

Zadanie 5

Autor: Samuel Schmidt

AIS id: 103120

Github odkaz























[GitHub odkaz](#)

GitHub classroom: <https://github.com/FIIT-DBS/zadanie-pdt-schmidt-8>

Uloha 1

Zadanie: Rozbehajte si 3 inštancie Elasticsearch-u

Rozbehal som ich pomocou docker-compose scriptu, ktorý som našiel na webe a upravil. Vytvoril som .env, ktorý bol potrebný pre správne fungovanie.

<input type="checkbox"/>	NAME	IMAGE	STATUS	PORT(S)	STARTED	ACTIONS
<input type="checkbox"/>	 zadanie_5	-	Running (3/3)			  
<input type="checkbox"/>	 es03-1 50448dca9ff0 	docker.elastic.co/elasticsearch/	Running	9202:9200 	2 hours ago	  
<input type="checkbox"/>	 es02-1 01e6b98e51fd 	docker.elastic.co/elasticsearch/	Running	9201:9200 	2 hours ago	  
<input type="checkbox"/>	 es01-1 3f97297f9922 	docker.elastic.co/elasticsearch/	Running	9200:9200 	2 hours ago	  

Po rozbehaní som si výsledok skontroloval prostredníctvom get requestu na tieto nodes:

GET: http://localhost:9200/_cat/nodes

Výsledok:

```
1 172.18.0.2 30 100 1 0.02 0.05 0.08 cdfhilmrstw * es01
2 172.18.0.3 35 99 1 0.02 0.05 0.08 cdfhilmrstw - es02
3 172.18.0.4 68 100 0 0.02 0.05 0.08 cdfhilmrstw - es03
4
```

Uloha 2

Zadanie: Vytvorte index pre Tweets, ktorý bude mať "optimálny" počet shardov a replík pre 3 nody (aby tam bola distribúcia dotazov vo vyhľadávaní, aj distribúcia uložených dát)

Pri tejto úlohe som sa riadil predovšetkým tým, že odporúča sa na jeden shard priradiť 5-20 GB dát. Tým, že finálna databáza bude mať približne 80GB som zvolil 4 shardy. Je to aj z toho dôvodu, že na čím menej

shardov bude DB rozdelená, tým bude rýchlejšie vyhľadávanie, keďže sa nebude musieť tak často prepínať medzi nimi (preto som volil hornú hranicu).

PUT: http://localhost:9200/_cat/nodes

Vytvorenie shardov(pre každý node 4).

PUT <http://localhost:9200/tweets> Send

Params Authorization Headers (8) **Body** Pre-request Script Tests Settings Cookies

none form-data x-www-form-urlencoded raw binary GraphQL JSON Beautify

```

1 {
2   "settings": {
3     "number_of_shards": 4,
4     "number_of_replicas": 2
5   }
6 }

```

Body Cookies Headers (4) Test Results 200 OK 636 ms 202 B Save Response

Pretty Raw Preview Visualize JSON

```

1 {
2   "acknowledged": true,
3   "shards_acknowledged": true,
4   "index": "tweets"
5 }

```

GET: http://localhost:9200/_cat/shards

Skontrolovanie vytvorených shardov

GET http://localhost:9200/_cat/shards Send

Params Authorization Headers (6) **Body** Pre-request Script Tests Settings Cookies

Body Cookies Headers (4) Test Results 200 OK 22 ms 309 B Save Response

Pretty Raw Preview Visualize Text

```

1 .geoip_databases 0 p STARTED 40 38.3mb 172.18.0.3 es02
2 .geoip_databases 0 r STARTED 40 38.3mb 172.18.0.4 es03
3 tweets          0 r STARTED 0 225b 172.18.0.3 es02
4 tweets          0 p STARTED 0 225b 172.18.0.2 es01
5 tweets          0 r STARTED 0 225b 172.18.0.4 es03
6 tweets          1 p STARTED 0 225b 172.18.0.3 es02
7 tweets          1 r STARTED 0 225b 172.18.0.2 es01
8 tweets          1 r STARTED 0 225b 172.18.0.4 es03
9 tweets          2 r STARTED 0 225b 172.18.0.3 es02
10 tweets         2 r STARTED 0 225b 172.18.0.2 es01
11 tweets         2 p STARTED 0 225b 172.18.0.4 es03
12 tweets         3 r STARTED 0 225b 172.18.0.3 es02
13 tweets         3 p STARTED 0 225b 172.18.0.2 es01
14 tweets         3 r STARTED 0 225b 172.18.0.4 es03

```

Uloha 3

Zadanie: Vytvorte mapping pre normalizované dáta z Postgresu (denormalizujte ich) – Každý Tweet teda musí obsahovať údaje rovnaké ako máte už uložené v PostgreSQL (všetky tabuľky). Dbajte na to, aby ste vytvorili polia v správnom dátovom type (polia ktoré má zmysel analyzovať analyzujte správne, tie ktoré nemá, aby

neboli zbytočne analyzované (keyword analyzer)) tak aby index nebol zbytočne veľký, pozor na nested – treba ho použiť správne. Mapovanie musí byť striktné. Čo sa týka väzieb cez referencies – pre ne zaindexujte type vstáhu, id, autor (id, name, username), content a hashtags.

Bola potrebné vytvoriť mapping podľa tabuliek a ich vzťahov definovaných v prvom zadaní. Po dlhšej analýze jednotlivých tabuliek a hodinách skúšaníach sa mi podarilo vytvoriť štruktúru, ktorá dostatočne dobre reprezentuje danú schému.

