

PDT - Elastic Search

https://github.com/MennoCoehoorn/PDT_5.git

Marek Štrba

November/December 2021

Obsah

1	Rozbehajte si 3 inštancie Elasticsearch-u	4
2	Vytvorte index pre Tweety, ktorý bude mať "optimálny" počet shardov a replík pre 3 nody (aby tam bola distribúcia dotazov vo vyhľadávaní, aj distribúcia uložených dát)	4
3	Vytvorte mapping pre normalizované dáta z Postgresu - Tweet musí obsahovať údaje rovnaké ako máte už uložené v PostgreSQL. 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. Mapovanie musí byť striktné.	5
4	Pre index tweets vytvorte 3 vlastné analyzéry (v settings) nasledovne:	6
4.1	Analýzér "englando". Tento analyzér bude obsahovať nasledovné:	6
4.2	Analýzér custom_ngram:	6
4.3	Analýzér custom_shingles:	7
4.4	Do mapovania pridajte:	8
4.4.1	každý anglický text (rátajme že každý tweet a description u autora je primárne v angličtine) nech je analyzovaný novým analyzérom "englando"	8
4.4.2	Priradte analyzery	8
4.4.3	Hashtagy indexujte ako lowercase	8
5	Vytvorte bulk import pre vaše normalizované Tweety a importujete dáta do Elasticsearchu prvých 5000 tweetov	8
6	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?	10
6.1	Všetko zapnute	11
6.2	Vypnutý Node-3	11
6.3	Vypnutý Node-2 a Node-3	12
6.4	Zapnutý iba Node-2	12
6.5	Vypnutý iba Node-1	12
7	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.	13
7.1	Všetko zapnute	13
7.2	Vypnutý Node-3	13
7.3	Vypnutý Node-2 a Node-3 a následne znovu zapnuté	14

- 8 Zrušte repliky a importujete všetky tweety 14
- 9 Vyhľadajte vo vašich tweetoch spojenie "gates s0ros vaccine micr0chip".
V query použite function_score 16
- 10 Konšpiračné teórie podľa Elasticu. Pracujte zo všetkými tweetami,
ktoré máte. Následne pre všetky týždne zistite pomocou vnorených
agregácií, koľko retweet.count sumárne majú tweety ktoré majú hash-
tagy z prvého zadania. Teda na základe hashtagov znova rozdeľte
tweety do konšpiračných teórií ale pomocou agregácií. 19

1 Rozbehajte si 3 inštancie Elasticsearch-u

Lokálne som si rozbehal 3 inštancie Elasticu (z troch rozbalených archívov). V .yml súboroch som musel nastaviť spoločný cluster.name, nastaviť rôzne node.name, node-1 som nastavil ako master a všetky som nastavil ako data. Dôležité bolo nastaviť vo všetkých .yml cluster.initial_master_nodes: ["node-1"].

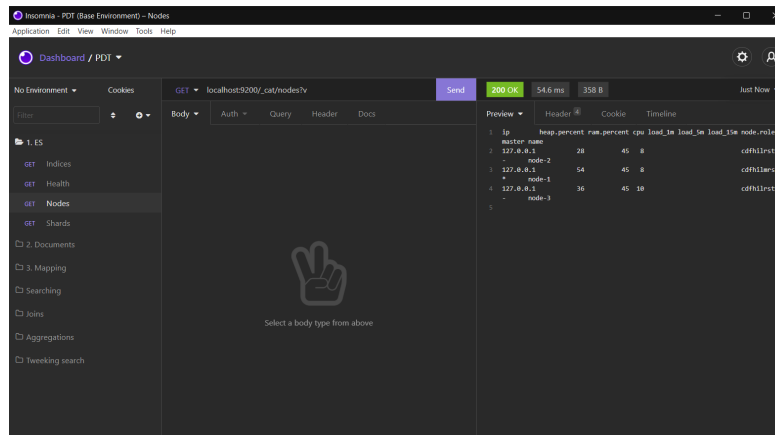


Figure 1: Ukážka stavu nodov na spustenom clusteru

2 Vytvorte index pre Tweety, ktorý bude mať “optimálny” počet shardov a replík pre 3 nody (aby tam bola distribúcia dotazov vo vyhľadávaní, aj distribúcia uložených dát)

Keďže som všetky 3 nodes nastavil ako data, tak som si mohol dovoliť dať 3 shardy. Viac ako 1 shard na node by už spomalovalo index. Každý shard má 2 replíky na ostatných nodoch, tak aby sa zabezpečila prístupnosť a backup. Každý node teda obsahuje jeden shard a replíky zvyšných 2 shardov.

