Оптимизация хранения данных в OpenSearch

GALS SOFTWARE

Антон Касимов

Поговорим о модели хранения данных и методах оптимизации хранения в OpenSearch

Что вас ждет:

- Как хранятся данные
- Как оптимизировать хранение

GALS SOFTWARE

- СПЕЦИАЛИЗИРОВАННЫЙ СИСТЕМНЫЙ ИНТЕГРАТОР (СИСТЕМЫ МОНИТОРИНГА)
- ВНЕДРЕНИЕ И НАСТРОЙКА OPENSEARCH
- ТЕХНИЧЕСКАЯ ПОДДЕРЖКА OPENSEARCH
- ОБУЧЕНИЕ НА ТРЕНИНГАХ «БАЗА» И «ПРОДВИНУТЫЙ»
- ПАРТНЕР Yandex Cloud

elastic

@ELASTICSTACK_RU

@MONITORIM_IT

Подписывайтесь на каналы

Тренинги в 2026 году:

- **OpenSearch База** → 18 - 20 февраля
- **OpenSearch Продвинутый** → 18 - 20 марта

* Полное расписание: https://gals.software/education/timetable

OpenSearch База (3 дня)

- Теория + практика (40/60)
- От основ к глубокому погружению
- Установка отказоустойчивого кластера
- Загрузка и трансформация данных через Logstash, DataPrepper, Fluent Bit, Vector
- Визуализации, ISM, снапшоты, Benchmark

Программа курса

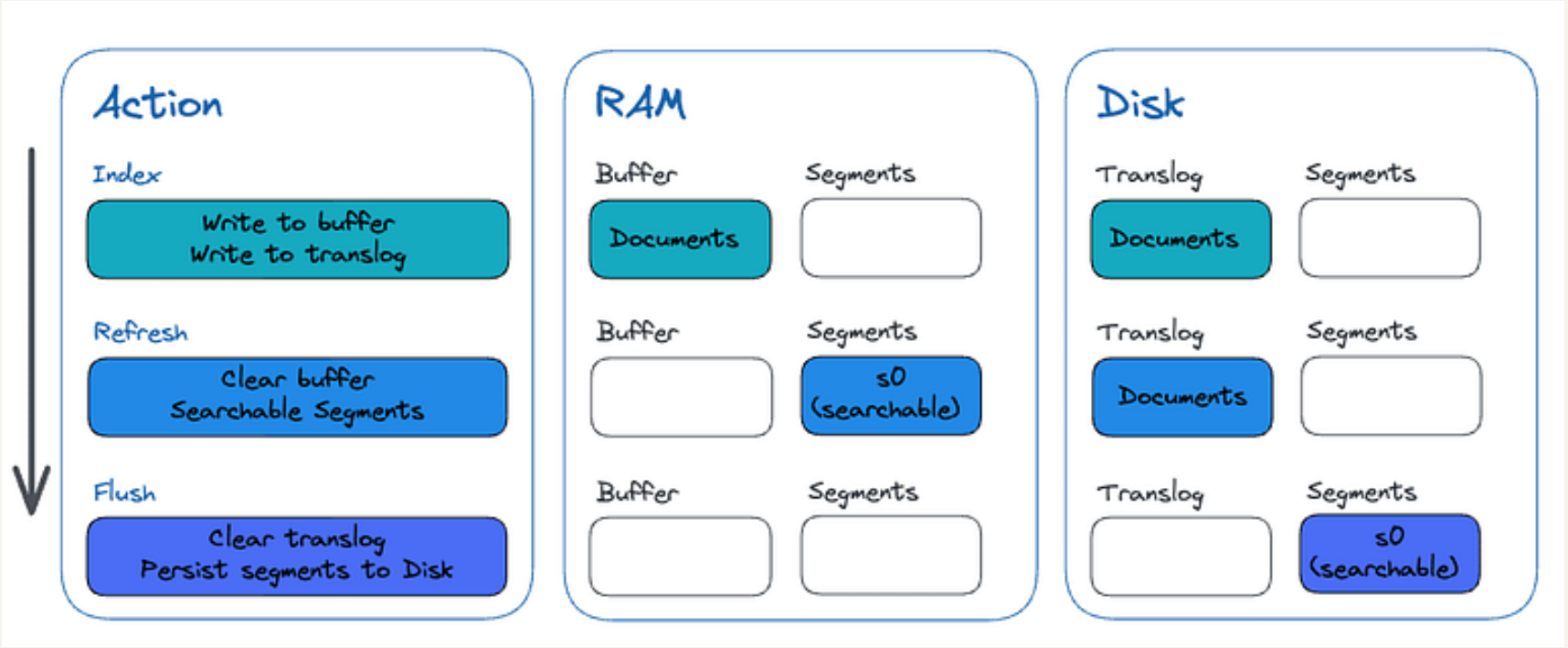
OpenSearch Продвинутый (3 дня)

- Теория + практика (40/60)
- Глубокая работа с OpenSearch
- Ролевая модель и аудит-лог, маппинг и observability
- Кросс-кластерная репликация и поиск, продвинутый ISM, оповещения
- Работа с Vector, Logstash, DataPrepper, OpenTelemetry Collector, Ingest Pipeline и Kafka

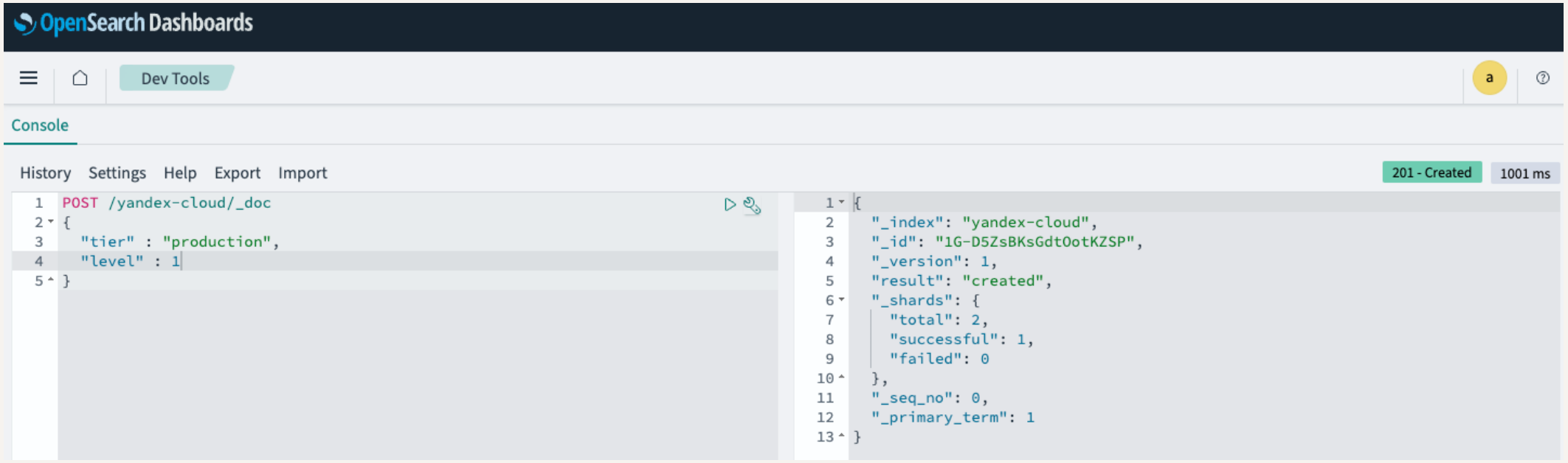
Программа курса

КАК ХРАНЯТСЯ ДАННЫЕ

Как хранятся данные в OpenSearch

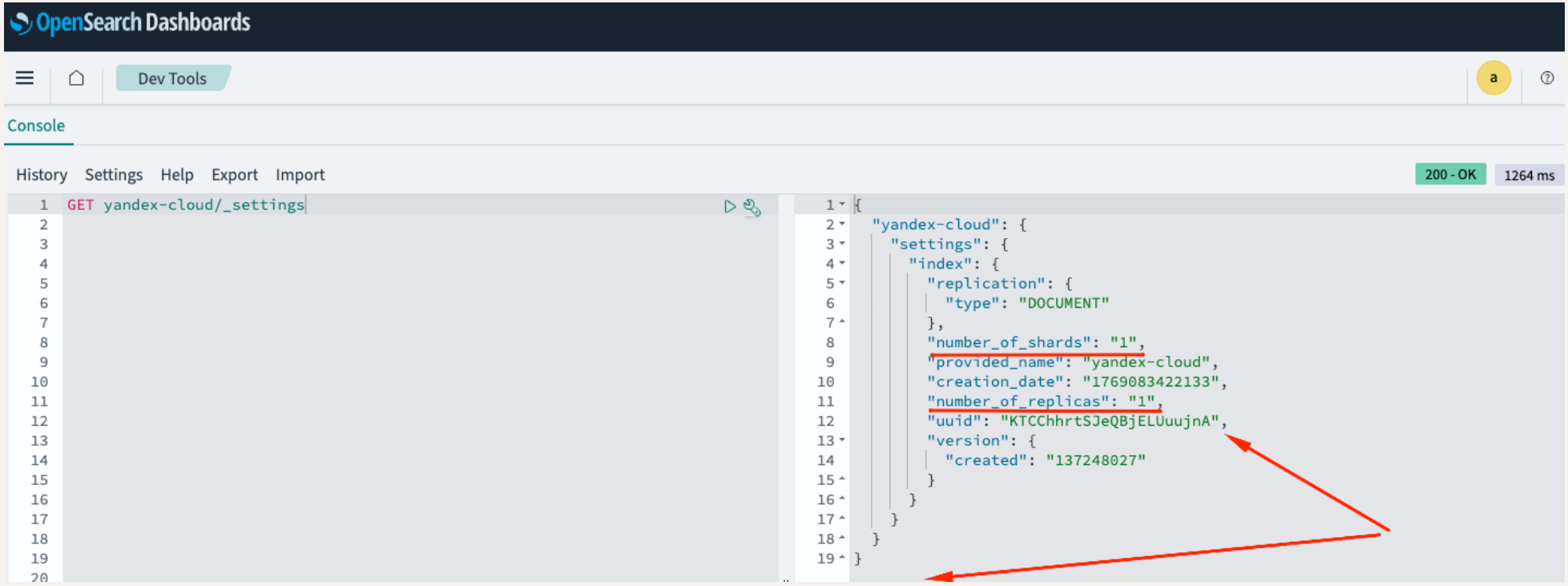


Запишем документ в индекс



Почему total = 2, successful = 1?

