIs

- Index:
  - añade o actualiza un documento específico
  - ```
    curl -XPUT "localhost:9200/twitter/tweet/1?pretty" -H
    "Content-Type: application/json" -d"
      {
        "user":"openwebinars",
        "post_date": "2009-11-15T14:12:12"
        "message": "pruebaq en Elasticsearch"
      }"
    ```
  - bdd: twitter
  - documento: 1
  - formato: json
- Get:
  - permite consultar documentos JSON de un índice con su ID
  - ```
    curl -XGET 'localhost:9200/twitter/tweet/1?pretty'
    ```
- Delete:
  - eliminar documentos con su ID
  - ```
     curl -XDELETE 'http://localhost:9200/twitter/tweet/6?pretty
    ```
- Update:
  - actualiza la información de un documento. Consulta, actualiza y vuelve a indexar
  - ```
    curl -XPOST 'localhost:9200/twitter/tweet/5/_update?pretty' -H
    ```

```
'Content-Type: application/json' -d'
{
  "script": "ctx._source.new_field='valor nuevo campo' "
}'
```

Search APIs

- URI:
  - en la propia consulta:

```
curl -XGET
'localhost:9200/twitter/tweet/_search?q=user:openwebinars&pretty'
```

  -
```
curl -XGET 'localhost:9200/twitter/tweet/_search?q=new_field:valor nuevo
campo&pretty'
```

  - cuerpo completo:

```
curl -XGET 'localhost:9200/twitter/tweet/_search?pretty'
-H 'Content-Type: application/json' -d'
{
  "query": {
    "term":{"user":"user1"}
    }
}'
```

- Templates:
  - plantillas para búsquedas repetidas y complejas, cambiando parámetros

  -
```
curl -XGET 'localhost:9200/_search/template?pretty' -H 'Content-Type:
application/json' -d’ {
 "inline" : {
 "query": { "match" : { "{{my_field}}" : "{{my_value}}" } },
 "size" : "{{my_size}}"
 },
 "params" : {
 "my_field" : "user",
 "my_value" : "openwebinars",
 "my_size" : 1
 } } '
```

- Shards API
  - devuelve los índices y shards sobre los que se ejecutará la búsqueda. No hace la búsqueda en sí.
  - útil para búsqueda de errores u optimización de rendimiento

  -
```
curl -XGET 'localhost:9200/twitter/_serach_shards?pretty'
```

Index APIs

- Create: se pueden crear múltiples índices, incluyendo operaciones

- ```
  curl -XPUT 'localhost:9200/new_index?pretty' -H 'ContentType:
  application/json' -d'
   {
   "settings" : {
   "index" : {
   "number_of_shards" : 3,
   "number_of_replicas" : 2
   } } }'
  ```
  - new_index
- Delete:
- ```
  curl -XDELETE 'localhost:9200/new_index?pretty'
  ```
  - ojo con parámetros **_all** o *
  - parámetro **action.destructive_requires_name:true** para proteger borrados totales accidentales
- Open/Close
  - un índice cerrado no sobrecarga el cluster
  - ```
    curl -XPOST 'localhost:9200/new_index/_close?pretty'
    curl -XPOST 'localhost:9200/new_index/_open?pretty'
    ```
- Mapping:
  - devuelve el mapping que se está realizando sobre un índice
  - ```
    curl -XGET ‘localhost:9200/twitter/_mapping/tweet?pretty'
    ```
  - se obtiene información sobre los campos.
  - el mapping hace un mapeo de los campos para declararlos y posteriormente poder acceder de una manera más organizada a la información: txt, fecha, número...
- Stats:
  - estadísticas de los índices de diferentes operaciones
  - ```
    curl -XGET 'localhost:9200/_stats?pretty'
    ```
  - cantidad de documentos, almacenamiento, segmentos... (monitorización)

query DSL

- usar JSON para definir consultas DSL (Lenguaje de Dominio Específico)
- **query context**: calcula un _score que representa cómo de bien un documento hace match con la consulta en comparación con el resto (incluso podemos decirle que porcentaje de score permitimos)
  - match all: score 1.0 (100%)

    ```
    curl -XGET 'localhost:9200/twitter/_search?pretty' -
    H 'Content-Type: application/json' -d'
    {
     "query": {
     "match_all": {}
     }
    }'
    ```
  - match:

    ```
    curl -XGET 'localhost:9200/twitter/_search?pretty' -
    H 'Content-Type: application/json' -d'
    ```
    ```

```
{
 "query": {
 "match" : {
 "user" : "user2"
} } } '
```

  - regexp:
    - permite usar expresiones regulares. regexp pesadas pueden provocar un gran consumo de recursos

    - ```
      curl -XGET 'localhost:9200/twitter/_search?pretty' -H
      'Content-Type: application/json' -d'
      {
       "query": {
       "regexp":{
       "user": "user.*"
      } } } '
      ```

- **filter context**: Solo sí o no coíncide con la consulta
  - filtros usados a menudo se cachean para mejorar rendimiento
      ```