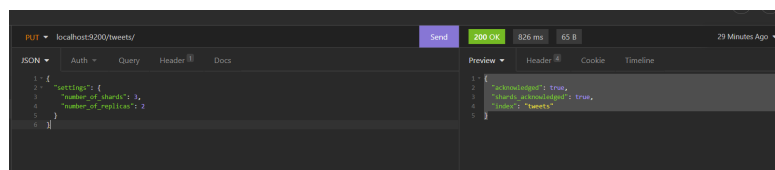


Figure 2: Tvorba indexu

Preview ▼	Header 4	Cookie	Timeline
1	.geoip_databases	0 p	STARTED 42 40.9mb 127.0.0.1 node-1
2	.geoip_databases	0 r	STARTED 42 40.9mb 127.0.0.1 node-2
3	tweets	1 p	STARTED 0 208b 127.0.0.1 node-1
4	tweets	1 r	STARTED 0 208b 127.0.0.1 node-3
5	tweets	1 r	STARTED 0 208b 127.0.0.1 node-2
6	tweets	2 r	STARTED 0 208b 127.0.0.1 node-1
7	tweets	2 r	STARTED 0 208b 127.0.0.1 node-3
8	tweets	2 p	STARTED 0 208b 127.0.0.1 node-2
9	tweets	0 r	STARTED 0 208b 127.0.0.1 node-1
10	tweets	0 p	STARTED 0 208b 127.0.0.1 node-3
11	tweets	0 r	STARTED 0 208b 127.0.0.1 node-2
12			

Figure 3: Aktívne shardy a repliky po vytvorení indexu

- 3 Vytvorte mapping pre normalizované dáta z Postgresu - Tweet musí obsahovať údaje rovnaké ako máte už uložené v PostgreSQL. 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. Mapovanie musí byť striktné.

Preview ▼	Header 4	Cookie	Timeline
1	{		
2	"acknowledged": true		
3	}		

Figure 4: Tvorba mapovania pre index

Údaje o authorovi som dal do nested typu author. Nested som vybral preto, lebo sa budú jednotlivé atribúty authora neskôr používať v podmienkach. Ako keyword som dal country, hashtags, pg_id (id v postgrese) a parent_id pretože tie budú porovnávané ako celé stringy alebo vôbec. Ako text som dal content, author.screen_name, author.name a author.description, keďže nad nimi sa bude hľadať slovné spojenie "gates s0ros vaccine micr0chip". Ostatné atribúty boli číselné a dal som ich buď ako integer alebo ako double, podľa potreby. Mentions som do datasetu nepridal, keďže z neskorších zadaní vyplynulo, že nie sú potrebné.

4 Pre index tweets vytvorte 3 vlastné analyzéry (v settings) nasledovne:

4.1 Analyzátor "englando". Tento analyzátor bude obsahovať nasledovné:

- filtre: english_possessive_stemmer, lowercase, english_stop, english_stemmer
- char_filter: html_strip
- tokenizer: štandardný

Navedel som nastaviť stemmer na possessive english bez toho aby som ho customisol s language nastaveným na possessive_english. English_stop a english_stemmer som dal iba ako stop a stemmer, keďže tie sú by default english.

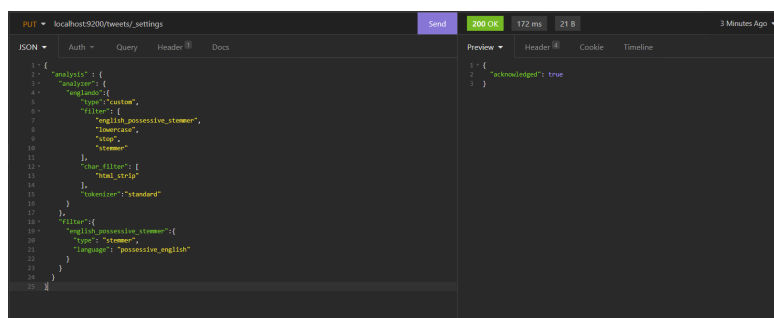


Figure 5: Tvorba englando analyzéra

4.2 Analyzátor custom_ngram:

- Filtre: lowercase, asciifolding, filter_ngrams (definujte si ho sami na rozmedzie 1-10)
- char_filter: html_strip

- tokenizer: štandardný

Kvôli zmene pri ngram filtri na 1-10 bolo treba zmeniť aj "max_ngram_diff" z default 1 na 10.

```

1: {
2:   "tokenizer": {
3:     "max_ngram_diff": 10
4:   },
5:   "analyzer": {
6:     "custom_ngram": {
7:       "type": "custom",
8:       "filter": [
9:         "lowercase",
10:        "asciifolding",
11:        "1_10_ngram"
12:      ],
13:       "char_filter": [
14:         "html_strip"
15:       ],
16:       "tokenizer": "standard"
17:     }
18:   },
19:   "filter": {
20:     "1_10_ngram": {
21:       "type": "ngram",
22:       "min_ngram": 1,
23:       "max_ngram": 10
24:     }
25:   }
26: }

```

```

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

```

Figure 6: Tvorba custom_ngram analyzéra

4.3 Analyzér custom_shingles:

- Filtre: Filtre: lowercase, asciifolding, filter_shingles (definujte si ho sami a dajte token_separator: "")
- char.filter: html_strip
- tokenizer: štandardný

