**5. zadanie zamerané na Elastic:**

Odovzdanie do 12.12. 23:59. Zadania 1-8 je dokopy za 7,5 boda tak isto ako aj 9 – 11. Dokopy teda za 15 bodov. Odovzdávate dokument s popisom a queries separátne každú ako json čo posielate a json čo dostanete ako odpoveď.

github: https://github.com/jaruji/elastic_search_PDT

**1.** Rozbehajte si 3 inštancie Elasticsearch-u
**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)
**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.
**4.** Pre index tweets vytvorte 3 vlastné analyzéry (v settings) nasledovne:
- **1.** Analyzér "englando". Tento analyzér bude obsahovať nasledovné:
  - **1.2.** fitre: english_possessive_stemmer, lowercase, english_stop, english_stemmer,
  - **1.3.** char_filter: html_strip
  - **1.4.** tokenizer: Štandardný
  - ukážku nájdete na stránke elastic.co pre anglický analyzér
- **2**. Analyzér custom_ngram:
  - **2.2**. Filtre: lowercase, asciifolding, filter_ngrams (definujte si ho sami na rozmedzie 1-10)
  - **2.3**. char_filter: html_strip
  - **2.4**. tokenizer: Štandardný
- **3**. Analyzér custom_shingles:
  - **3.2**. Filtre: lowercase, asciifolding, filter_shingles (definujte si ho sami a dajte token_separator: "")
  - **3.3**. char_filter: html_strip
  - **3.4**. tokenizer: Štandardný

Do mapovania pridajte:
- **1**. každý anglický text (rátajme že každý tweet a description u autora je primárne v angličtine) nech je analyzovaný novým analyzérom "englando"
- **2**. Priraďte analýzery
  - a. author.name nech má aj mapovania pre custom_ngram, a custom_shingles,
  - b. author. screen_name nech má aj custom_ngram,
  - c. author.description nech má aj custom_shingles. Toto platí aj pre mentions, ak tam tie záznamy máte.
- **3**. Hashtagy indexujte ako lowercase

**5**. Vytvorte bulk import pre vaše normalizované Tweety.

**6**. Importujete dáta do Elasticsearchu prvych 5000 tweetov
**7**. 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?
**8**. Upravujte počet retweetov pre vami vybraný tweet pomocou vašeho jednoduchého scriptu (v rámci Elasticsearchu) a sledujte ako sa mení _seq_no a _primary_term pri tom ako zabíjate a spúšťate nódy.
**9**. Zrušte repliky a importujete všetky tweety
**10**. Vyhľadajte vo vašich tweetoch spojenie "gates s0ros vaccine micr0chip". V query použite function_score, kde jednotlivé medzikroky sú nasledovné:

Query:

**1**. Must - vyhľadajte vo viacerých poliach (konkrétne: author.name (pomocou shingle), content (cez analyzovaný anglický text), author.description (pomocou shingles), author.screen_name (pomocou ngram)) spojenie "gates s0ros vaccine micr0chip", zapojte podporu pre preklepy, operátor je OR.

**2.1** tieto polia vo vyhľadávaní boost-nite nasledovne - author.name * 6, content * 8, author.description * 6, author.screen_name * 10.

**3.** Filter - vyfiltrujte len tie, ktoré majú author.statuses_count > 1000 a tie, ktoré majú hashtag „qanon“

**4.** Should – boost-nite 10 krat tie, ktoré obsahujú v mentions.name (tento objekt je typu nested) cez ngram string "real".