Využil som 'strict' mapovanie, aby som predišiel deformácii mappingu a drzal sa striktné predpisu.

Príkaz som spúšťal cez dopyt: **PUT:** http://localhost:9200/tweets/_mapping

Response z requestu:

```
1 {  
2   "acknowledged": true  
3 }
```

Body:

```
{  
  {  
    "dynamic": "strict",  
    "properties": {  
      "id": {  
        "type": "keyword"  
      },  
      "content": {  
        "type": "text",  
        "analyzer": "englando"  
      },  
      "language": {  
        "type": "text"  
      },  
      "source": {  
        "type": "text"  
      },  
      "retweet_count": {  
        "type": "integer"  
      },  
      "reply_count": {  
        "type": "integer"  
      },  
      "like_count": {  
        "type": "integer"  
      },  
      "quote_count": {  
        "type": "integer"  
      },  
      "created_at": {  
        "type": "date"  
      }  
    }  
  }  
}
```

```
    },
    "possibly_sensitive": {
      "type": "boolean"
    },
    "author": {
      "properties": {
        "id": {
          "type": "keyword"
        },
        "name": {
          "type": "text",
          "analyzer": "standard",
          "fields": {
            "field_ngram": {
              "type": "text",
              "analyzer": "custom_ngram"
            },
            "custom_shingles": {
              "type": "text",
              "analyzer": "custom_shingles"
            }
          }
        }
      },
      "username": {
        "type": "text",
        "analyzer": "standard",
        "fields": {
          "field_ngram": {
            "type": "text",
            "analyzer": "custom_ngram"
          }
        }
      },
      "description": {
        "type": "text",
        "analyzer": "standard",
        "fields": {
          "field_englando": {
            "type": "text",
            "analyzer": "englando"
          },
          "field_shingles": {
            "type": "text",
            "analyzer": "custom_shingles"
          }
        }
      },
      "followers_count": {
        "type": "integer"
      },
      "following_count": {
        "type": "integer"
      },
      "tweet_count": {
```

```
        "type": "integer"
      },
      "listed_count": {
        "type": "integer"
      }
    }
  },
  "annotations": {
    "type": "nested",
    "properties": {
      "value": {
        "type": "text"
      },
      "type": {
        "type": "text"
      },
      "probability": {
        "type": "float"
      }
    }
  },
  "context_annotations": {
    "type": "nested",
    "properties": {
      "domain": {
        "properties": {
          "name": {
            "type": "keyword"
          },
          "description": {
            "type": "text",
            "analyzer": "englando"
          }
        }
      }
    }
  },
  "entity": {
    "properties": {
      "name": {
        "type": "keyword"
      },
      "description": {
        "type": "text",
        "analyzer": "englando"
      }
    }
  }
},
"hashtags": {
  "type": "text",
  "fields": {
    "keyword": {
      "type": "keyword",
      "normalizer": "lc_hash"
```

```

    }
  }
},
"references": {
  "type": "nested",
  "properties": {
    "type": {
      "type": "keyword"
    },
    "id": {
      "type": "keyword"
    },
    "author": {
      "properties": {
        "id": {
          "type": "keyword"
        },
        "name": {
          "type": "text",
          "analyzer": "standard",
          "fields": {
            "field_ngram": {
              "type": "text",
              "analyzer": "custom_ngram"
            },
            "field_shingles": {
              "type": "text",
              "analyzer": "custom_shingles"
            }
          }
        }
      },
      "username": {
        "type": "text",
        "analyzer": "standard",
        "fields": {
          "field_ngram": {
            "type": "text",
            "analyzer": "custom_ngram"
          }
        }
      }
    },
    "hashtags": {
      "type": "text",
      "fields": {
        "keyword": {
          "type": "keyword",
          "normalizer": "lc_hash"
        }
      }
    },
    "content": {
      "type": "text",

```

```

        "analyzer": "englando"
      }
    },
    "links": {
      "type": "nested",
      "properties": {
        "url": {
          "type": "text"
        },
        "title": {
          "type": "text",
          "analyzer": "englando"
        },
        "description": {
          "type": "text",
          "analyzer": "englando"
        }
      }
    }
  }
}
}
}

```

Uloha 4

Zadanie: Pre index tweets vytvorte 3 vlastné analyzéry (v settings) nasledovne: a. Analyzér "englando". Tento analyzér bude obsahovať nasledovné: i. filtre: english_possessive_stemmer, lowercase, english_stop, english_stemmer, ii. char_filter: html_strip iii. tokenizer: štandardný - ukážku nájdete na stránke elastic.co pre anglický analyzér b. Analyzér custom_ngram: i. filtre: lowercase, asciifolding, filter_ngrams (definujte si ho sami na rozmedzie 1- 10) ii. char_filter: html_strip iii. tokenizer: štandardný c. Analyzér custom_shingles: i. filtre: lowercase, asciifolding, filter_shingles (definujte si ho sami a dajte token_separator: "") ii. char_filter: html_strip iii. tokenizer: štandardný d. Do mapovania pridajte: i. každý anglický text (rátajme že každý tweet a description u autora je primárne v angličtine) nech je analyzovaný novým analyzérom "englando" ii. Priradíte analyzery

1. a. author.name nech má aj mapovania pre custom_ngram, a custom_shingles
2. b. author.screen_name nech má aj custom_ngram,
3. c. author.description nech má aj custom_shingles. Toto platí aj pre mentions, ak tam tie záznamy máte.

iii. Hashtagy indexujte ako lowercase

Na samom začiatku ešte predtým, ako sa dal updatovať indey a pridať do neho analyzátory bolo treba ho zavrieť. To som vykonal prostredníctvom _close. Request a výsledok je znázornený nižšie.