Bolo treba samostatne zadefinovať tokenizer pre shingles filter.

```

1: {
2:   "tokenizer": {
3:     "analyzer": {
4:       "custom_shingles": {
5:         "type": "custom",
6:         "filter": [
7:           "lowercase",
8:           "asciifolding",
9:           "filter_shingles"
10:        ],
11:        "char_filter": [
12:          "html_strip"
13:        ],
14:        "tokenizer": "standard"
15:      }
16:    },
17:    "filter": {
18:      "filter_shingles": {
19:        "type": "shingle",
20:        "min_shingle_size": 2,
21:        "max_shingle_size": 10,
22:        "token_separator": "",
23:        "token_separator": ""
24:      }
25:    }
26:  }
27: }

```

```

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

```

Figure 7: Tvorba custom_shingles analyzéra

4.4 Do mapovania pridajte:

4.4.1 každý anglický text (rátajme že každý tweet a description u autora je primárne v angličtine) nech je analyzovaný novým analyzárom "eng-lando"

4.4.2 Priradte analyzery

- author.name nech má aj mapovania pre custom_ngram, a custom_shingles,
- author.screen_name nech má aj custom_ngram,
- author.description nech má aj custom_shingles. Toto platí aj pre mentions, ak tam tie záznamy máte.

4.4.3 Hashtagy indexujte ako lowercase

Keďže sa v Elasticu nedá zadať viac ako jeden analyzátor pre field v mapingu musel som použiť multimapping pomocou fields parametra. Pri hashtagoch som ich musel pretypovať na text vo fields, aby sa na nich mohol použiť analyzátor. Indexovanie hashtagov v lower case som zaručil normalizerom "lowercase".

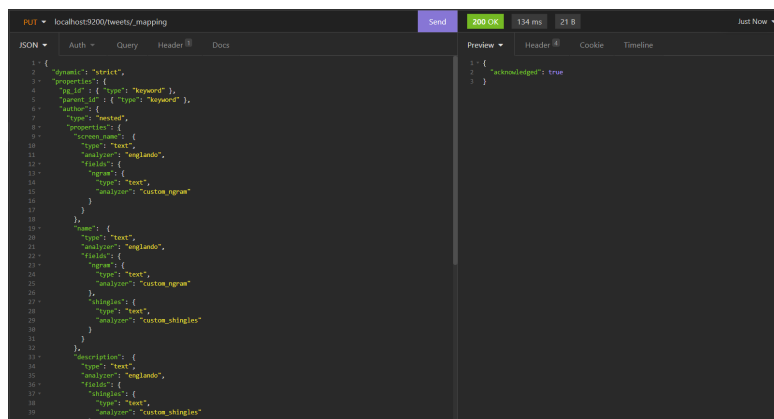


Figure 8: Ukážka z tvorby mapovania

5 Vytvorte bulk import pre vaše normalizované Tweety a importujete dáta do Elasticsearchu prvych 5000 tweetov

Na vytvorenie json súboru pre bulk import som si vytvoril jednoduchý python script:


```

b_file = open("D:/FIIT/ING/1_Semester/PDT/PDT_5/5k.json", encoding='utf-8')

create_string = '{ "create" : {} }\n'
string_to_write = ''
counter = 0
with open('D:/FIIT/ING/1_Semester/PDT/PDT_5/5k.json', encoding='utf-8') as file:
    for line in file:
        counter+=1
        string_to_write = string_to_write + create_string + a_file.readline()
        if counter % 1000 == 1:
            with open('bulk_insert_1.json', "a", encoding='utf-8') as file_object:
                file_object.write(string_to_write)
                print(counter)
                string_to_write = ''

```

Figure 9: Script, ktorý pridáva create volania do jsonu, ktorý vytvorí postgresql, tak aby sa pomocou bulk api dali dáta jednoducho importovať do elasticu

Pri tvorbe json exportu z postgresql som na vytvorenie author objektu využil `jsonb_build_object`. Musel som pomocou regexu odstrániť new line znaky a úvodzovky, keďže to predstavovalo problém pri importe do Elasticu.

```

COPY(SELECT row_to_json(results)
FROM (
    SELECT t.id as pg_id, t.parent_id, jsonb_build_object(
        'screen_name', a.screen_name,
        'name', a.name,
        'description', regexp_replace(a.description, E'[\n\r\f\\u000B\\u0085\\u2028\\u2029"]+', ' ', 'g' ),
        'followers_count', a.followers_count,
        'friends_count', a.friends_count,
        'statuses_count', a.statuses_count
    ) as author, regexp_replace(t.content, E'[\n\r\f\\u000B\\u0085\\u2028\\u2029"]+', ' ', 'g' ) as content,
    t.retweet_count, t.favorite_count, t.happened_at,
    c.name as country, t.neg,
    t.neu, t.pos, t.compound,
    array_agg(h.value) as hashtags
    FROM tweets t
    INNER JOIN tweet_hashtags th ON t.id = th.tweet_id
    INNER JOIN hashtags h ON th.hashtag_id = h.id
    INNER JOIN countries c ON t.country_id = c.id
    INNER JOIN accounts a ON t.author_id = a.id
    GROUP BY t.id, a.screen_name, a.name, a.description, a.followers_count, a.friends_count,
    a.statuses_count, c.name LIMIT 5000
) results) TO 'D:/FIIT/ING/1_Semester/PDT/PDT_5/5k.json' WITH (FORMAT text, HEADER FALSE);

```

Figure 10: Export z postgresql databázy

Pri importe som sa stretol s errorom pri použití Insomnie, ktorý som nevedel vyriešiť, tak som na toto volanie použil Postmana.