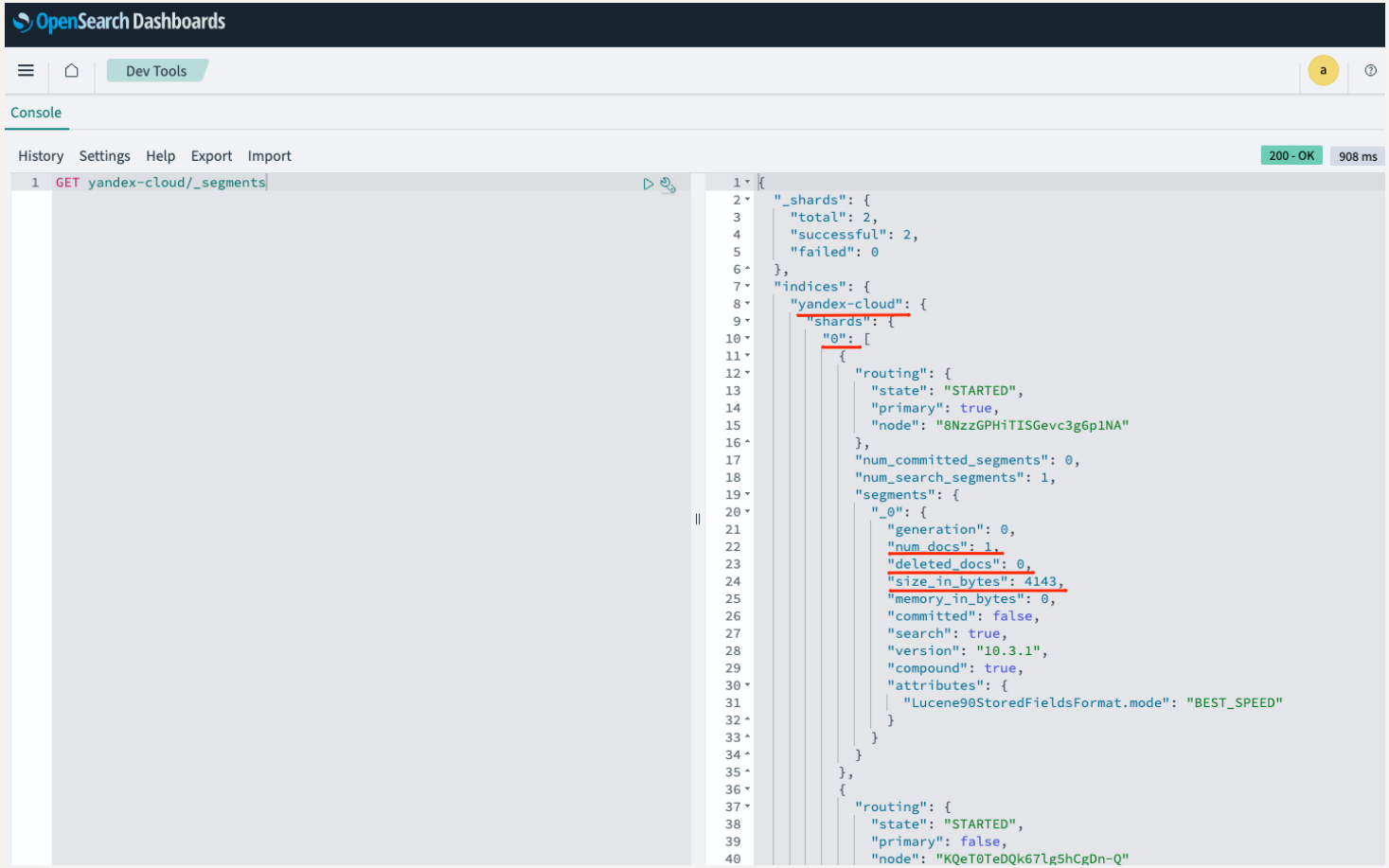
Посмотрим на шарды и сегменты




```
root@vm-opensearch01:~# ls -la /var/lib/opensearch/nodes/0/indices/KTCChhrtSJeQBjELUuujnA/0/index/
total 24
drwxr-xr-x 2 opensearch opensearch 4096 Jan 22 15:03 .
drwxr-xr-x 5 opensearch opensearch 4096 Jan 22 15:03 ..
-rw-r--r-- 1 opensearch opensearch  517 Jan 22 15:03 _0.cfe
-rw-r--r-- 1 opensearch opensearch 3297 Jan 22 15:03 _0.cfs
-rw-r--r-- 1 opensearch opensearch  329 Jan 22 15:03 _0.si
-rw-r--r-- 1 opensearch opensearch  325 Jan 22 15:03 segments_4
-rw-r--r-- 1 opensearch opensearch    0 Jan 22 15:03 write.lock
```