POST: http://localhost:9200/tweets/_close

POST ⌵ http://localhost:9200/tweets/_close Send

Params Authorization Headers (7) Body Pre-request Script Tests Settings Co

KEY	VALUE	DESCRIPTION	...	Bulk I
Key	Value	Description		

Body Cookies Headers (4) Test Results 200 OK 1425 ms 213 B Save Respon

Pretty Raw Preview Visualize JSON ⌵ ≡

```

1  {
2    "acknowledged": true,
3    "shards_acknowledged": true,
4    "indices": {
5      "tweets": {
6        "closed": true
7      }
8    }
9  }

```

Následne bolo potrebné vytvoriť analyzátory podľa zadania a index updatnúť o tieto novo vzniknuté analyzátory. Toto cele bolo potrebné urobiť ešte predtým, než sa spustilo samotné mapovanie.

PUT: http://localhost:9200/tweets/_settings

PUT ⌵ http://localhost:9200/tweets/_settings ... Send

Params Authorization Headers (8) Body ● Pre-request Script Tests Settings Cc

Body Cookies Headers (4) Test Results 200 OK 66 ms 176 B Save Respor

Pretty Raw Preview Visualize JSON ⌵ ≡

```

1  {
2    "acknowledged": true
3  }

```

Body:

```

{
  "analysis": {
    "filter": {
      "english_possessive_stemmer": {
        "type": "stemmer",
        "language": "possessive_english"
      },
      "english_stop": {
        "type": "stop",
        "stopword": "_english_"
      },
      "english_stemmer": {

```



```
        "type": "stemmer",
        "language": "english"
    },
    "filter_ngrams": {
        "type": "ngram",
        "min_gram": 2,
        "max_gram": 3
    },
    "filter_shingles": {
        "type": "shingle",
        "token_separator": ""
    }
},
"analyzer": {
    "englando": {
        "type": "custom",
        "filter": [
            "english_possessive_stemmer",
            "lowercase",
            "english_stop",
            "english_stemmer"
        ],
        "char_filter": ["html_strip"],
        "tokenizer": "standard"
    },
    "custom_ngram": {
        "type": "custom",
        "filter": [
            "lowercase",
            "asciifolding",
            "filter_ngrams"
        ],
        "char_filter": ["html_strip"],
        "tokenizer": "standard"
    },
    "custom_shingles": {
        "type": "custom",
        "filter": [
            "lowercase",
            "asciifolding",
            "filter_shingles"
        ],
        "char_filter": ["html_strip"],
        "tokenizer": "standard"
    }
},
"normalizer": {
    "lc_hash": {
        "type": "custom",
        "char_filter": [],
        "filter": [
            "lowercase"
        ]
    }
}
```

```

    }
  }
}
```

Na samom konci bolo potrebné už len indexy opäť otvoriť aby sa nad nimi dali vykonávať read/write operácie.

POST: http://localhost:9200/tweets/_open

POST http://localhost:9200/tweets/_open Send

Params Authorization Headers (7) Body Pre-request Script Tests Settings C

Query Params

KEY	VALUE	DESCRIPTION	...	Bull
Key	Value	Description		

Body Cookies Headers (4) Test Results 200 OK 458 ms 187 B Save Respo

Pretty Raw Preview Visualize JSON ↻

```

1  {
2    "acknowledged": true,
3    "shards_acknowledged": true
4  }
```

Uloha 5

Zadanie: Vytvorte bulk import pre vaše normalizované Tweety.

Najskôr bolo potrebné vytvoriť dopyt, pomocou ktorého sa budú dáta extrahovať z databázy v požadovanom formáte (definovanom v mappingu). Na základe tohoto som vyprodukoval query uvedené nižšie. Najskôr som ju realizoval prostredníctvom INNER JOINOV a potom vytvoril jeden veľký objekt, v ktorom boli dáta agregované v json objektoch. Nakoniec sa však ukázalo, že v tomto prípade je veľmi neefektívne použiť inner join a omnoho lepšie je použiť SELECT. Query pre 5000 záznamov zbehla približne za 1-2 s, zatiaľ čo INNER JOIN zaberal niečo okolo 2-3 minút.

Samozrejme bolo potrebné vytvoriť aj indexy pred tým, než som spustil túto query pre efektívne vyhľadávanie v jednotlivých tabuľkách na základe poloziek, ktoré ich spájali s Conversations (tabuľka Tweetov).

```

CREATE INDEX IF NOT EXISTS fk_index_1 ON public.conversations(author_id);
CREATE INDEX IF NOT EXISTS fk_index_2 ON
public.conversation_hashtags(conversation_id);
CREATE INDEX IF NOT EXISTS fk_index_3 ON public.conversation_hashtags(hashtag_id);
CREATE INDEX IF NOT EXISTS fk_index_4 ON
public.context_annotations(conversation_id);
CREATE INDEX IF NOT EXISTS fk_index_5 ON
public.conversation_references(conversation_id);
CREATE INDEX IF NOT EXISTS fk_index_6 ON
public.conversation_references(parent_id);
CREATE INDEX IF NOT EXISTS fk_index_7 ON public.annotations(conversation_id);
```

```
CREATE INDEX IF NOT EXISTS fk_index_8 ON public.conversations(id);
CREATE INDEX IF NOT EXISTS fk_index_9 ON public.links(conversation_id);
```