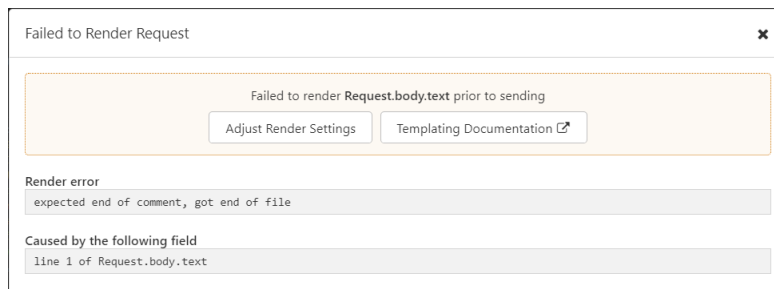


Figure 11: Chyba v Insomnii

Import zlyhal na 2 vstupoch, ale to mi prišlo ako zanedbateľné množstvo.

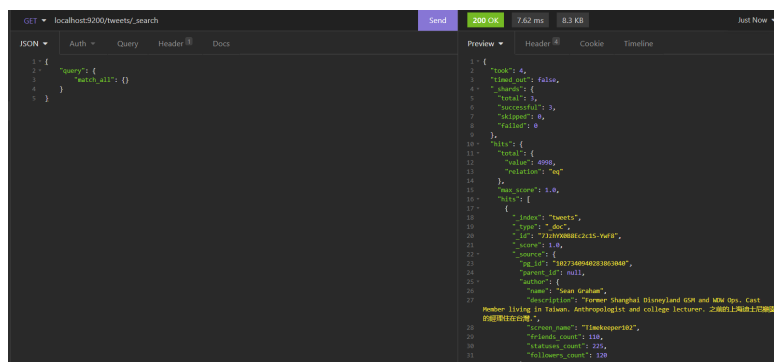


Figure 12: Výpis match-all query nad dokumentami, v hits môžeme vidieť, že sa úspešne importovalo 4998 dokumentov

6 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?

Vytvoril som si dopyty na konkrétny document podľa id, jeho update, pridanie a delete. Tiež som si vytvoril query na vyhľadanie tweetov od účtov s viac ako 100 000 followermi.

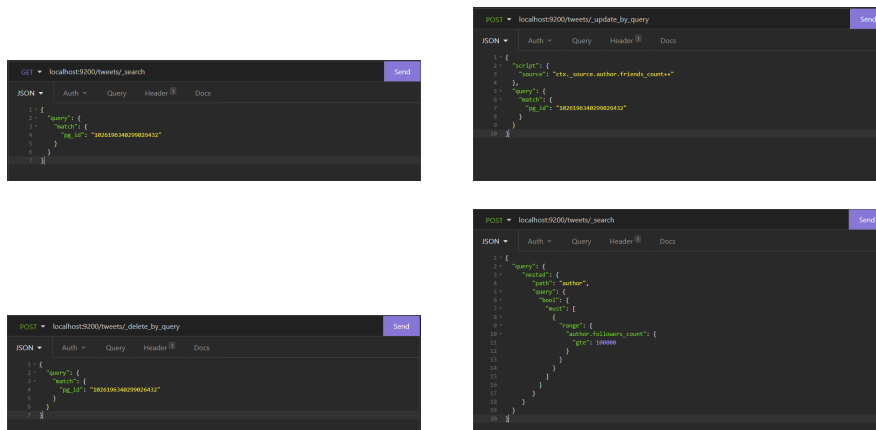


Figure 13: Dopyty použité v rámci tejto úlohy a-get one, b-update, c-delete, d-search. Insert som robil cez bulk api.

6.1 Všetko zapnute

Health - green

- get - "took": 5, "_shards": {"total": 3, "successful": 3, "skipped": 0, "failed": 0}
- update - "took": 69
- delete - "took": 34
- add - "took": 33, "_seq_no": 6795, "_primary_term": 5, "status": 201
- search - "took": 4, "_shards": {"total": 3, "successful": 3, "skipped": 0, "failed": 0}

6.2 Vypnutý Node-3

Health - yellow

- get - "took": 499, "_shards": {"total": 3, "successful": 3, "skipped": 0, "failed": 0}
- update - "took": 75
- delete - "took": 23
- add - "took": 70, "_seq_no": 6888, "_primary_term": 4, "status": 201
- search - "took": 6, "_shards": {"total": 3, "successful": 3, "skipped": 0, "failed": 0}

6.3 Vypnutý Node-2 a Node-3

Health - yellow

- get - "took": 919, "_shards": {"total": 3, "successful": 3, "skipped": 0, "failed": 0}
- update - "took": 52
- delete - "took": 14
- add - "took": 13, "_seq_no": 6891, "_primary_term": 4, "status": 201
- search - "took": 4, "_shards": {"total": 3, "successful": 3, "skipped": 0, "failed": 0}

6.4 Zapnutý iba Node-2

Health sa nedal zistiť, query faulovala - Error: Failure when receiving data from the peer