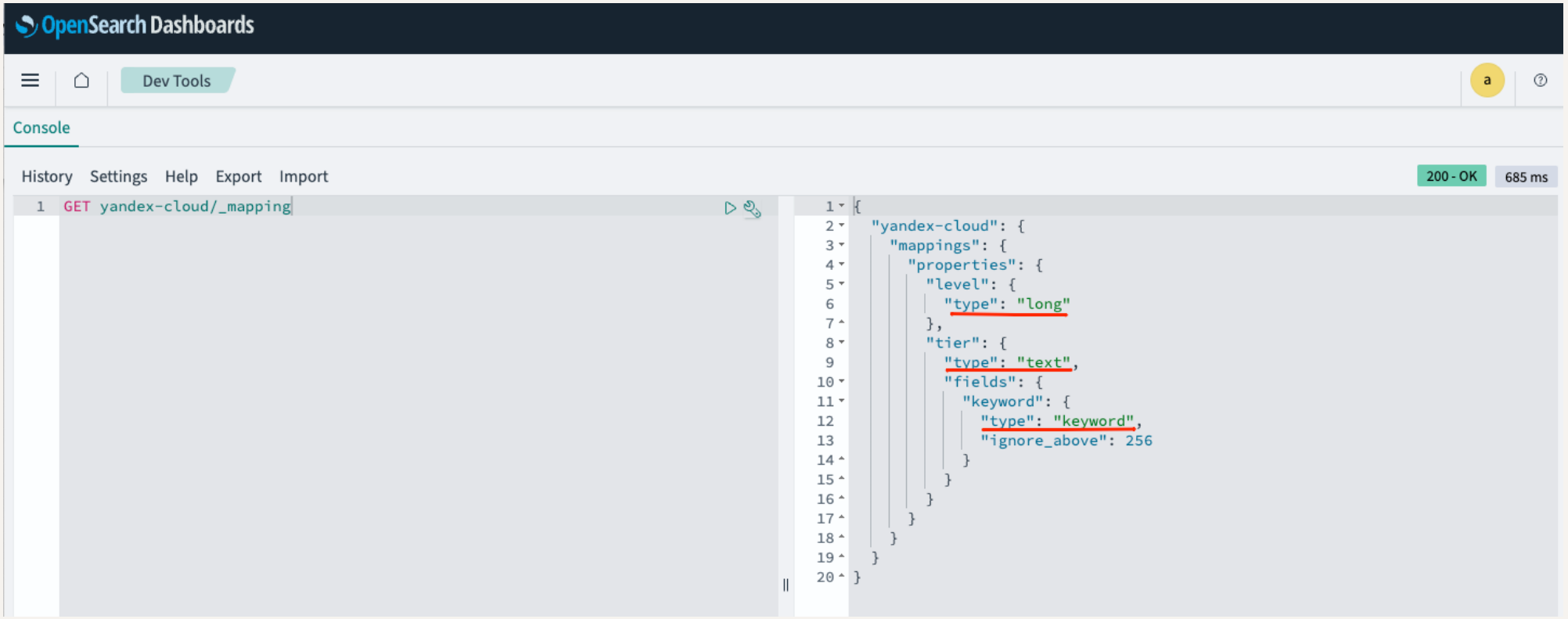
0 segment

Посмотрим на сегменты

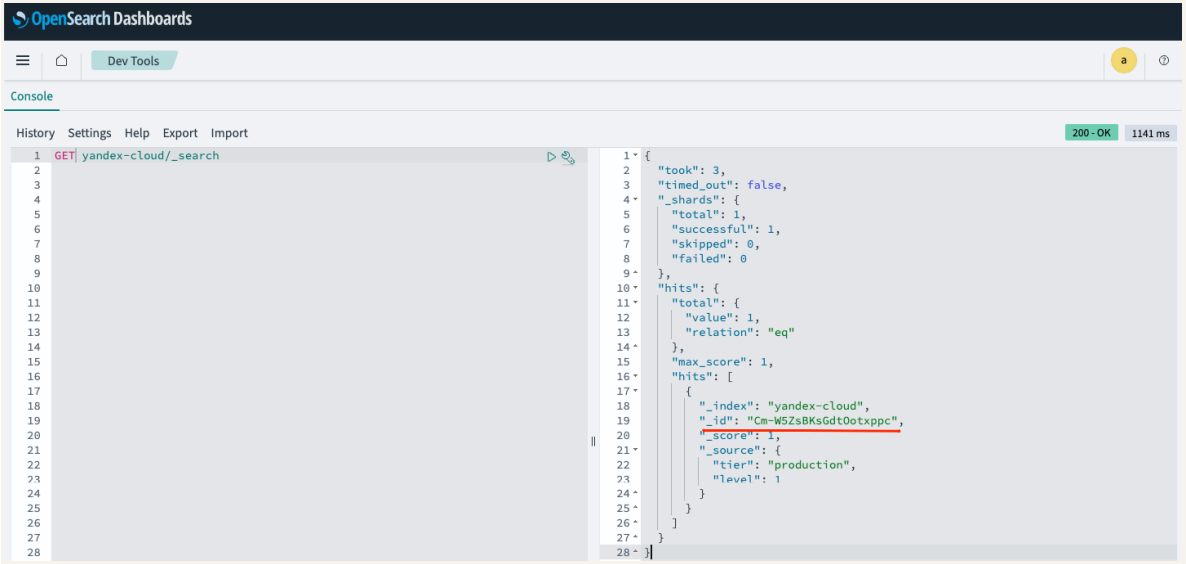


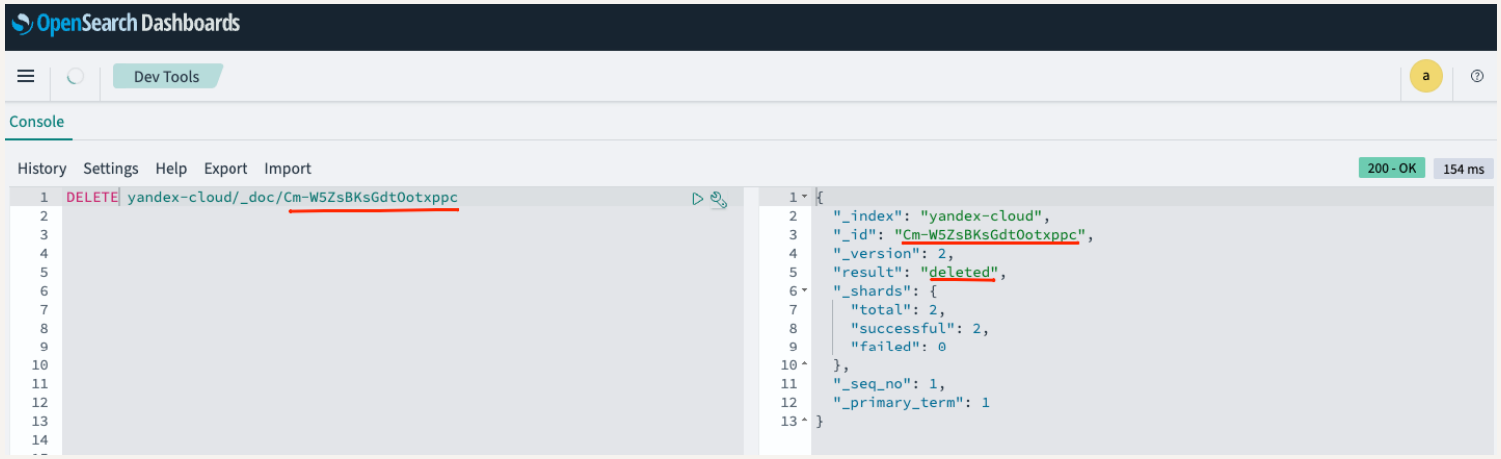
Обратите внимание на размер индекса

Проверим созданный маппинг полей

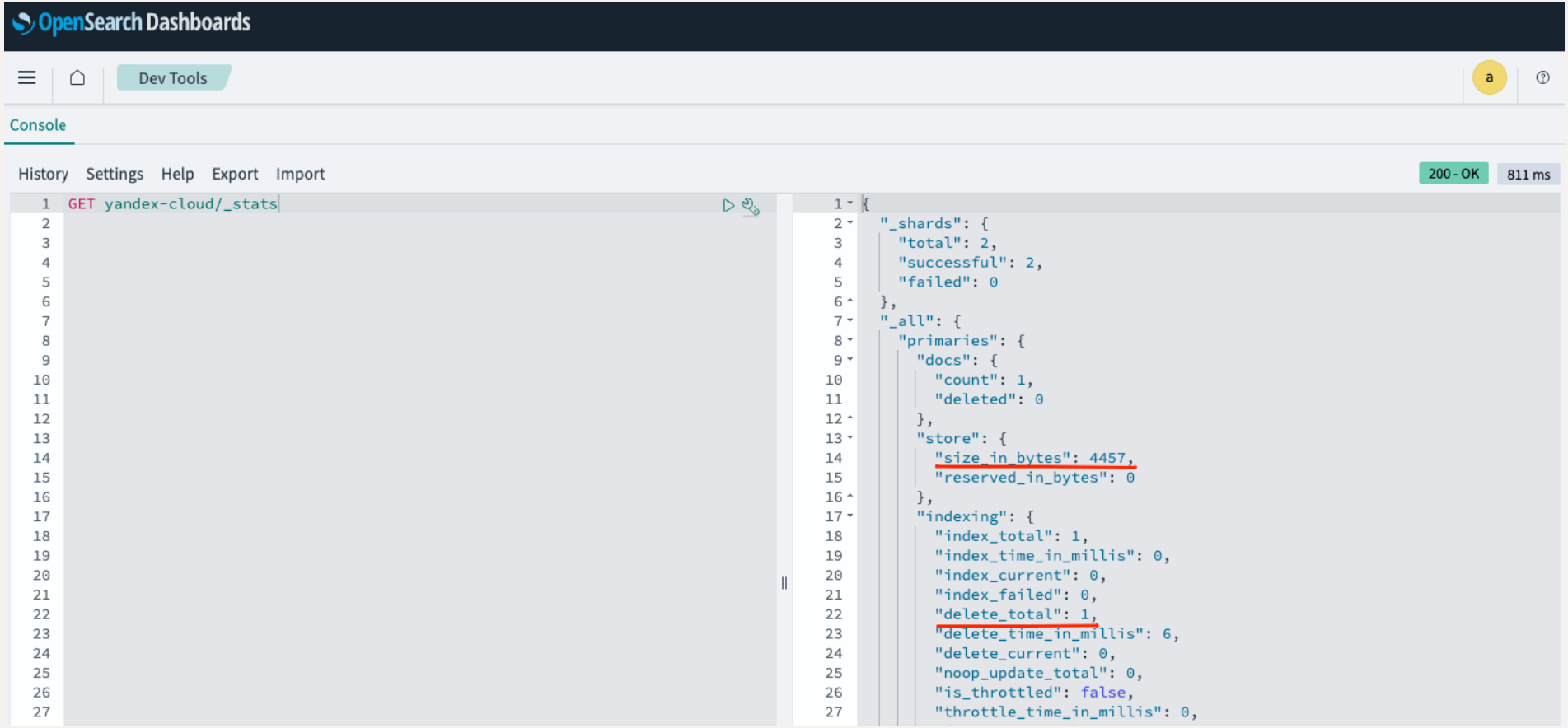


Удалим документ



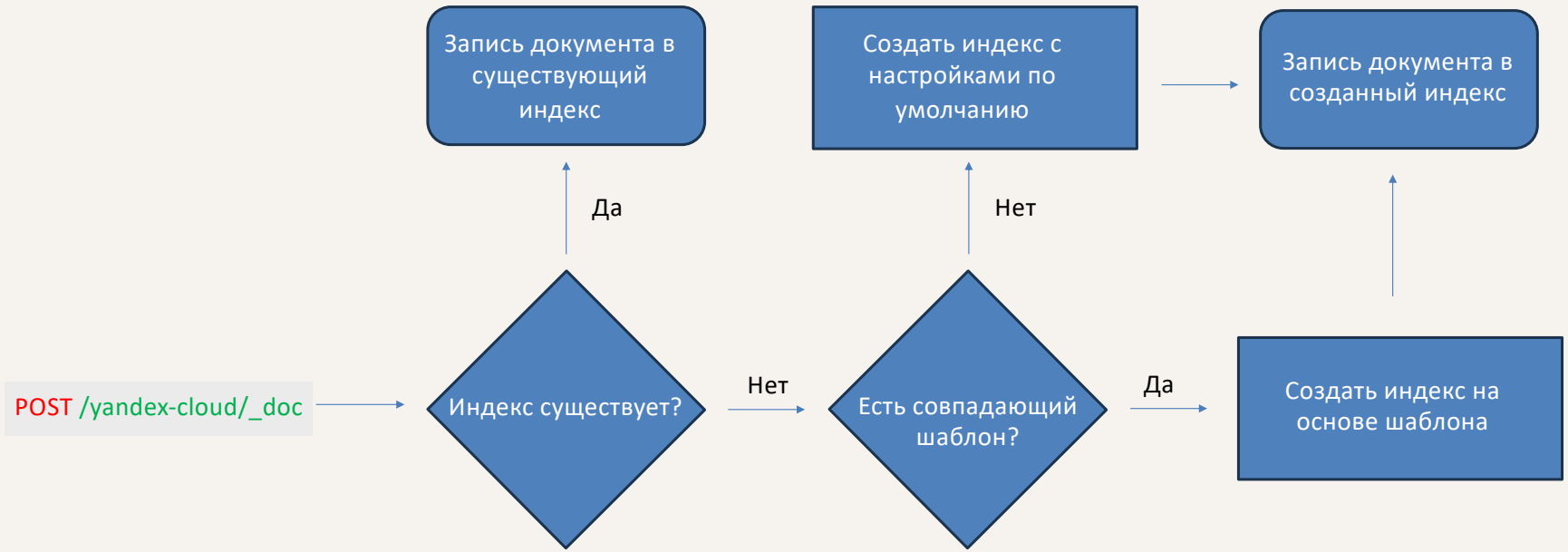


Смотрим размер индекса



Обратите внимание на размер индекса