Query pre vytvorenie JSON objektov

```
select  conversations.id,
        (SELECT json_agg(jsonb_build_object(
            'id', authors.id,
            'name', authors.name,
            'username', authors.username,
            'description', regexp_replace(authors.description,
E'[\n\r\f\u000B\u0085\u2028\u2029"'+"', ' ', 'g' ),
            'followers_count', authors.followers_count,
            'following_count', authors.following_count,
            'tweet_count', authors.tweet_count,
            'listed_count', authors.listed_count
        )) FROM authors WHERE id = conversations.author_id) as author,
        regexp_replace(conversations.content,
E'[\n\r\f\u000B\u0085\u2028\u2029"'+"', ' ', 'g' ) as content,
        conversations.possibly_sensitive,
        conversations.language,
        conversations.source,
        conversations.retweet_count,
        conversations.reply_count,
        conversations.like_count,
        conversations.quote_count,
        conversations.created_at,
        (SELECT json_agg(distinct hashtags.tag) FROM hashtags WHERE hashtags.id in
        (SELECT hashtag_id FROM conversation_hashtags WHERE conversation_id =
        conversations.id)) as hashtags,
        (SELECT json_agg(distinct jsonb_build_object(
            'url', links.url,
            'title', links.title,
            'description', links.description
        )) FROM links WHERE conversation_id = conversations.id) as links,
        (SELECT json_agg(distinct jsonb_build_object(
            'value', annotations.value,
            'type', annotations.type,
            'probability', annotations.probability
        ))FROM annotations WHERE conversation_id = conversations.id) as
        annotations,
        (SELECT json_agg(distinct jsonb_build_object(
            'domain', (SELECT jsonb_build_object(
                'name', context_domains.name,
                'description', context_domains.description) FROM context_domains
WHERE id = context_annotations.context_domain_id),
            'entity', (SELECT jsonb_build_object(
                'name', context_entities.name,
                'description', context_entities.description) FROM context_entities
WHERE id = context_annotations.context_entity_id)
        )) FROM context_annotations WHERE conversation_id = conversations.id) as
        context_annotations,
```

```

        (SELECT json_agg(distinct jsonb_build_object(
            'type', conversation_references.type,
            'id', (SELECT conversations.id FROM conversations WHERE id =
conversation_references.parent_id),
            'content', regexp_replace((SELECT conversations.content FROM
conversations WHERE id = conversation_references.parent_id),
E'[\n\\r\\f\\u000B\\u0085\\u2028\\u2029"'+""]+', ' ', 'g' ),
            'author', (SELECT jsonb_build_object(
                'id', authors.id,
                'name', authors.name,
                'username', authors.username) FROM authors WHERE id = (SELECT
conversations.author_id FROM conversations WHERE id =
conversation_references.parent_id)),
            'hashtags', (SELECT json_agg(distinct hashtags.tag) FROM hashtags
WHERE hashtags.id in (SELECT hashtag_id FROM conversation_hashtags WHERE
conversation_id = (SELECT conversations.id FROM conversations WHERE id =
conversation_references.parent_id)))
        )) FROM conversation_references WHERE conversation_id = conversations.id)
as references

from conversations
group by conversations.id
limit 5000;

```

Ďalej už len bolo potrebné extrahovať tieto dáta do json súboru:

```

COPY (SELECT json_agg(row_to_json(result)))
FROM (...
'Query'
...) AS result
) TO 'path\dump_1.jsonl' WITH(HEADER FALSE);

```

Uloha 6

Zadanie: Importujete dáta do Elasticsearchu prvých 5000 tweetov

Pre túto úlohu som v pythone vytvoril krátky script, ktorý slúži na transformovanie jsonl súboru z úlohy 5 do požadovaného elastic formátu (pridal som každému riadku { "index": { "_index": "tweets" } } \n). Následne som tieto dáta importoval úspešne do Elasticu.

```

import requests
import json

js_file = open('dump.jsonl', encoding='utf8')

payload = ""

```

```
for line in js_file:
    payload += '{ "index": { "_index": "tweets" } }\n' + line

res = requests.post('http://localhost:9200/tweets/_bulk',
headers={'Content-Type': 'application/x-ndjson'},
data=payload.encode('utf8'))

print(res.text)
```