- get - 503 - service not available - cluster_block_exception
- update - 503 - service not available - cluster_block_exception
- delete - 503 - service not available - cluster_block_exception
- add - query timeoutla
- search - 503 - service not available - cluster_block_exception

6.5 Vypnutý iba Node-1

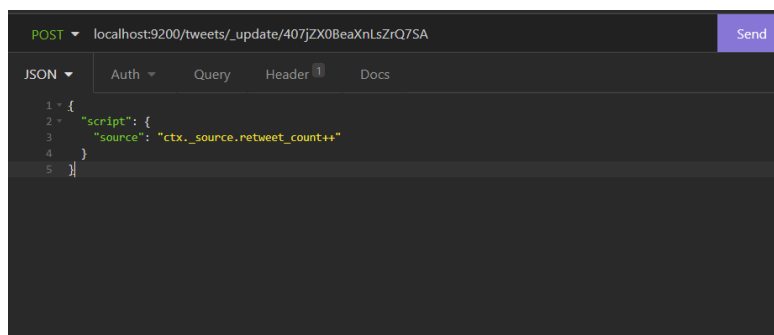
Health failol - 503 - master_not_discovered_exception

- get - 503 - service not available - cluster_block_exception
- update - 503 - service not available - cluster_block_exception
- delete - 503 - service not available - cluster_block_exception
- add - add - query timeoutla
- search - 503 - service not available - cluster_block_exception

Ak by sme chceli aby elastic fungoval tak aby stačilo mať zapnutý hociktorý jeden node, tak by museli byť všetky nody ako master-eligible. Ja mám ako master node nastavený iba node-1 preto mi môj cluster funguje iba keď je spustený node-1 bez ohľadu na to aké iné nody sú spustené. Mení sa iba rýchlosť vykonania dopytov. Ak chceme mať len jeden node na elasticu, treba pri tvorbe indexu vytvoriť iba 1 shard a nastaviť discovery.type: single-node v .yml súbore.

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

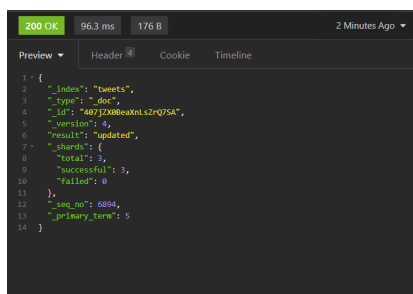
Na testovanie som použil nasledovný dopyt:



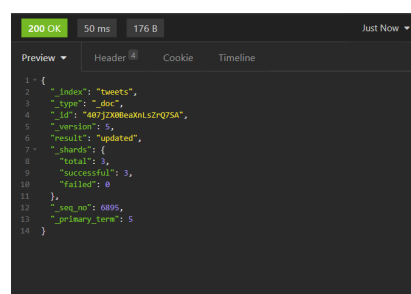
```
POST localhost:9200/tweets/_update/407jZX0BeaXnLsZrQ7SA
{
  "script": {
    "source": "ctx._source.retweet_count++"
  }
}
```

V každom settingu, ktoré som si určil som dopyt spustil 2-krát.

7.1 Všetko zapnute



```
200 OK 96.3 ms 176 B
{
  "_index": "tweets",
  "_type": "_doc",
  "_id": "407jZX0BeaXnLsZrQ7SA",
  "_version": 4,
  "result": "updated",
  "_shards": {
    "total": 3,
    "successful": 3,
    "failed": 0
  },
  "_seq_no": 6894,
  "_primary_term": 5
}
```



```
200 OK 50 ms 176 B
{
  "_index": "tweets",
  "_type": "_doc",
  "_id": "407jZX0BeaXnLsZrQ7SA",
  "_version": 5,
  "result": "updated",
  "_shards": {
    "total": 3,
    "successful": 3,
    "failed": 0
  },
  "_seq_no": 6895,
  "_primary_term": 5
}
```

7.2 Vypnutý Node-3

Po vypnutí node-3 sa nezmenil primary term, čo znamená, že dokument bol uložený na primary sharde, ktorý nie je na node-3.

```
200 OK 368 ms 176 B Just Now
Preview Header Cookie Timeline
1 {
2   "_index": "tweets",
3   "_type": "_doc",
4   "_id": "407jZ0MeaknIszrQ75A",
5   "_version": 6,
6   "result": "updated",
7   "_shards": {
8     "total": 3,
9     "successful": 2,
10    "failed": 0
11  },
12  "_seq_no": 6896,
13  "_primary_term": 5
14 }
```

```
200 OK 29.8 ms 176 B Just Now
Preview Header Cookie Timeline
1 {
2   "_index": "tweets",
3   "_type": "_doc",
4   "_id": "407jZ0MeaknIszrQ75A",
5   "_version": 7,
6   "result": "updated",
7   "_shards": {
8     "total": 3,
9     "successful": 2,
10    "failed": 0
11  },
12  "_seq_no": 6897,
13  "_primary_term": 5
14 }
```