Как создается индекс



Как еще можно протестировать

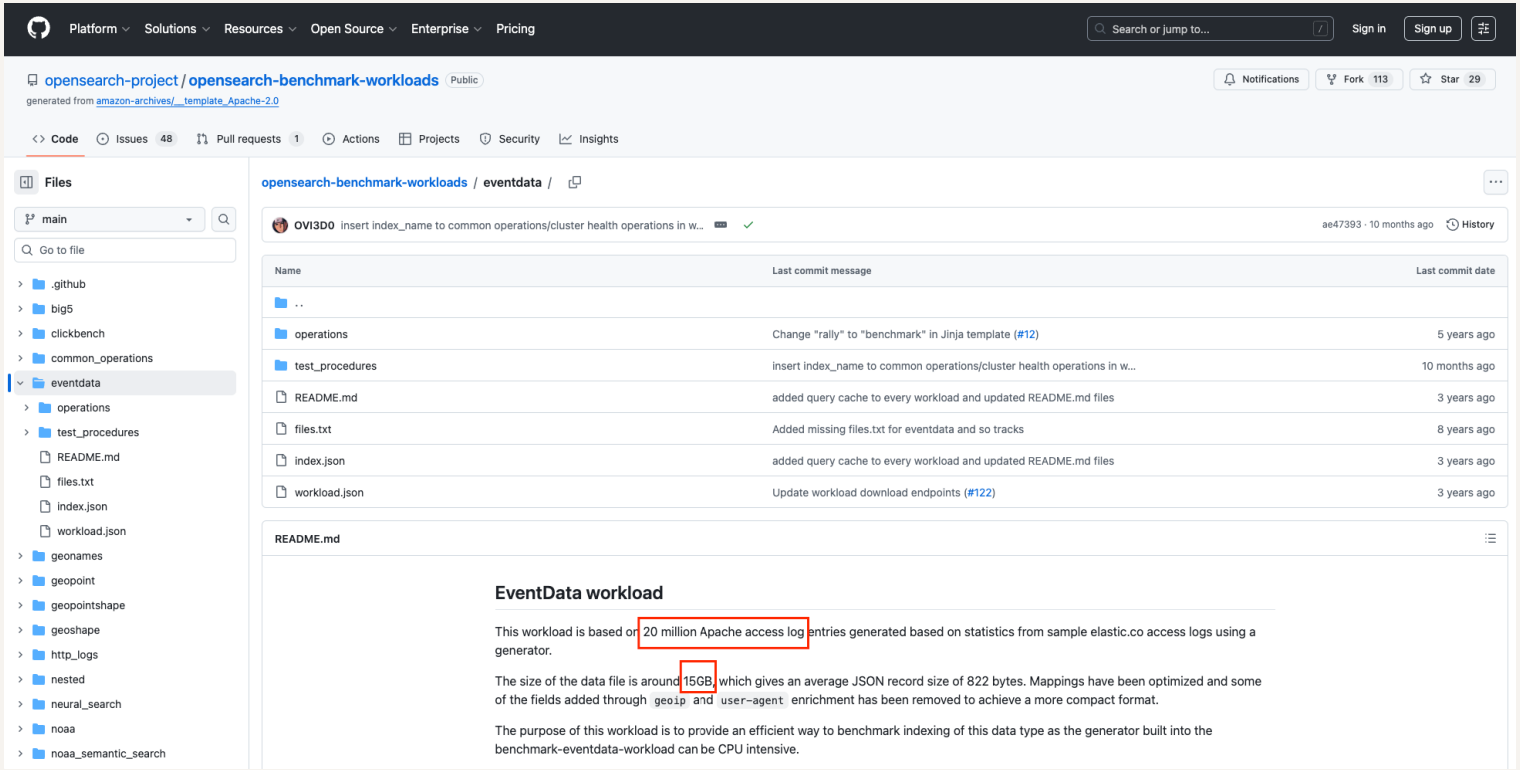
https://docs.opensearch.org/latest/benchmark

Датасет eventlog

Логи Apache

20 млн документов

~15 Гб сырых данных



https://github.com/opensearch-project/opensearch-benchmark-workloads/tree/main/eventdata

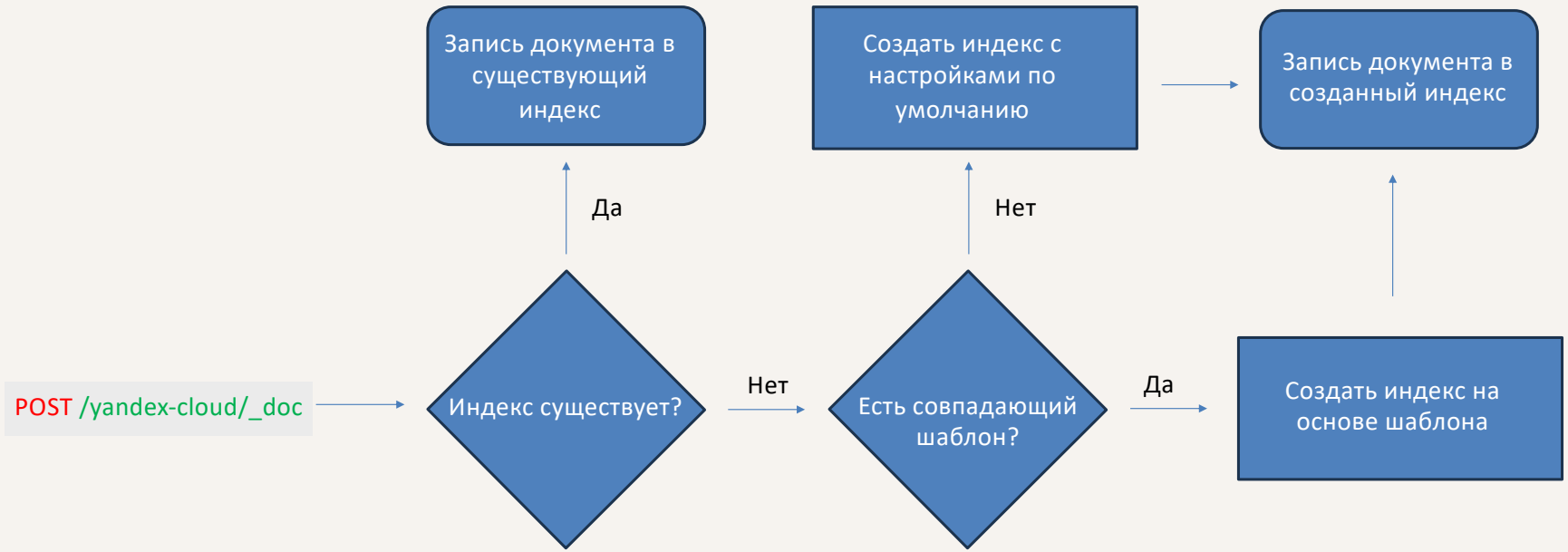
Датасет eventlog

Позже мы его загрузим в кластер и посмотрим...