Zobrazenie výpisu do konzoly po úspešnom načítaní:

```
tweets', '_id': '01e6PIUBBMw-Q38wXOwx', '_version': 1, 'result': 'created', '_shards': {'total': 3, 'successful': 3, 'failed': 0},
  'seq_no': 1199, 'primary_term': 8, 'status': 201}}, {'index': {'_index': 'tweets', '_id': '1Fe6PIUBBMw-Q38wXOwx', '_version': 1
, 'result': 'created', '_shards': {'total': 3, 'successful': 3, 'failed': 0}}, {'seq_no': 1191, 'primary_term': 9, 'status': 201}}
, {'index': {'_index': 'tweets', '_id': '1Ve6PIUBBMw-Q38wXOwx', '_version': 1, 'result': 'created', '_shards': {'total': 3, 'succe
ssful': 3, 'failed': 0}}, {'seq_no': 1200, 'primary_term': 8, 'status': 201}}, {'index': {'_index': 'tweets', '_id': '1le6PIUBBMw-
Q38wXOwx', '_version': 1, 'result': 'created', '_shards': {'total': 3, 'successful': 3, 'failed': 0}}, {'seq_no': 1192, 'primary_t
erm': 9, 'status': 201}}, {'index': {'_index': 'tweets', '_id': '11e6PIUBBMw-Q38wXOwx', '_version': 1, 'result': 'created', '_shar
ds': {'total': 3, 'successful': 3, 'failed': 0}}, {'seq_no': 1256, 'primary_term': 9, 'status': 201}}, {'index': {'_index': 'tweet
s', '_id': '2Fe6PIUBBMw-Q38wXOwx', '_version': 1, 'result': 'created', '_shards': {'total': 3, 'successful': 3, 'failed': 0}}, 'se
q_no': 1201, 'primary_term': 8, 'status': 201}}, {'index': {'_index': 'tweets', '_id': '2Ve6PIUBBMw-Q38wXOwx', '_version': 1, 're
sult': 'created', '_shards': {'total': 3, 'successful': 3, 'failed': 0}}, {'seq_no': 1273, 'primary_term': 8, 'status': 201}}, {'i
ndex': {'_index': 'tweets', '_id': '2le6PIUBBMw-Q38wXOwx', '_version': 1, 'result': 'created', '_shards': {'total': 3, 'successful
': 3, 'failed': 0}}, {'seq_no': 1274, 'primary_term': 8, 'status': 201}}, {'index': {'_index': 'tweets', '_id': '21e6PIUBBMw-Q38wX
Owx', '_version': 1, 'result': 'created', '_shards': {'total': 3, 'successful': 3, 'failed': 0}}, {'seq_no': 1275, 'primary_term':
8, 'status': 201}}, {'index': {'_index': 'tweets', '_id': '3Fe6PIUBBMw-Q38wXOwx', '_version': 1, 'result': 'created', '_shards':
{'total': 3, 'successful': 3, 'failed': 0}}, {'seq_no': 1257, 'primary_term': 9, 'status': 201}}, {'index': {'_index': 'tweets', '_
id': '3Ve6PIUBBMw-Q38wXOwx', '_version': 1, 'result': 'created', '_shards': {'total': 3, 'successful': 3, 'failed': 0}}, {'seq_no'
index': 'tweets', '_id': '5Fe6PIUBBMw-Q38wXOwx', '_version': 1, 'result': 'created', '_shards': {'total': 3, 'successful': 3, 'fai
led': 0}}, {'seq_no': 1260, 'primary_term': 9, 'status': 201}}, {'index': {'_index': 'tweets', '_id': '5Ve6PIUBBMw-Q38wXOwx', '_ve
rsion': 1, 'result': 'created', '_shards': {'total': 3, 'successful': 3, 'failed': 0}}, {'seq_no': 1194, 'primary_term': 9, 'statu
s': 201}}, {'index': {'_index': 'tweets', '_id': '5le6PIUBBMw-Q38wXOwx', '_version': 1, 'result': 'created', '_shards': {'total':
3, 'successful': 3, 'failed': 0}}, {'seq_no': 1261, 'primary_term': 9, 'status': 201}}, {'index': {'_index': 'tweets', '_id': '51e
6PIUBBMw-Q38wXOwx', '_version': 1, 'result': 'created', '_shards': {'total': 3, 'successful': 3, 'failed': 0}}, {'seq_no': 1262, 'p
rimary_term': 9, 'status': 201}}, {'index': {'_index': 'tweets', '_id': '6Fe6PIUBBMw-Q38wXOwx', '_version': 1, 'result': 'created
', '_shards': {'total': 3, 'successful': 3, 'failed': 0}}, {'seq_no': 1263, 'primary_term': 9, 'status': 201}}, {'index': {'_index
': 'tweets', '_id': '6Ve6PIUBBMw-Q38wXOwx', '_version': 1, 'result': 'created', '_shards': {'total': 3, 'successful': 3, 'failed':
0}}, {'seq_no': 1276, 'primary_term': 8, 'status': 201}}, {'index': {'_index': 'tweets', '_id': '6le6PIUBBMw-Q38wXOwx', '_version
': 1, 'result': 'created', '_shards': {'total': 3, 'successful': 3, 'failed': 0}}, {'seq_no': 1195, 'primary_term': 9, 'status': 2
01}}, {'index': {'_index': 'tweets', '_id': '61e6PIUBBMw-Q38wXOwx', '_version': 1, 'result': 'created', '_shards': {'total': 3, 's
uccessful': 3, 'failed': 0}}, {'seq_no': 1277, 'primary_term': 8, 'status': 201}}, {'index': {'_index': 'tweets', '_id': '7Fe6PIUB
BMw-Q38wXOwx', '_version': 1, 'result': 'created', '_shards': {'total': 3, 'successful': 3, 'failed': 0}}, {'seq_no': 1278, 'prima
ry_term': 8, 'status': 201}}, {'index': {'_index': 'tweets', '_id': '7Ve6PIUBBMw-Q38wXOwx', '_version': 1, 'result': 'created', '_
shards': {'total': 3, 'successful': 3, 'failed': 0}}, {'seq_no': 1206, 'primary_term': 8, 'status': 201}}, {'index': {'_index': 't
weets', '_id': '7le6PIUBBMw-Q38wXOwx', '_version': 1, 'result': 'created', '_shards': {'total': 3, 'successful': 3, 'failed': 0}},
{'seq_no': 1196, 'primary_term': 9, 'status': 201}}, {'index': {'_index': 'tweets', '_id': '71e6PIUBBMw-Q38wXOwx', '_version': 1,
'result': 'created', '_shards': {'total': 3, 'successful': 3, 'failed': 0}}, {'seq_no': 1279, 'primary_term': 8, 'status': 201}}]
}
```