7.3 Vypnutý Node-2 a Node-3 a následne znovu zapnuté

Opakovane som vypínal a zapínal Node-2 a Node-3 a raz som vypol aj celý cluster. V rámci týchto procesov sa musel dokument zapísať na iný shard a preto sa primary term zmenil z 5 na 10, čo indikuje 5 zmien.

```
Preview Header Cookie Timeline
1 {
2   "_index": "tweets",
3   "_type": "_doc",
4   "_id": "407jZ0MeaknIszrQ75A",
5   "_version": 8,
6   "result": "updated",
7   "_shards": {
8     "total": 3,
9     "successful": 3,
10    "failed": 0
11  },
12  "_seq_no": 6898,
13  "_primary_term": 10
14 }
```

```
Preview Header Cookie Timeline
1 {
2   "_index": "tweets",
3   "_type": "_doc",
4   "_id": "407jZ0MeaknIszrQ75A",
5   "_version": 8,
6   "result": "updated",
7   "_shards": {
8     "total": 3,
9     "successful": 3,
10    "failed": 0
11  },
12  "_seq_no": 6898,
13  "_primary_term": 10
14 }
```

Sequence number sa menil každým dopytom, keďže ráta akcie vykonané nad indexom. Ak by sme v dopyte dokument iba getovali, tak by sa sequence number nezvýšil, no tým, že sme zasiahli do jeho údajov updatom, tak sa zvýšil po každom dopyte.

8 Zrušte repliky a importujete všetky tweety

Pri tvorbe exportu z pg som použil nasledovnú query:

```

COPY(SELECT row_to_json(results)
FROM (
  SELECT t.id as pg_id, t.parent_id, jsonb_build_object(
    'screen_name', a.screen_name,
    'name', a.name,
    'description', regexp_replace(a.description, E'[\n\r\f\u000B\u0085\u0202\u2029]+' , ' ', 'g' ),
    'followers_count', a.followers_count,
    'friends_count', a.friends_count,
    'statuses_count', a.statuses_count
  ) as author, regexp_replace(t.content, E'[\n\r\f\u000B\u0085\u0202\u2029]+' , ' ', 'g' ), t.tweet_count,
  c.name as country, t.neg,
  t.new, t.pos, t.compound,
  array_agg(h.value) as hashtags
  FROM tweets t
  INNER JOIN tweet_hashtags th ON t.id = th.tweet_id
  INNER JOIN hashtags h ON th.hashtag_id = h.id
  INNER JOIN countries c ON t.country_id = c.id
  INNER JOIN accounts a ON t.author_id = a.id
  GROUP BY t.id, a.screen_name, a.name, a.description, a.followers_count, a.friends_count,
  a.statuses_count, c.name LIMIT 500000
) results) TO 'D:\FIIT\ING\1_Semester\PDT\PDT_5\500k2.json' WITH (FORMAT text, HEADER FALSE);

```

Po jej použití som získal ale iba niečo cez 120 000 záznamov. Keď som ju zbehol s offsetom:

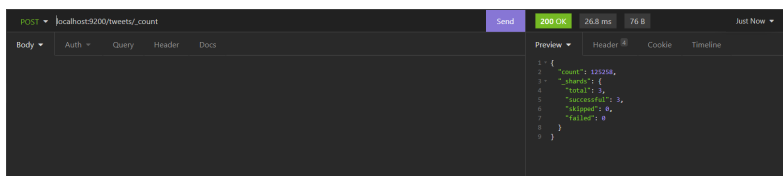
```

5      'name', a.name,
6      'description', regexp_replace(a.description, E'[\n\r\f\u000B\u0085\u0202\u2029]+' , ' ', 'g' ),
7      'followers_count', a.followers_count,
8      'friends_count', a.friends_count,
9      'statuses_count', a.statuses_count
10     ) as author, regexp_replace(t.content, E'[\n\r\f\u000B\u0085\u0202\u2029]+' , ' ', 'g' ), t.tweet_count,
11     c.name as country, t.neg,
12     t.new, t.pos, t.compound,
13     array_agg(h.value) as hashtags
14   FROM tweets t
15   INNER JOIN tweet_hashtags th ON t.id = th.tweet_id
16   INNER JOIN hashtags h ON th.hashtag_id = h.id
17   INNER JOIN countries c ON t.country_id = c.id
18   INNER JOIN accounts a ON t.author_id = a.id
19   GROUP BY t.id, a.screen_name, a.name, a.description, a.followers_count, a.friends_count,
20   a.statuses_count, c.name LIMIT 500000 OFFSET 130000
21 ) results) TO 'D:\FIIT\ING\1_Semester\PDT\PDT_5\500k2.json' WITH (FORMAT text, HEADER FALSE);

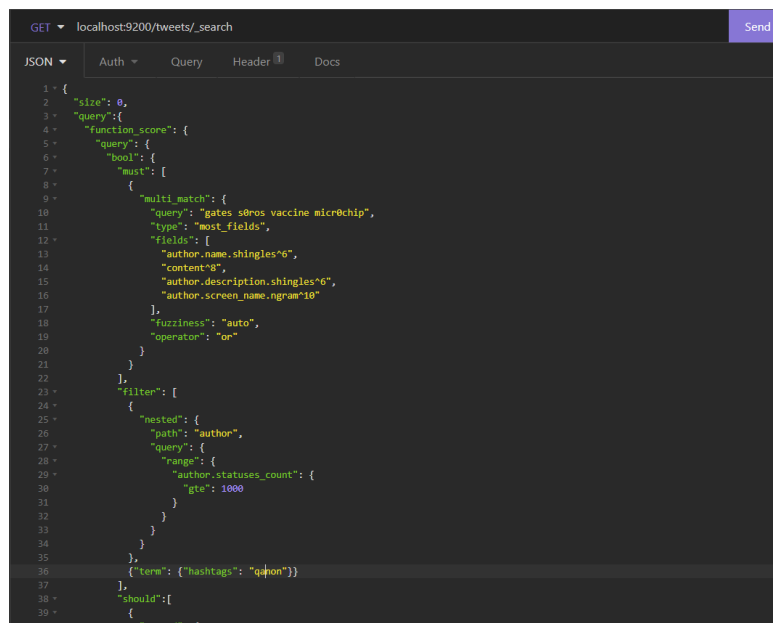
```