КАК ОПТИМИЗИРОВАТЬ ХРАНЕНИЕ

Работать с маппингом

Как создается индекс



Динамический маппинг полей (dynamic)

JSON	OPENSEARCH
string	• text + keyword • date • float или long
integer	long
float	float
boolean	boolean
object	object
array	зависит от первого не пустого значения

Как выглядит маппинг

PostgreSQL

```
CREATE TABLE pages (
   rating      decimal(40),
   content      varchar(40),
   product_id       integer,
   author_first_name varchar(40),
   author_last_name varchar(40),
   author_email varchar(40),
   CONSTRAINT product_id PRIMARY KEY(product_id)
);
```

OpenSearch

```
PUT pages
{
 "mappings": {
  "properties": {
   "rating": { "type": "float" },
   "content": { "type": "text" },
   "product_id": { "type": "integer" },
   "author": {
    "properties": {
     "first_name": { "type": "text" },
     "last_name": { "type": "text" },
     "email": { "type": "keyword" }
    }
   }
  }
 }
}
```

Делаем explicit-маппинг

Типы полей

object

long

float

keyword

double

text

short

integer

date

* это далеко не все типы

Типы полей

keyword

text

Анализаторы (standard)

```
POST /_analyze
{
  "text": "Съешь ещё.  .. этих мягких французских булок, да выпей 2 литра сока!!!!!",
  "analyzer": "standard"
}
```

=

```
POST /_analyze
{
  "text": "Съешь ещё.  .. этих мягких французских булок, да выпей 2 литра сока!!!!!",
  "char_filter": [],
  "tokenizer": "standard",
  "filter": ["lowercase"]
}
```

Анализатор (standard)

Результат прохождения через анализатор

```
POST /_analyze
{
  "text": "Съешь ещё.  .. этих мягких французских булок, да выпей 2 литра сока!!!!!",
  "analyzer": "standard"
}
```


```
{
  "tokens": [
    {
      "token": "съешь",
      "start_offset": 0,
      "end_offset": 5,
      "type": "<ALPHANUM>",
      "position": 0
    },
    {
      "token": "ещё",
      "start_offset": 6,
      "end_offset": 9,
      "type": "<ALPHANUM>",
      "position": 1
    },
    {
      "token": "этих",
      "start_offset": 15,
      "end_offset": 19,
      "type": "<ALPHANUM>",
      "position": 2
    },
    {
      "token": "мягких",
      "start_offset": 20,
      "end_offset": 26,
      "type": "<ALPHANUM>",
      "position": 3
    },
    {
      "token": "французских",
      "start_offset": 27,
      "end_offset": 38,
      "type": "<ALPHANUM>",
      "position": 4
```

Инвертированный индекс

DOCUMENT #1
"text": "Съешь ещё. .. этих мягких французских булок, да выпей 2 литра сока!!!!!"

DOCUMENT #2
"text": "Съешь мягких булок"

TERM	DOCUMENT #1	DOCUMENT #2
съешь	x	x
ещё	x	
этих	x	
мягких	x	x
французских	x	
булок	x	x
да	x	
выпей	x	
2	x	
литра	x	
сока	x	

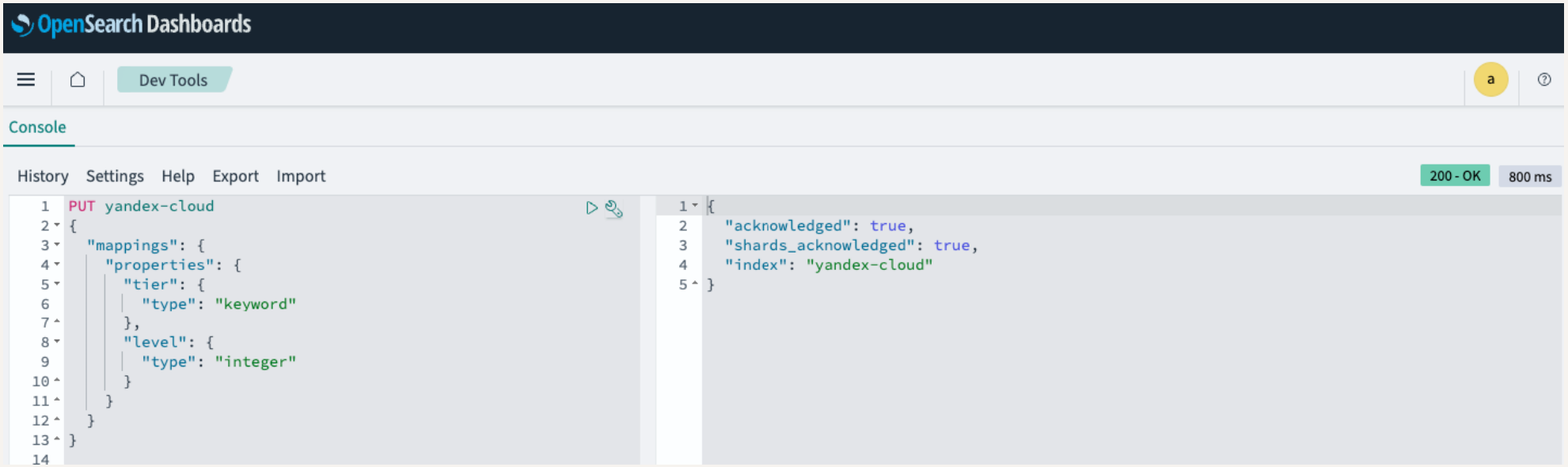
Анализатор (keyword)

```
POST /_analyze
{
  "text": "Съешь ещё.  .. этих мягких французских булок, да выпей 2 литра сока!!!!!",
  "analyzer": "keyword"
}
```

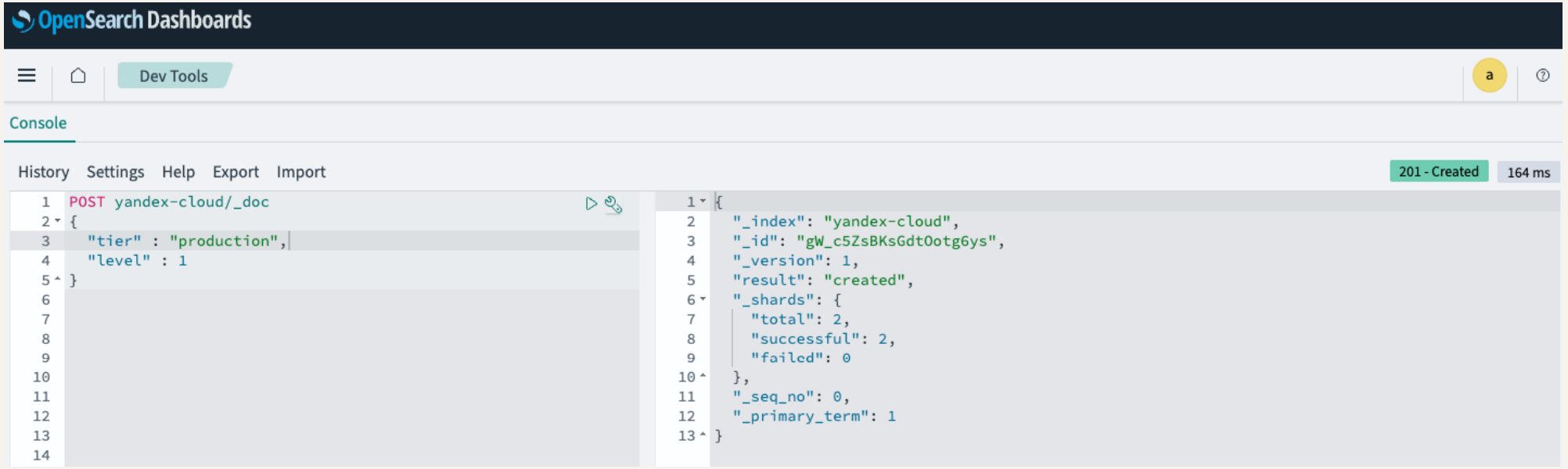


```
{
  "tokens": [
    {
      "token": "Съешь ещё.  .. этих мягких французских булок, да выпей 2 литра сока!!!!!",
      "start_offset": 0,
      "end_offset": 72,
      "type": "word",
      "position": 0
    }
  ]
}
```