PS C:\Users\Samuel Schmidt\Desktop\Files\UNI\ING\1.rocnik_zimny\PDT\Zadanie_5> █

Na skontrolovanie, či sa do Elasticu importovali všetky dáta, som použil nasledovný request: **GET**:

http://localhost:9200/tweets/_count

GET

http://localhost:9200/tweets/_count

Send

Params

Authorization

Headers (6)

Body

Pre-request Script

Tests

Settings

Cookies

Query Params

	KEY	VALUE	DESCRIPTION	...	Bulk Edit
	Key	Value	Description		

Body

Cookies

Headers (4)

Test Results

200 OK 15 ms 220 B

Save Response

Pretty

Raw

Preview

Visualize

JSON

```
1 {
2   "count": 5000,
3   "_shards": {
4     "total": 4,
5     "successful": 4,
6     "skipped": 0,
7     "failed": 0
8   }
9 }
```

Uloha 7

Zadanie: Experimentujte s nódami, a zistite koľko nódov musí bežať (a ktoré) aby vám Elasticsearch vedel pridávať dokumenty, mazať dokumenty, prezerať dokumenty a vyhľadávať nad nimi? Dá sa nastaviť Elastic tak, aby mu stačil jeden nód? Čo je dôvodom toho že existuje nejaké kvórum?

Pre toto zadanie som si vytvoril dokument, podľa mappingu:

Body:

```
{
  "id": 4213123097427188182,
  "content": "Bonjorno",
  "possibly_sensitive": false,
  "language": "en",
  "source": "Budmerike",
  "retweet_count": 10,
  "reply_count": 100,
  "like_count": 10000,
  "quote_count": 111,
  "created_at": "2022-07-04T23:12:08.000Z",
  "author": {
    "name": "Sami",
    "username": "Sminx",
    "description": "Bc ako Bijec",
    "followers_count": 100,
    "following_count": 10000,
    "tweet_count": 100,
    "listed_count": 100
  },
  "hashtags": ["jbmnt", "VelosPodPeros"],
  "references": null,
}
```

```
"links": {
  "url": null,
  "title": null,
  "description": null
},
"annotations": {
  "value": null,
  "type": null,
  "probability": null
},
"context_annotations": null
}
```

Pridávanie dokumnetov som robil cez request: **PUT**:

http://localhost:9200/tweets/_doc/4213123097427188182

Mazanie dokumnetov som robil cez request: **DELETE**:

http://localhost:9200/tweets/_doc/4213123097427188182

Vyhľadávanie nad dokumentmi som robil cez request: **POST**: http://localhost:9200/tweets/_search

Body:

```
{
  "size": 10,
  "query": {
    "match_all": {}
  }
}
```

Prezeranie dokumnetov som robil cez request: **GET**:

http://localhost:9200/tweets/_doc/4213123097427188182

Postupne som skúšal jednotlivé kombincie zapínania a vypínania nodov a pre každú kombináciu som vyskúšal všetky 4 requesty. Moje výsledky radšej len popíšem, pretože by to zabralo veľa priestoru a pridalo na neprehľadnosti.

Zistil som že: HTTP requesty sa nedajú robiť, ak nie je zapnutý node poslúchajúci na porte :9200(v tomto prípade). Ak nie je zapnutý, žiadny z uvedených requestov nezbehne.

Vyhľadávanie a prezeranie dokumentov ide aj s jedným nodom, pokiaľ tým jedným nodom je node poslúchajúci na porte :9200. Ak tento node vypneme, žiadna iná kombinácia nefunguje.

Pridávanie a mazanie dokumentov je možné len vtedy, ak je zapnutý master node a rovnako aj node poslúchajúci na porte :9200. Najskôr som mal poslúchajúci node es01 a master node es03. Pri vypínaní a zapínaní sa však stalo to, že keď som nechal zaplnutý len es01 a es02, zmenil mastera z es03 na es01 a tým pádom zbehli všetky requesty. Ak som však ešte pred tým, než sa stal es01 master ale bol len listener, pustil tieto requesty, prebehli len requesty na vyh+adávanie a prezeranie.

Dôvod tohto kvôra je teda to, že môžeme redukovat' nody a funkcionalita sa nezmení, dokedy bude aktívny node odpočívajúci na nami zvolenom porte + bude aktívny aj master node.

Pridávam len pre ukážku 2 prípady: 1.) listener na porte :9200 bol vypnutý


GET ⌵ http://localhost:9200/tweets/_doc/4213123097427188182 Send

Params Authorization Headers (6) Body Pre-request Script Tests Settings Coc

Query Params

KEY	VALUE	DESCRIPTION	...	Bulk E
-----	-------	-------------	-----	--------

Response



Could not send request

Error: connect ECONNREFUSED 127.0.0.1:9200 | [View in Console](#)

[Learn more about troubleshooting API requests](#)

2.) listner je zapnutý, ale master je vypnutý

Nedá sa pridať dokument:

PUT ⌵ http://localhost:9200/tweets/_doc/4213123097427188182 Sen

Params Authorization Headers (8) **Body** ● Pre-request Script Tests Settings

● none ● form-data ● x-www-form-urlencoded ● raw ● binary ● GraphQL **JSON** ⌵

1 {

Body Cookies Headers (4) Test Results ⌵ 503 Service Unavailable 1 m 0.06 s 314 B Save Res

Pretty Raw Preview Visualize **JSON** ⌵ ⌵

```
1 {
2   "error": {
3     "root_cause": [
4       {
5         "type": "cluster_block_exception",
6         "reason": "blocked by: [SERVICE_UNAVAILABLE/2/no master];"
7       }
8     ],
9     "type": "cluster_block_exception",
10    "reason": "blocked by: [SERVICE_UNAVAILABLE/2/no master];"
11  },
12  "status": 503
13 }
```