Query returned successfully in 2 min 18 secs.

Tak mi to nevrátilo žiadne záznamy. Túto chybu sa mi nepodarilo opraviť, preto som nakoniec importoval iba 125 258 tweetov. Keďže som mal už predtým naindexovaných pôvodných 5000 dokumentov, ktoré bolo tiež treba vymazať, tak som prestavenie nodov riešil zmazaním indexu a znovuvytvorením bez replík. 125 258 tweetov som manuálne cez postmana nainportoval po batchoch po 5000.



9 Vyhľadajte vo vašich tweetoch spojenie "gates s0ros vaccine micr0chip". V query použite function_score



```
1 {
2   "size": 0,
3   "query": {
4     "function_score": {
5       "query": {
6         "bool": {
7           "must": [
8             {
9               "multi_match": {
10                "query": "gates s0ros vaccine micr0chip",
11                "type": "most_fields",
12                "fields": [
13                  "author.name.shingles*6",
14                  "content*8",
15                  "author.description.shingles*6",
16                  "author.screen_name.ngram*10"
17                ],
18                "fuzziness": "auto",
19                "operator": "or"
20              }
21            ]
22          },
23          "filter": [
24            {
25              "nested": {
26                "path": "author",
27                "query": {
28                  "range": {
29                    "author.statuses_count": {
30                      "gte": 1000
31                    }
32                  }
33                }
34              }
35            ]
36          },
37          "term": {"hashtags": "q4p0n"}
38        },
39        "should": [
40          {
41            "nested": {
```

Figure 14: Size:0 som dal lebo ma nezaujímali výsledné dokumenty ale len určená agregácia nad hashtagmi. Celé query-bool som musel obaliť do function_score. Keďže bolo treba zabezpečiť vyhľadávanie vo viacerých fieldoch naraz tak som použil funkciu multi_match. Do jej argumentov som do fields pridaval všetky potrebné polia a cez . notáciu som vybral (keďže išlo o multi-field polia), ktorý field má byť pre dané pole použitý podľa potrebného analyzátora. Type som nechal na most_fields, keďže mi prišlo ako vhodné zoradiť výsledky podľa početnosti výskytu daných fráz v rôznych poliach. Boostovanie jednotlivých polí som zabezpečil pomocou notácie. Pri filtrovaní som musel použiť nested query pri dopyte na status count, keďže je to atribút vnorený v objekte. Dopyt na hashtag bol priamočiary term.


```

    "should": [
      {
        "nested": {
          "path": "author",
          "query": {
            "match": {"author.name.ngram": {"query": "real", "boost": 12}}
          }
        }
      }
    ],
    "functions": [
      {
        "filter": {"range": {"retweet_count": {"gte": 100, "lte": 500}}},
        "weight": 6
      },
      {
        "filter": {
          "nested": {
            "path": "author",
            "query": {
              "range": {"author.followers_count": {"gt": 100}}
            }
          }
        },
        "weight": 3
      }
    ]
  },
}

```

Figure 15: Should musel byť podobne ako filter spravený pomocou nested query, keďže sa znova jednalo o atribút objektu autor. Boost som riešil ako parameter. Funkcie boli priamočiare, pomocou filter a range som určil príslušným dokumentom následne atribútom weight žiadajú hodnotu.

```

"aggs": {
  "hashtags": {
    "terms": {
      "field": "hashtags"
    }
  }
}

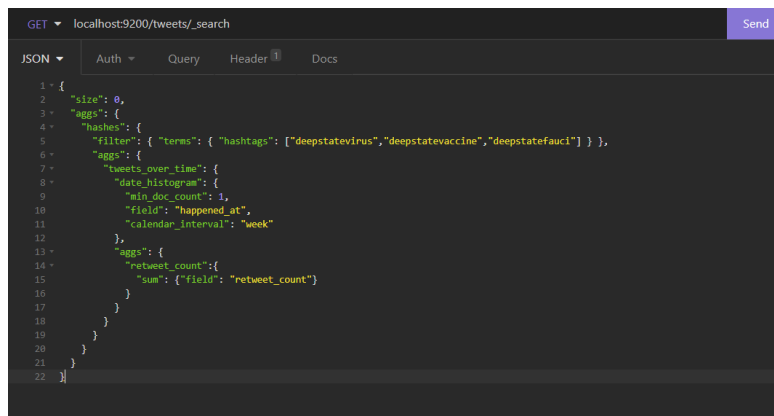
```