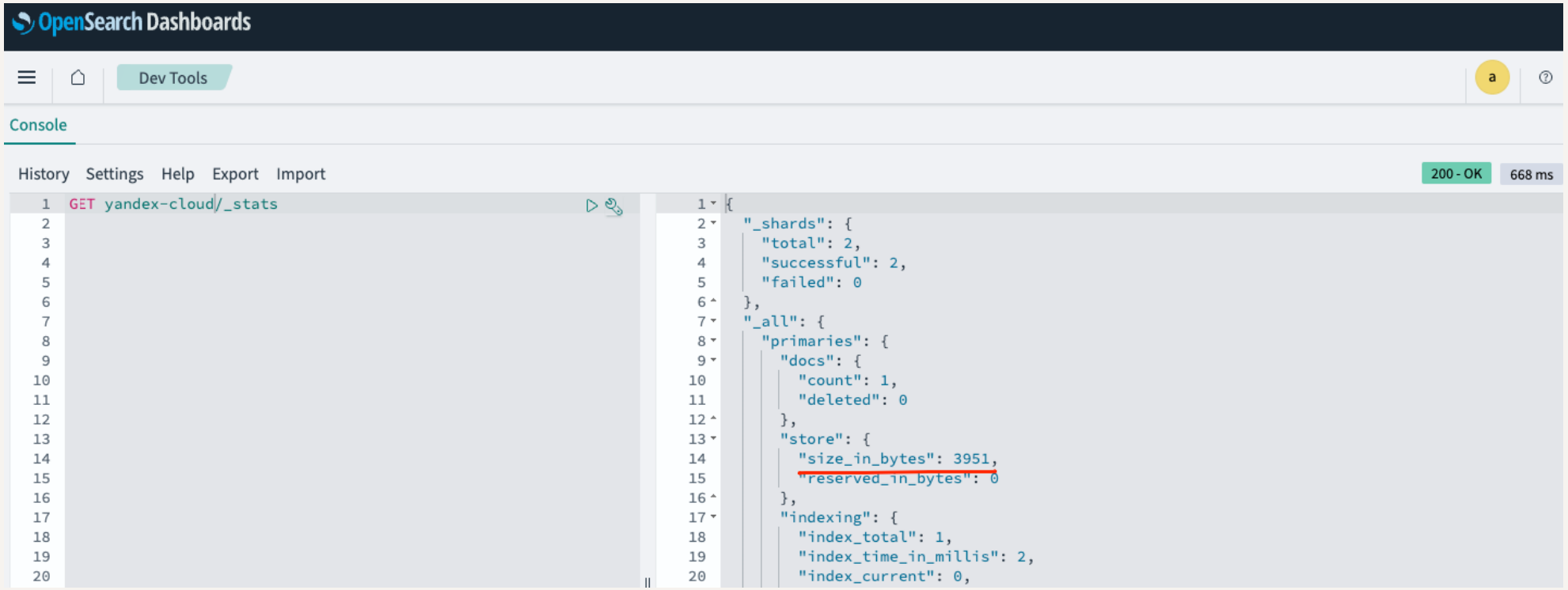
Создаем маппинг



Записываем документ



Получаем результат

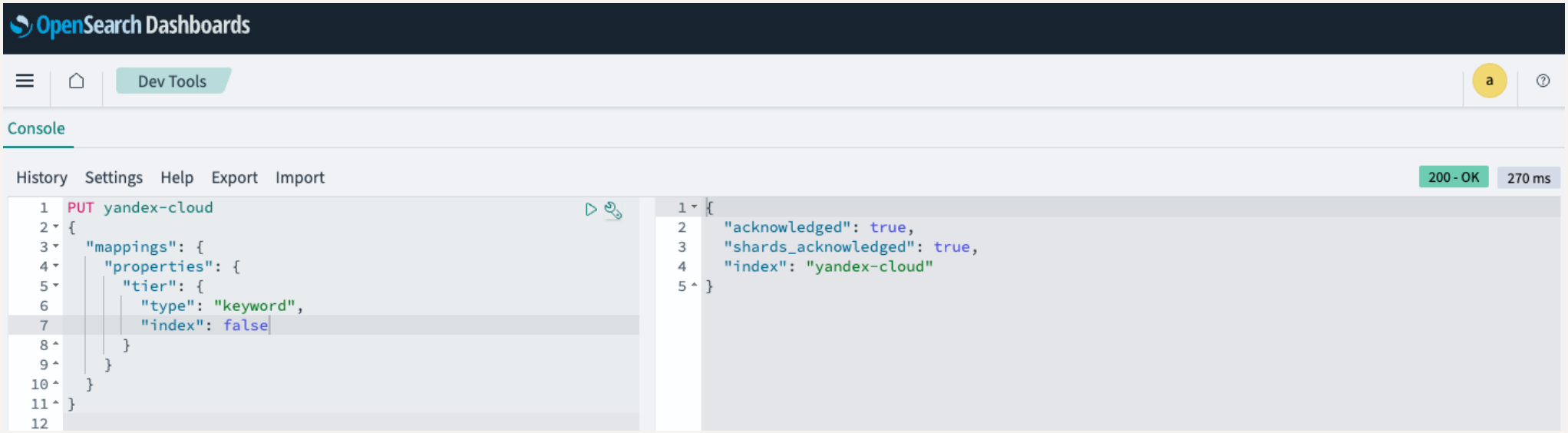


* было 4143

Параметры маппинга

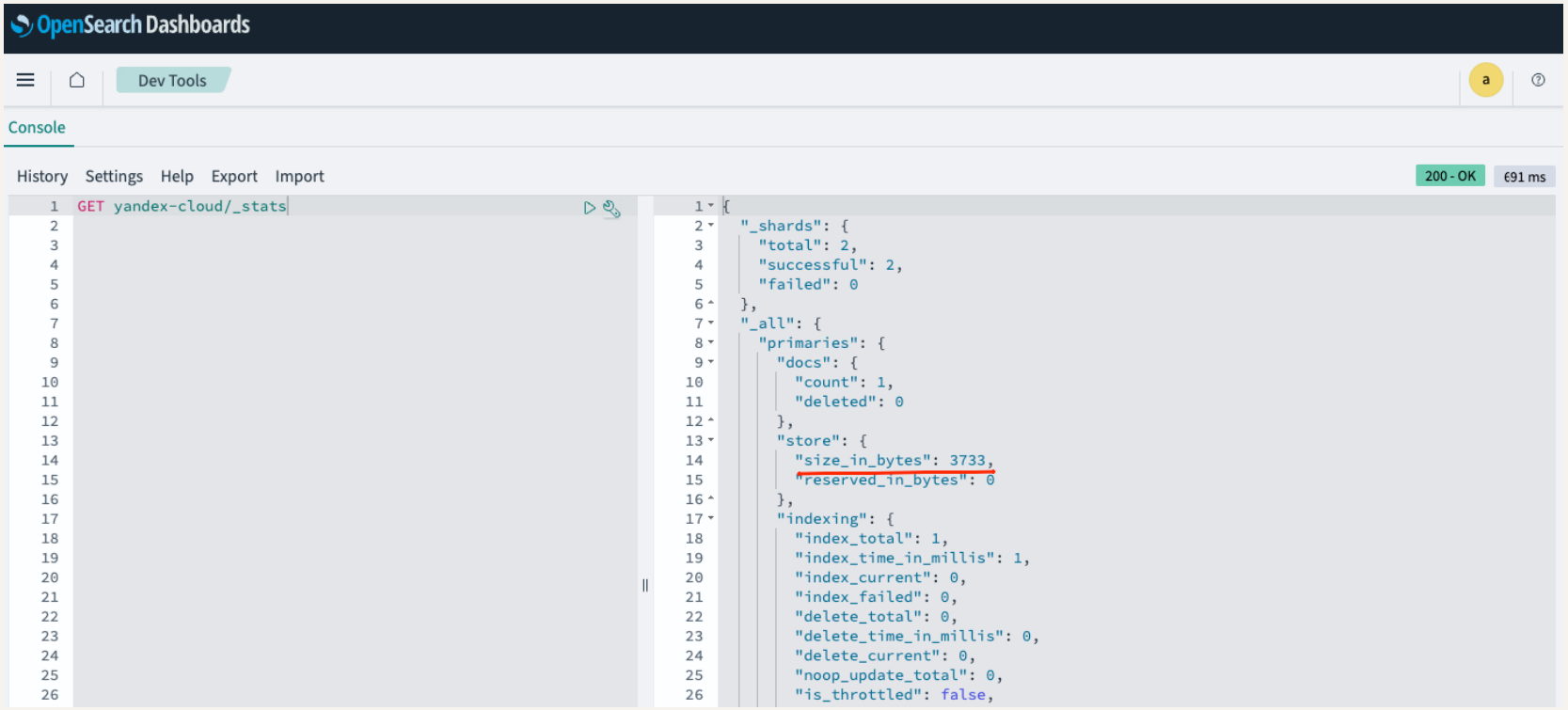
analyzer, coerce, copy_to, doc_values, dynamic, eager_global_ordinals, enabled, format, index.mapping. ignore_above, ignore_malformed, index, index_options, index_phrases, index_prefixes, meta, fields, normalizer, norms, null_value, position_increment_gap, properties, search_analyzer, similarity, store, subobjects, term_vector

Добавляем параметр index



* отключение записи в инвертированный индекс

Смотрим размер



Отключение скоринга для поля (norms)

```
PUT yandex-cloud
{
  "properties": {
    "title": {
      "type": "text",
      "norms": false
    }
  }
}
```

Принудительная типизация (coerce)

```
PUT yandex-cloud
{
  "settings": {
    "index.mapping.coerce": false
  },
  "mappings": {
    "properties": {
      "number_one": {
        "type": "integer",
        "coerce": true
      },
      "number_two": {
        "type": "integer"
      }
    }
  }
}
```

```
PUT yandex-cloud/_doc/1
{ "number_one": "10" }
```

OK

```
PUT yandex-cloud/_doc/1
{ "number_two": "10" }
```

REJECT

Тип поля match_only_text

Экономия на хранении до 25% по сравнению с типом text

Отключен скоринг, учет позиций и частоты

```
PUT yandex-cloud
{
  "mappings": {
    "properties": {
      "tier": {
        "type": "match_only_text"
      }
    }
  }
}
```

Отключение динамического маппинга

Отключение динамического маппинга (дополнительные поля документов не будут проиндексированы, но запишутся в _source)

```
PUT yandex-cloud
{
  "mappings": {
    "dynamic": false,
    "properties": {
      "tier": {
        "type": "text"
      }
    }
  }
}
```

Ограничение создания документов (документ, содержащий дополнительные поля, будет отброшен)

```
PUT yandex-cloud
{
  "mappings": {
    "dynamic": strict,
    "properties": {
      "tier": {
        "type": "text"
      }
    }
  }
}
```

Полу-динамический маппинг

Подсказывает OpenSearch правильный тип данных

```
PUT /dynamic_template_test
{
  "mappings": {
    "dynamic_templates": [
      {
        "integers": {
          "match_mapping_type": "long",
          "mapping": {
            "type": "integer"
          }
        }
      }
    ]
  }
}
```

Полу-динамический маппинг

Типизация по названию полей (match и unmatch)

```
PUT yandex-cloud
{
 "mappings": {
  "dynamic_templates": [
   {
    "strings_only_text": {
     "match_mapping_type": "string",
     "match": "text_*",
     "unmatch": "*_keyword",
     "mapping": {
      "type": "text"
     }
    }
   },
   {
    "strings_only_keyword": {
     "match_mapping_type": "string",
     "match": "*_keyword",
     "mapping": {
      "type": "keyword"
     }
    }
   }
  ]
 }
}
```

Изменить кодек для сжатия

Что такое кодеки

> **Кодеки индекса определяют способ сжатия и хранения на диске полей, хранящихся в индексе. Кодек индекса управляется статической настройкой index.codec, определяющей алгоритм сжатия. Эта настройка влияет на размер сегмента индекса и производительность операций индексирования.**

Какие бывают кодеки

default — этот кодек использует алгоритм LZ4 с предустановленным словарем, который отдает приоритет производительности, а не степени сжатия. Он обеспечивает более быструю индексацию и поиск по сравнению с другими кодеками best_compression, но может привести к увеличению размера индекса/шарда. Если в настройках индекса не указан кодек, то в качестве алгоритма сжатия по умолчанию используется LZ4.

best_compression — этот кодек использует zlib в качестве базового алгоритма сжатия. Он обеспечивает высокие коэффициенты сжатия, что приводит к уменьшению размеров индексов. Однако это может повлечь за собой дополнительную нагрузку на CPU во время операций индексирования и, следовательно, привести к высоким задержкам индексирования и поиска.

zstd (OpenSearch 2.9 и более поздние) — этот кодек обеспечивает значительное сжатие, сравнимое с кодеком best_compression, при разумном использовании CPU и улучшенной производительности индексирования и поиска по сравнению с кодеком default.

zstd_no_dict (OpenSearch 2.9 и более поздние) – этот кодек похож на zstd, но не включает функцию сжатия словаря. Он обеспечивает более быструю индексацию и операции поиска по сравнению с zstd, но за счет немного большего размера индекса.