⌵ Bootcamp ⌵ Runner

Dá sa však prezerať dokument:

GET ⌵ http://localhost:9200/tweets/_doc/4213123097427188182 Send

Params Authorization Headers (6) Body Pre-request Script Tests Settings Cool

Body Cookies Headers (4) Test Results 404 Not Found 40 ms 218 B Save Response

Pretty Raw Preview Visualize JSON ⌵ 🔄

```

1 {
2   "_index": "tweets",
3   "_id": "4213123097427188182",
4   "found": false
5 }
```

Master node som si našiel pomocou requestu: **GET:** http://localhost:9200/_cat/master

GET ⌵ http://localhost:9200/_cat/master Send

Params Authorization Headers (6) Body Pre-request Script Tests Settings Cool

KEY	VALUE	DESCRIPTION	...	Bulk Ec
Key	Value	Description		

Body Cookies Headers (4) Test Results 200 OK 603 ms 204 B Save Response

Pretty Raw Preview Visualize Text ⌵ 🔄

```

1 njj2VbrYRPyrFXISy20Bew 172.18.0.3 172.18.0.3 es02
2
```

Uloha 8

Zadanie: Upravujte počet retweetov pre vami vybraný tweet pomocou vášho jednoduchého scriptu (v rámci Elasticsearchu) a sledujte ako sa mení `_seq_no` a `_primary_term` pri tom ako zabíjate a spúšťate nódy.

Pre túto úlohu som si napísal krátky script na inkrementovanie retweet count-u pre `id = 983678505710911488`. Toto id malo index v elasticu = `Z1hTPYUBBMw-Q38wUUrX`.

Volal som teda tento script prostredníctvom requestu: **POST:**
http://localhost:9200/tweets/_update/Z1hTPYUBBMw-Q38wUUrX

Body:

```

{
  "script": "ctx._source.retweet_count += 1"
}
```

Keď boli zapnuté všetky nodes, inkrementovalo sa každým updatom len `seq_no`. Podľa dokumentácie by toto číslo malo predstavovať číslo zmeny nad týmto dokumentom, a je dobré ho mať najmä preto, ak by nastali

nejaké konflikty pri zmenách nad týmto dokumentom. Ide o to aby nejaka stará verzia tohto dokumentu ne prepísala novú updatnutú verziu (nová má vyššie `seq_no`).

The image displays two screenshots of a REST client interface, likely Postman, showing the response of a POST request to the endpoint `http://localhost:9200/tweets/_update/Z1hTPYUBBMw-Q38wUUrX`. The request is a POST with a JSON body.

First Screenshot:

- Method: POST
- URL: `http://localhost:9200/tweets/_update/Z1hTPYUBBMw-Q38wUUrX`
- Body (JSON):

```
{  "_index": "tweets",  "_id": "Z1hTPYUBBMw-Q38wUUrX",  "_version": 19,  "result": "updated",  "_shards": {    "total": 3,    "successful": 3,    "failed": 0  },  "_seq_no": 1256,  "_primary_term": 19}
```
- Status: 200 OK, 552 ms, 291 B

Second Screenshot:

- Method: POST
- URL: `http://localhost:9200/tweets/_update/Z1hTPYUBBMw-Q38wUUrX`
- Body (JSON):

```
{  "_index": "tweets",  "_id": "Z1hTPYUBBMw-Q38wUUrX",  "_version": 20,  "result": "updated",  "_shards": {    "total": 3,    "successful": 3,    "failed": 0  },  "_seq_no": 1257,  "_primary_term": 19}
```
- Status: 200 OK, 321 ms, 291 B

Je vidieť že sa zmenilo len `seq_no`. V prípade že však vypneme master node a tým pádom aj hlavný shard, inkrementuje sa okrem `seq_no` aj `primary_key`.

	NAME	IMAGE	STATUS	PORT(S)	STARTED	ACTIONS
	zadanie_5	-	Running (3/3)			
	es03-1 50448dca9ff0	docker.elastic.co/elasticsearch/		9202:9200	21 minutes ago	
	es02-1 01e6b98e51fd	docker.elastic.co/elasticsearch/		9201:9200	21 minutes ago	
	es01-1 3f97297f9922	docker.elastic.co/elasticsearch/	Running	9200:9200	2 minutes ago	

POST

http://localhost:9200/tweets/_update/Z1hTPYUBBMw-Q38wUUrx

Send

Params

Authorization

Headers (8)

Body

Pre-request Script

Tests

Settings

Cool

none

form-data

x-www-form-urlencoded

raw

binary

GraphQL

JSON

Beaut

1 5

Body

Cookies

Headers (4)

Test Results

200 OK 321 ms 291 B

Save Response

Pretty

Raw

Preview

Visualize

JSON

```

1  {
2    "_index": "tweets",
3    "_id": "Z1hTPYUBBMw-Q38wUUrx",
4    "_version": 20,
5    "result": "updated",
6    "_shards": {
7      "total": 3,
8      "successful": 3,
9      "failed": 0
10   },
11   "_seq_no": 1257,
12   "_primary_term": 19

```

Čo sa týka **primary_term**, ten sa menil najmä vtedy ak bol vypnutý hlavný shard a bola vykonaná zmena. Vtedy tieto requesty idú na repliku a vtedy sa zvýši **primary_term**. Keď sa hlavný shard opäť zapne, tak vidí že na replike je iný **primary_term** je vyšší ako predtým a teda sa updatne.