Figure 16: Agregácia bola priamočiara, stačilo použiť type terms nad polom hashtags. Počet výskytu jednotlivých hashtagov sa následne pre ich bucket vrátil ako doc_count

```
200 OK 554 ms 444 B A Minute Ago ▼
Preview ▼ Header 4 Cookie Timeline
1 {
2   "took": 550,
3   "timed_out": false,
4   "_shards": {
5     "total": 3,
6     "successful": 3,
7     "skipped": 0,
8     "failed": 0
9   },
10  "hits": {
11    "total": {
12      "value": 2,
13      "relation": "eq"
14    },
15    "max_score": null,
16    "hits": []
17  },
18  "aggregations": {
19    "hashtags": {
20      "doc_count_error_upper_bound": 0,
21      "sum_other_doc_count": 0,
22      "buckets": [
23        {
24          "key": "coronavirusoutbreak",
25          "doc_count": 2
26        },
27        {
28          "key": "coronovirus",
29          "doc_count": 2
30        },
31        {
32          "key": "eugenics",
33          "doc_count": 2
34        },
35        {
36          "key": "impeachmenttrial",
37          "doc_count": 2
38        },
39        {
40          "key": "narmv".
```

Figure 17: Keďže sa mi nepodarilo importovať všetky tweety, tento dopyt mi pôvodne nič nevracal, ale keď som pridal do must "virus" a hashtags zmenil z qanon na qarmy tak mi to vrátilo správne buckety hashtagov.

- 10 Konšpiračné teórie podľa Elasticu. Pracujte zo všetkými tweetami, ktoré máte. Následne pre všetky týždne zistite pomocou vnorených agregácií, koľko retweet_count sumárne majú tweety ktoré majú hashtagy z prvého zadania. Teda na základe hashtagov znova rozdeľte tweety do konšpiračných teórií ale pomocou agregácií.



```
1 {
2   "size": 0,
3   "aggs": {
4     "hashtags": {
5       "filter": { "terms": { "hashtags": ["deepstatevirus", "deepstatevaccine", "deepstatefauci"] } },
6       "aggs": {
7         "tweets_over_time": {
8           "date_histogram": {
9             "min_doc_count": 1,
10            "field": "happened_at",
11            "calendar_interval": "week"
12          },
13          "aggs": {
14            "retweet_count": {
15              "sum": { "field": "retweet_count" }
16            }
17          }
18        }
19      }
20    }
21  }
22 }
```

Figure 18: Na požadovaný výsledok som potreboval vytvoriť tri vnorené agregácie. Vonkajšia je typu filter a pomocou terms vyfiltruje iba dokumenty obsahujúce dané hashtagy. Následne sa dokumenty pomocou vnorenej agregácie typu date.histogram podľa atribútu happened_at zatriedia do bucketov po týždni ("calendar_interval" = week). Dátumová agregácia má v sebe vnorenú poslednú agregáciu, ktorá je typu sum a pod názvom retweet.count vráti zrátaný kumulovaný počet retweetov pre daný bucket.

```
200 OK 14.2 ms 1676 B Just Now
Preview Header 4 Cookie Timeline
--
17 },
18 "aggregations": {
19   "hashes": {
20     "doc_count": 140,
21     "tweets_over_time": {
22       "buckets": [
23         {
24           "key_as_string": "2020-01-20T00:00:00.000Z",
25           "key": 1579478400000,
26           "doc_count": 8,
27           "retweet_count": {
28             "value": 0.0
29           }
30         },
31         {
32           "key_as_string": "2020-01-27T00:00:00.000Z",
33           "key": 1580083200000,
34           "doc_count": 24,
35           "retweet_count": {
36             "value": 20.0
37           }
38         },
39         {
40           "key_as_string": "2020-02-03T00:00:00.000Z",
41           "key": 1580688000000,
42           "doc_count": 6,
43           "retweet_count": {
44             "value": 1.0
45           }
46         },
47         {
48           "key_as_string": "2020-02-10T00:00:00.000Z",
49           "key": 1581292800000,
50           "doc_count": 3,
51           "retweet_count": {
52             "value": 0.0
53           }
54         },
55         {
56           "key_as_string": "2020-02-17T00:00:00.000Z",
57           "key": 1581897600000,
58           "doc_count": 3,
59           "retweet_count": {
60             "value": 0.0
61           }
62         }
63       ]
64     }
65   }
66 }
```

Figure 19: Keďže sa mi nepodarilo importovať všetky tweety, tento dopyt nie vždy niečo vrátil a keď vrátil, tak určite nie všetky výsledky. Jeho funkčnosť, je však napríklad dobre viditeľná na teórii Qanon, kde vrátil celkovo 140 dokumentov a pekne vytvoril požadované týždňové buckety.

Výsledky všetkých teórií sú uložené ako samostatné jsony na githube, sú v priečinku 11 a nazvané sú podľa danej teórie.