Какие бывают кодеки

default — этот кодек использует алгоритм LZ4 с предустановленным словарем, который отдает приоритет производительности, а не степени сжатия. Он обеспечивает более быструю индексацию и поиск по сравнению с другими кодеками best_compression, но может привести к увеличению размера индекса/шарда. Если в настройках индекса не указан кодек, то в качестве алгоритма сжатия по умолчанию используется LZ4.

best_compression — этот кодек использует zlib в качестве базового алгоритма сжатия. Он обеспечивает высокие коэффициенты сжатия, что приводит к уменьшению размеров индексов. Однако это может повлечь за собой дополнительную нагрузку на CPU во время операций индексирования и, следовательно, привести к высоким задержкам индексирования и поиска.

zstd (OpenSearch 2.9 и более поздние) — этот кодек обеспечивает значительное сжатие, сравнимое с кодеком best_compression, при разумном использовании CPU и улучшенной производительности индексирования и поиска по сравнению с кодеком default.

zstd_no_dict (OpenSearch 2.9 и более поздние) – этот кодек похож на zstd, но не включает функцию сжатия словаря. Он обеспечивает более быструю индексацию и операции поиска по сравнению с zstd, но за счет немного большего размера индекса.

Дополнительное сжатие

> **zstd** и **zstd_no_dict** дополнительно позволяют указать степень сжатия [1..6]. По умолчанию 3.

> Более высокий уровень сжатия приводит к более высокому коэффициенту сжатия (меньшему объему памяти) с ухудшением скорости (более низкие скорости сжатия и распаковки приводят к большей задержке индексирования и поиска).

```
PUT yandex-cloud
{
  "settings": {
    "index.codec": "zstd",
    "index.codec.compression_level": 6
  }
}
```

Датасет eventdata

Логи Apache, 20 млн записей, 15 Гб в сыром виде

```
root@vm-opensearch01:~# ls -la /root/.osb/benchmarks/data/eventdata/eventdata.json
-rw-r--r-- 1 root root 16437108429 Jan 27 14:38 /root/.osb/benchmarks/data/eventdata/eventdata.json
```

Example Document


```
{
    "agent": "\"Mozilla/5.0 (Macintosh; Intel Mac OS X 10_10_2) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/48.0.2
    "useragent": {
        "os": "Mac OS X 10.10.2",
        "os_name": "Mac OS X",
        "name": "Chrome"
    },
    "geoip": {
        "country_name": "India",
        "location": [80.2833, 13.083300000000008]
    },
    "clientip": "122.178.238.140",
    "referrer": "\"-\"",
    "request": "/apple-touch-icon-144x144.png",
    "bytes": 0,
    "verb": "GET",
    "response": 304,
    "httpversion": "1.1",
    "@timestamp": "2017-07-03T07:51:49.995Z",
    "message": "122.178.238.140 - - [2017-07-03T07:51:49.995Z] \"GET /apple-touch-icon-144x144.png HTTP/1.1\" 304 0 '
}
```

Загрузим eventdata с разной степенью сжатия

> **default vs zstd 6**

Indexes (2)

Refresh | Actions | + Create Index

event

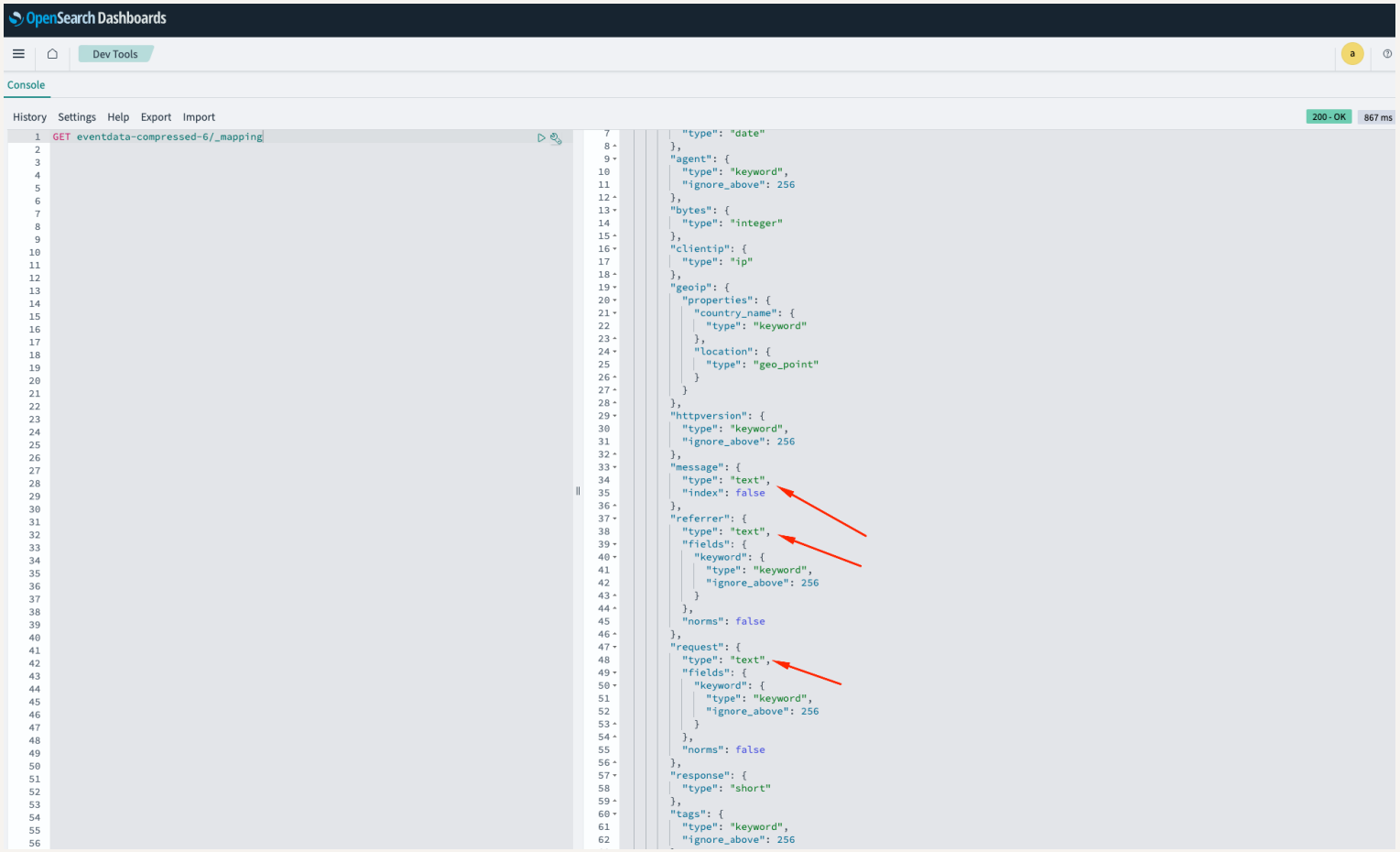
Show data stream indexes

Index	Health	Managed by policy	Status	Total size	Size of primaries	Total documents	Deleted documents	Primaries	Replicas
eventdata-compressed-6	Green	No	Open	3gb	3gb	20000000	0	5	0
eventdata	Green	No	Open	4.9gb	4.9gb	20000000	0	5	0