Uloha 9

Zadanie: Zrušte repliky a importujete všetky tweety

Ulohu 9 sa mi nepodarilo vypracovať, nakoľko nemám dostupné prostriedky na realizovanie takej veľkej migrácie dát.

Uloha 10

Zadanie: Vyhľadajte vo vašich tweetoch, kde použité **function_score** pre jednotlivé medzikroky nasledovne: a.

Must: i. Vyhľadajte vo viacerých poliach naraz (konkrétne: **author.description.shingles** (pomocou **shingle**) – **boost 10**, **content** (cez analyzovaný anglický text) spojenie – **boost 6** "put1n chr1stian fake jew", zapojte podporu pre preklepy, operátor je **OR**. ii. V **poly** **references.content** slovo "nazi" iii. Hashtag "ukraine" b. Filter: i. vyfiltrujte len tie, ktoré majú **author.following_count > 100**, tie ktoré majú **author.followers_count > 100** a tie, ktoré majú nejakú linku

c. Should: i. Ak sa v context_annotations.domain.name nachádza "Person" boostnite o 5 ii. Ak sa v context_annotations.entity.name nachádza "Soros" boostnite o 10 iii. Ak je vyhľadany string "put1n chr1stian fake jew" aj fráza s tým ze sa môže stat jedna výmena slov boostnite o 5

d. Agregácie: i. Vytvorte bucket pro-russia ktorý obsahuje hastagy používané Kremľom na propagandu: istandwithputin, racism, 1trillion, istandwithrussia, isupportrussia, blacklivesmatter, racism, racistukraine, africansinukraine, palestine, israel, freepalestine, istandwithpalestine, racisteu, putin 1. Pre neho spravte týždňový histogram, kde pre každý týždeň zobrazte štatistiky

Tento request som volal následovne: **GET:** http://localhost:9200/tweets/_search

Body:

```
{
  "query": {
    "function_score": {
      "query": {
        "bool": {
          "must" : [
            {
              "multi_match" : {
                "query" : "put1n chr1stian fake jew",
                "fields": ["content^6",
"author.description.custom_shingles^10"],
                "operator": "or",
                "fuzziness": "AUTO"
              }
            }
          ]
        },
        // {
        //   "match" : {
        //     "referencies.content": "nazi"
        //   }
        // },
        // {
        //   "match" : {
        //     "hashtags": "ukraine"
        //   }
        // }
      ],
      "filter": [
        { "range": { "author.following_count": { "gte": 100 } } },
        { "range": { "author.followers_count": { "gte": 100 } } },
        {"bool": {"must_not":{"match" : {"links": {"query": "null"}}}}}
      ],
      "should" : [
        {"match": {
          "context_annotations.domain.name" : {
            "query" : "Person",
            "boost" : 5
          }
        }
      ]
    }
  }
}
```

21 / 23

Response:

The screenshot shows a REST client interface with a GET request to `http://localhost:9200/tweets/_search`. The response is a JSON object with the following structure:

```
1 {
2   "took": 116,
3   "timed_out": false,
4   "_shards": {
5     "total": 4,
6     "successful": 4,
7     "skipped": 0,
8     "failed": 0
9   },
10  "hits": {
11    "total": {
12      "value": 1161,
13      "relation": "eq"
14    },
15    "max_score": 91.009964,
16    "hits": [
```

The screenshot continues the JSON response from the previous one, showing a list of document hits:

```
12897     "key_as_string": "2019-01-07T00:00:00.000Z",
12898     "key": 1546819200000,
12899     "doc_count": 12,
12900     "tags": {
12901       "doc_count": 7,
12902       "statistics": {
12903         "doc_count_error_upper_bound": 0,
12904         "sum_other_doc_count": 8,
12905         "buckets": [
12906           {
12907             "key": "putin",
12908             "doc_count": 7
12909           },
12910           {
12911             "key": "kgb",
12912             "doc_count": 2
```

Na začiatku som začal s podmienkou Must. Tu som zapojil custom_shingles z tvorby analyzátorov a multi_match. Fuzziness = "Auto" zabezpečuje rozumnú toleranciu preklepov. Nakoľko sa mi nepodarilo importovať celú databázu kvoli pamäťovej náročnosti, pracoval som iba s 5000 záznamami, čo znamená že zvyšné 2 podmienky pre MUST sú zakomentované z toho dôvodu, aby request niečo vracal. Obe podmienky sú však funkčné (stačí len odkomentovať).

Ďalej som vytváral Filtre. Tu nebol problém urobiť filtre pre following_count a followers_count. Problém, ktorý sa mi žiaľ nepodarilo vyriešiť bolo pre links. Podmienka must_not totiž vracia aj také výsledky, kde links=null.

Podmienka exists nefungovalo a nedopátral som sa k nicomu lepsiemu.

Should pre Sorosa a Person je len jednoduchý match, každý len boostnutý o inú hodnotu. Multi_match však aj so zámenou slov sa mi nepodarilo implementovať.

Na samom konci to boli agregácie, kde som posutpoval tak ža, na samom začiatku sa vytvorí histogram po jednotlivých týždňoch. Na kazdom zázname teda prvý doc_count predstavuje počet dokumentov v danom týždni. V parametri **tags** je uvedený druhý doc_count, ktorý hovorí o tom, koľko požadovaných hashtagov sa nachádza v dokumentoch z daného týždňa. V parametri **statistics** je aj presne vidieť, ktoré požadované hashtagy sa v danom týždni vyskytovali a aj koľkokrát.