**5.** Nastavte podmienené váhy cez functions nasledovne:

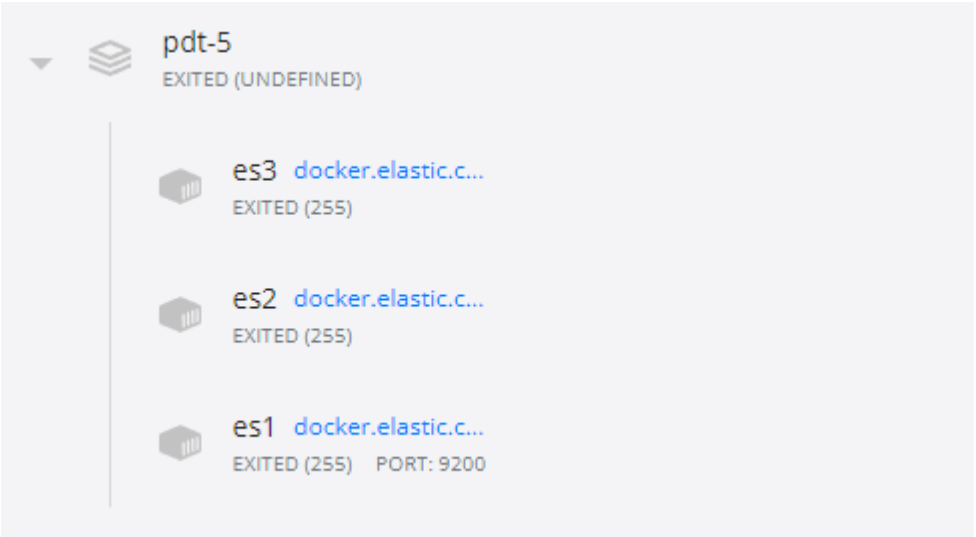
**5.1.** retweet_count, ktorý je väčší rovný ako 100 a menší rovný ako 500 na 6, **5.2.** author.followers_count väčší ako 100 na 3

Zobrazte agregácie pre výsledky na konci. Vytvorte bucket hashtags podľa hashtagov a spočítajte hodnoty výskytov (na webe by to mohli byť facety).

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

**Riešenie:**

1. Cez docker compose.




2. Shardy som zvolil 3, kvôli tomu že optimálny počet shardov má byť rovný počtu nodov (ak máme 1 index). Keďže repliky zrýchľujú vyhľadávanie no sú náročné na miesto na disku, zvolil som hodnotu 2.
3. Dáta som neťahal z postgresu ale z mongoDB, keďže som doňho tweety prenášal pri minulom zadaní. Výhodou monga je, že všetky relevantné dáta tam mám, rovnako aj hashtagy ktoré sú groupnuté v rámci dokumentu. Jedinou úpravou bolo pridanie author objektu do tweet dokumentov ako ich atribút. Výhodou monga je aj to, že dáta mi prídu ako json a môžem ich rovno namapovať do elasticu bez nejakých úprav. V úlohe neplánujem pracovať co všetkými tweetami ale iba s konšpiračnými (cca 700k, keďže mám na disku už extrémne málo miesta). V mapovaní som taktiež nezohľadnoval mentiony, keďže tie v dátach v mongoDB nemám spracované.

4. Tvar jsonu ktorým som vytváral index je možno vidieť na gite. Všetko som robil podľa inštrukcií, jedine som nezahrnul location z postgresu lebo som ho v mongoDB nemal načítané (iba null, z nejakého dôvodu sa pri prenose nepreniesli korektne tie dáta z pg do monga). Taktiež som neriešil mentiony.

5. Importol som prvých 5000 cez funkciu bulk z knižnice elasticsearch v pythone, pričom som potreboval ošetrovať aj NaN hodnoty v country_id (z nejakého dôvodu atribút null_value v mappings pre atribút country_id nebol dostačujúci a musel som ich nastavovať na nejakú hodnotu takýmto hackom).

```
bigbulk = []

for index, row in enumerate(df.to_dict(orient='records')):
    if index == 5000:
        break
    row.pop('_id')
#     row['author'].pop('_id')
    doc = row
    doc['_index'] = 'normalized'
    if(str(doc['country_id']).lower() == "NaN".lower()):
        doc['country_id'] = -99
    bigbulk.append(doc)
```

**6.** Importol som prvých 5000 dokumentov získaných z df v pythone získaných z monga.

```
In [67]: bulk(es, bigbulk)

Out[67]: (5000, [])
```

```
{
  "took": 1760,
  "timed_out": false,
  "_shards": {
    "total": 3,
    "successful": 3,
    "skipped": 0,
    "failed": 0
  },
  "hits": {
    "total": {
      "value": 5000,
      "relation": "eq"
    },
    "max_score": 1,
    "hits": [
      {
        "_index": "normalized",
        "_type": "_doc",
        "_id": "iKO8r30BIgFuB6tls0P0",
        "_score": 1,
        "_source": {
          "id": "1261437291278397444",
          "content": "Evolution of a #Qanon \r\n\r\nHandle with care ;) https://t.co/yjrpWED0uB",
          "location": null,
          "retweet_count": 16,
          "favorite_count": 0,
          "compound": 0.6249,
          "neu": 0.541,
          "neg": 0,
          "pos": 0.459,
          "happened_at": "2020-05-15T21:24:21",
          "author_id": "790231392810524673",
          "country_id": -99,
          "parent_id": null,
          "hashtags": [
            "MAGA",
            "KAG"
          ],
          "author": {
            "screen_name": "grammyk3boys1gi",
            "name": "GrammyKwokeup & @AwakenedbyGod #FreeAssange/Flynn",
            "description": "",
            "id": "790231392810524673",
            "follower_count": 2290,
            "friends_count": 2862,
            "statuses_count": 13708
          }
        }
```