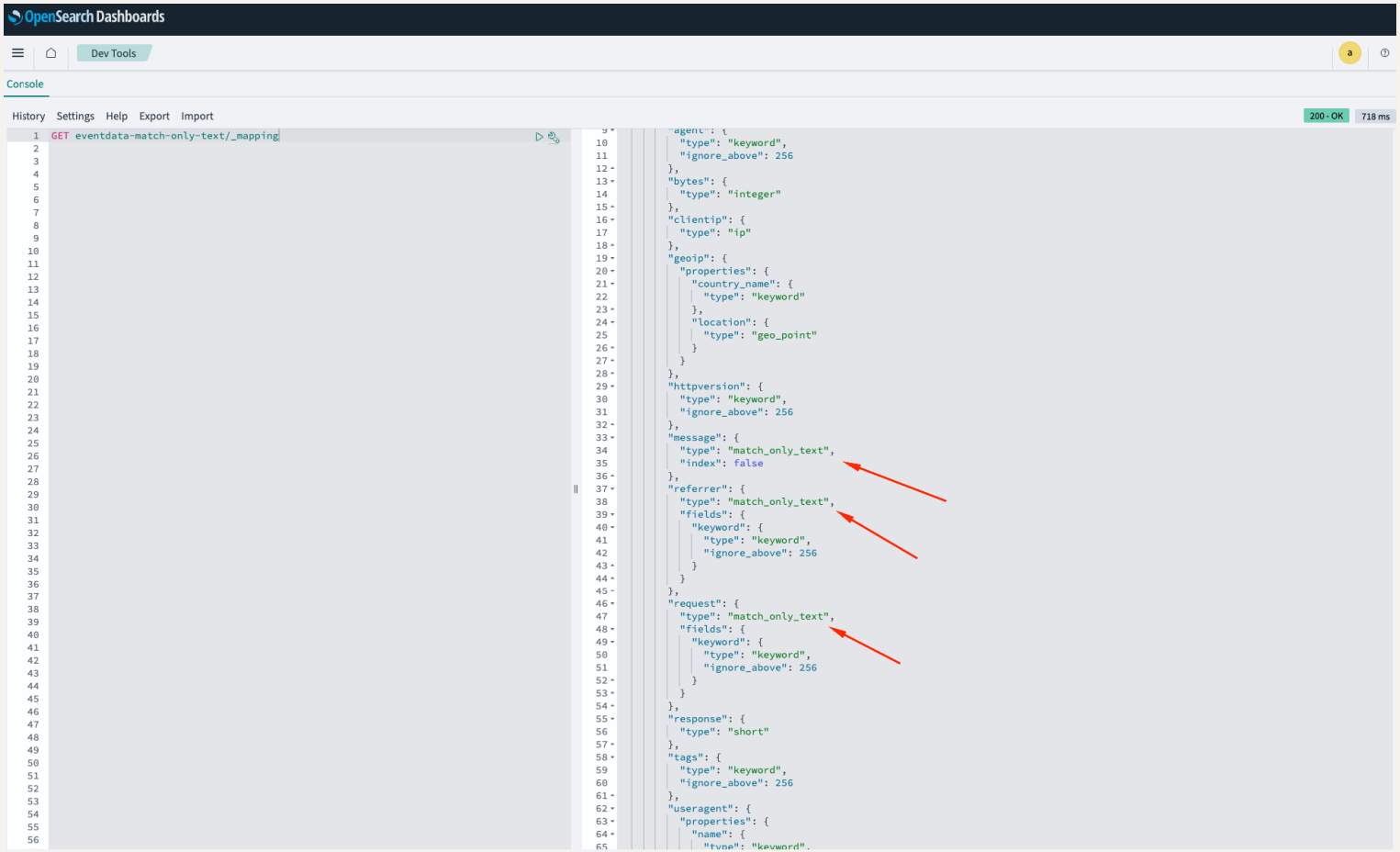
Rows per page: 20

1

Смотрим на маппинг

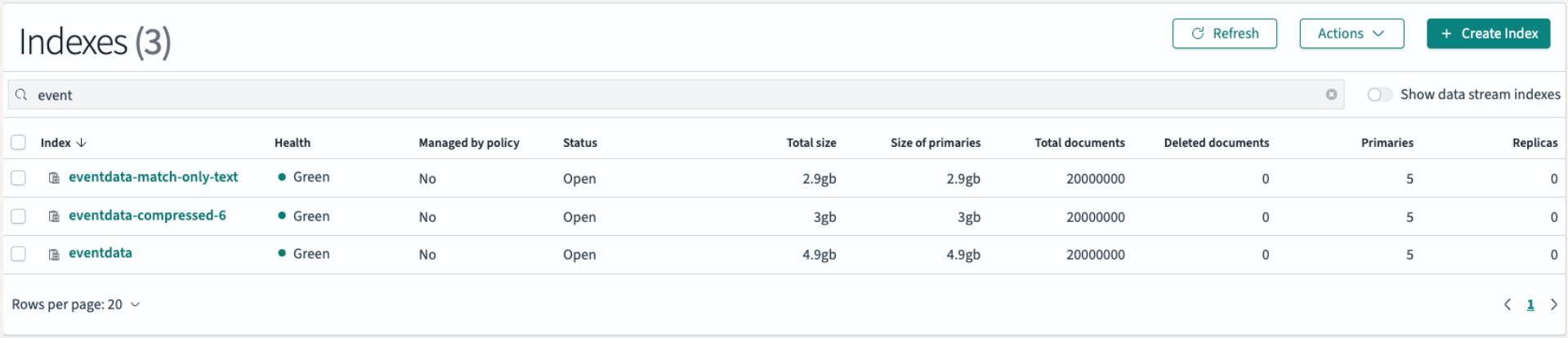


Вспоминаем про match_only_text



Загрузим eventdata снова

text vs match_only_text



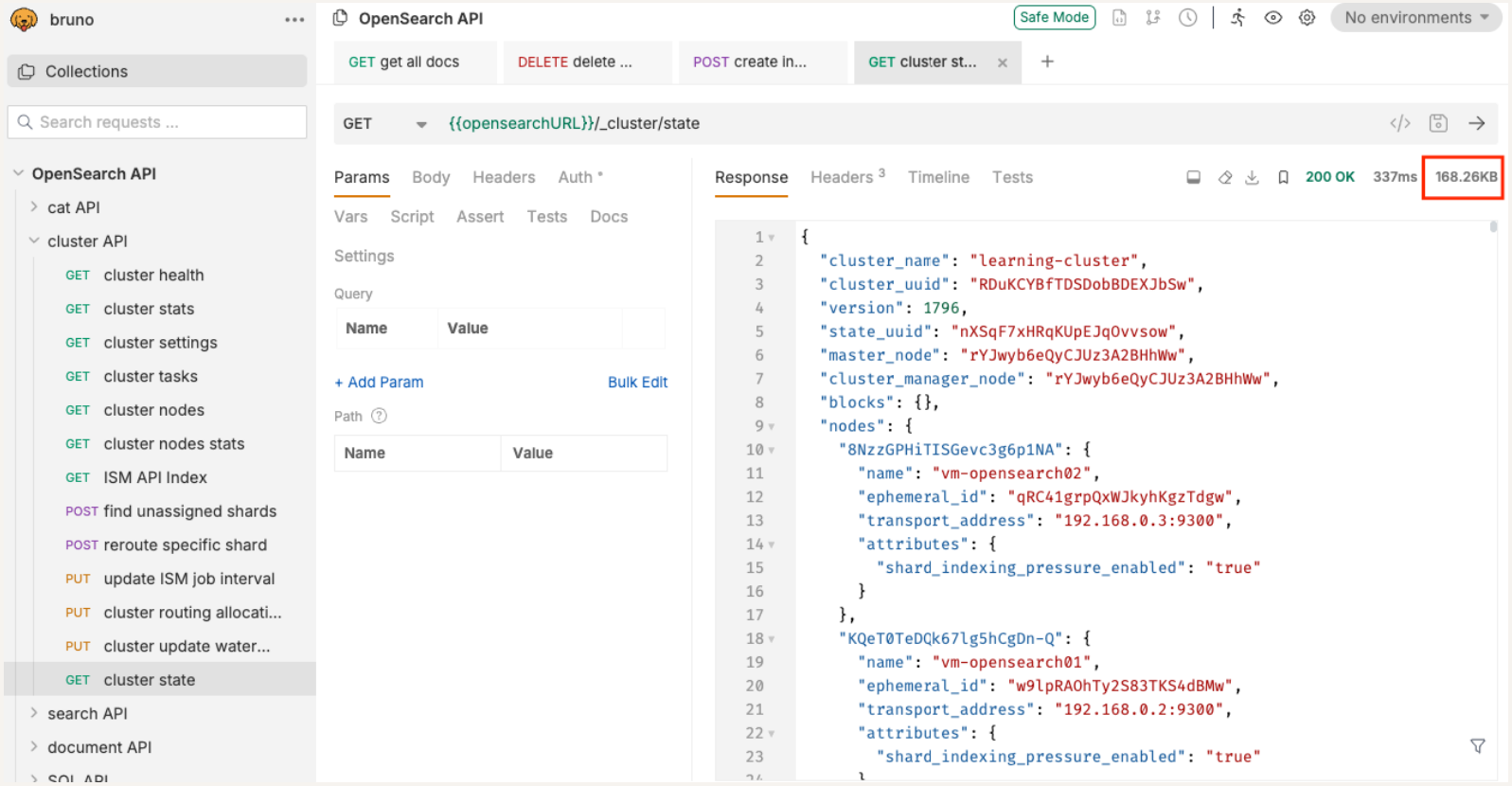
Работать с cluster_state

Определение cluster_state

Внутренняя структура данных, которая отслеживает информацию, необходимую каждому узлу. Распространяется при помощи мастер-ноды. Включает в себя:

- Идентификаторы и атрибуты других узлов в кластере.
- Настройки для всего кластера.
- Метаданные индекса, включая сопоставление и настройки для каждого индекса.
- Местоположение и состояние каждой копии сегмента в кластере.
- Выбранный главный узел гарантирует, что каждый узел в кластере имеет копию одного и того же состояния кластера. API состояния кластера позволяет получить представление этого внутреннего состояния для целей отладки или диагностики.

Определение текущего размера _state



Если 

, то 

Делаем как для себя.

gals.software