7. Dá sa nastaviť aby využíval 1 node, no o to náročnejšie bude vykonávanie menovaných operácií.
8. Použil som es.update funkciu ktorej bolo potrebné zadať atribút _id, ktorý som vybral náhodný z výsledkov searchu v indexe.

```
def modifyRetweetCount(id):
    es.indices.refresh(index='normalized')
    es.update(
        index='normalized', id = id,
        body={"doc": { "retweet_count": 80085 }}
    )
```

```
modifyRetweetCount("jqO8r30BIgFuB6tls0P0")
```

```
es.search(index='normalized', query = {
    "match": {
        "id": "1246183844815888385"
    }
})
```

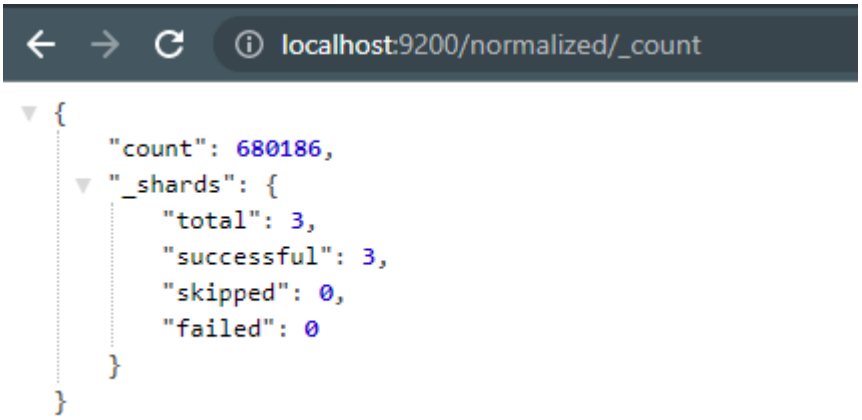

```
{'took': 7,
 'timed_out': False,
 '_shards': {'total': 3, 'successful': 3, 'skipped': 0, 'failed': 0},
 'hits': {'total': {'value': 1, 'relation': 'eq'},
  'max_score': 6.4916334,
  'hits': [{'_index': 'normalized',
    '_type': '_doc',
    '_id': 'jqO8r30BIgFuB6tls0P0',
    '_score': 6.4916334,
    '_source': {'id': '1246183844815888385',
     'content': 'When everything is all said and done, and "The Plan" is ac
nd direct the greatest story ever told! I think he should name it "Operatic
\n\r\n#QAnon #WWG1WGA https://t.co/xwNXjQ9s69',
     'location': None,
     'retweet_count': 80085,
     'favorite_count': 0,
```


9. Importovať všetky dáta som musel po bulkoch (aj keď pracujem len so subsetom z dôvodu, že mam plný disk). Vkladal som po 20k nech mi nezhorí pc. Repliky som zmazal pri inicializácii indexu `"number_of_replicas":0,` z dôvodu ušetrenia miesta.

```
In [*]: bigbulk = []

for index, row in enumerate(df.to_dict(orient='records')):
    row['author'].pop('_id', None)
    row.pop('_id', None)
    doc = row
    doc['_index'] = 'normalized'
    if(str(doc['country_id']).lower() == "NaN".lower()):
        doc['country_id'] = -99
    bigbulk.append(doc)
    if(index % 20000 == 0):
        bulk(es, bigbulk)
        bigbulk = []

bulk(es, bigbulk)
```




Do indexu mi načítalo všetkých 680k konšpiračných tweetov, s ktorými som pracoval aj v zadaniach 1 a 3.

**10.**

Query vyzeralo nasledovne:

```
es.search(index ="normalized", query = {
    "function_score": {
        "query": {
            "bool": {
                "must": [
                    {
                        "bool": {
                            "should": [
                                {
                                    "match": {
                                        "content": {
                                            "query": "gates s0ros vaccine micr0chip",
                                            "fuzziness": "AUTO",
                                            "analyzer": "englando",
                                            "boost": 8.0
                                        }
                                    }
                                },
                                {
                                    "match": {
                                        "author.name": {
                                            "query": "gates s0ros vaccine micr0chip",
                                            "fuzziness": "AUTO",
                                            "analyzer": "custom_shingles",
                                            "boost": 6.0
                                        }
                                    }
                                },
                                {
                                    "match": {
                                        "author.screen_name": {
                                            "query": "gates s0ros vaccine micr0chip",
                                            "fuzziness": "AUTO",
                                            "analyzer": "custom_ngram",
                                            "boost": 10.0
                                        }
                                    }
                                },
                                {
                                    "match": {
                                        "author.description": {
                                            "query": "gates s0ros vaccine micr0chip",
                                            "fuzziness": "AUTO",
                                            "analyzer": "custom_shingles",
                                            "boost": 6.0
                                        }
                                    }
                                }
                            ]
                        }
                    }
                ],
```

```
                "filter": [
                    {
                        "nested": {
                            "path": "author",
                            "query": {
                                "range": {
                                    "author.statuses_count": {
                                        "gt": 1000
                                    }
                                }
                            }
                        }
                    },
                    {
                        "term": {
                            "hashtags": "qanon"
                        }
                    }
                ]
            }
        },
        "functions": [
            {
                "filter": {
                    "range": {
                        "retweet_count": {
                            "gte": 100,
                            "lte": 500
                        }
                    }
                },
                "weight": 6
            },
            {
                "filter": {
                    "range": {
                        "author.followers_count": {
                            "gt": 100
                        }
                    }
                },
                "weight": 3
            }
        ]
    }
}, aggs = {
         "hashtags":{
        "terms":{
            "field":"hashtags",
            "size": 10000
        }
    }
})
```

Čítal som, že atribúte size v agregačnej funkcii (počet bucketov) je potrebné nastaviť na

nejaké vysoké číslo lebo používanie 0 na dynamickú veľkosť je deprecated.


```
'buckets': [{'key': 'qanon', 'doc_count': 45557},
 {'key': 'wwg1wga', 'doc_count': 12764},
 {'key': 'maga', 'doc_count': 5744},
 {'key': 'trump', 'doc_count': 3148},
 {'key': 'q', 'doc_count': 2900},
 {'key': 'thegreatawakening', 'doc_count': 2604},
 {'key': 'qarmy', 'doc_count': 2328},
 {'key': 'epstein', 'doc_count': 2288},
 {'key': 'kag', 'doc_count': 2282},
 {'key': 'coronavirus', 'doc_count': 1991},
 {'key': 'greatawakening', 'doc_count': 1805},
 {'key': 'twgrp', 'doc_count': 1734},
 {'key': 'wwg1wgaworldwide', 'doc_count': 1713},
 {'key': 'qanon2020', 'doc_count': 1598},
 {'key': 'tuesdaymorning', 'doc_count': 1578},
 {'key': 'qanon2018', 'doc_count': 1478},
```


ukážka prvých n bucketov (pracoval som so subsetom dát, len s konšpiračnými tweetami). Dáta je možné si pozrieť v notebooku na predom linknutom gite.

**11.** Keďže som pracoval iba so subsetom z prvého zadania, tak som mal hashtagy už vyfiltrované a nebolo ich potrebné filtrovať opätovne. Následne som počítal sumu retween count konšpiračných tweetov po týždňoch.

```
es.search(index ="normalized", query = {
#     "query": {"hashtags":  tags}
    "match_all": {}
},
aggs = {
    "weeks": {
      "date_histogram": {
        "field": "happened_at",
        "calendar_interval": "week"
      },
        "aggs": {
            "sum_of_retweets": {
              "sum": {
                "field": "retweet_count"
              }
          }
        }
    }
 })
```

```
  sum_of_retweets : { value : 66014409.0}},
{'key_as_string': '2020-05-04T00:00:00',
 'key': 1588550400000,
 'doc_count': 63146,
 'sum_of_retweets': {'value': 68017187.0}},
{'key_as_string': '2020-05-11T00:00:00',
 'key': 1589155200000,
 'doc_count': 95577,
 'sum_of_retweets': {'value': 61671588.0}},
{'key_as_string': '2020-05-18T00:00:00',
 'key': 1589760000000,
 'doc_count': 39349,
 'sum_of_retweets': {'value': 21643309.0}},
{'key_as_string': '2020-05-25T00:00:00',
 'key': 1590364800000,
 'doc_count': 80830,
 'sum_of_retweets': {'value': 181107863.0}}]}}}
```

Ukážka výsledkov, mali by byť v jupyter notebooku na gite.