

# Оптимизация хранения данных в OpenSearch



Антон Касимов



**Поговорим о модели хранения  
данных и методах оптимизации  
хранения в OpenSearch**

# Что вас ждет:

Как хранятся  
данные

Как  
оптимизировать  
хранение



**СПЕЦИАЛИЗИРОВАННЫЙ СИСТЕМНЫЙ ИНТЕГРАТОР  
(СИСТЕМЫ МОНИТОРИНГА)**

**ВНЕДРЕНИЕ И НАСТРОЙКА OPENSEARCH**

**ТЕХНИЧЕСКАЯ ПОДДЕРЖКА OPENSEARCH**

**ОБУЧЕНИЕ НА ТРЕНИНГАХ «БАЗА»  
И «ПРОДВИНУТЫЙ»**

**ПАРТНЕР Yandex Cloud**

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

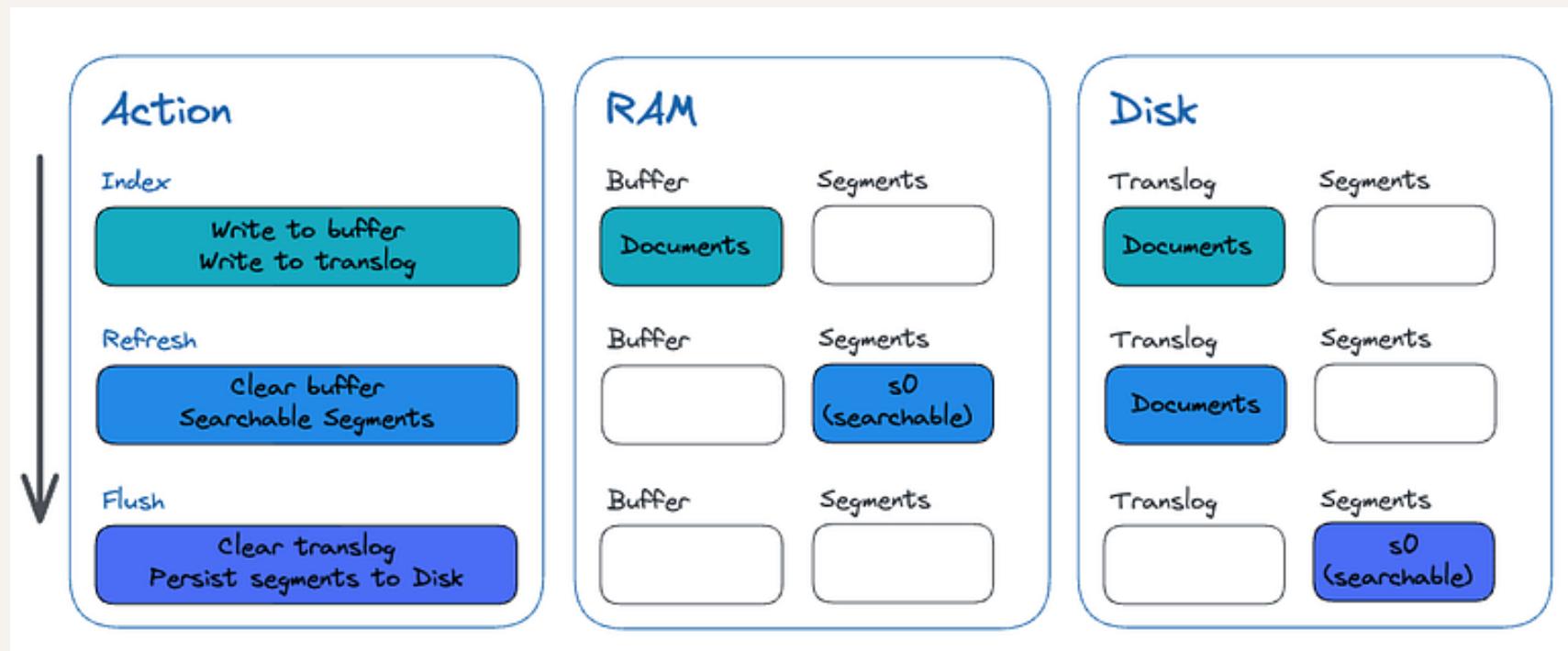
Работа с **Vector, Logstash, DataPrepper, OpenTelemetry Collector, Ingest Pipeline** и **Kafka**



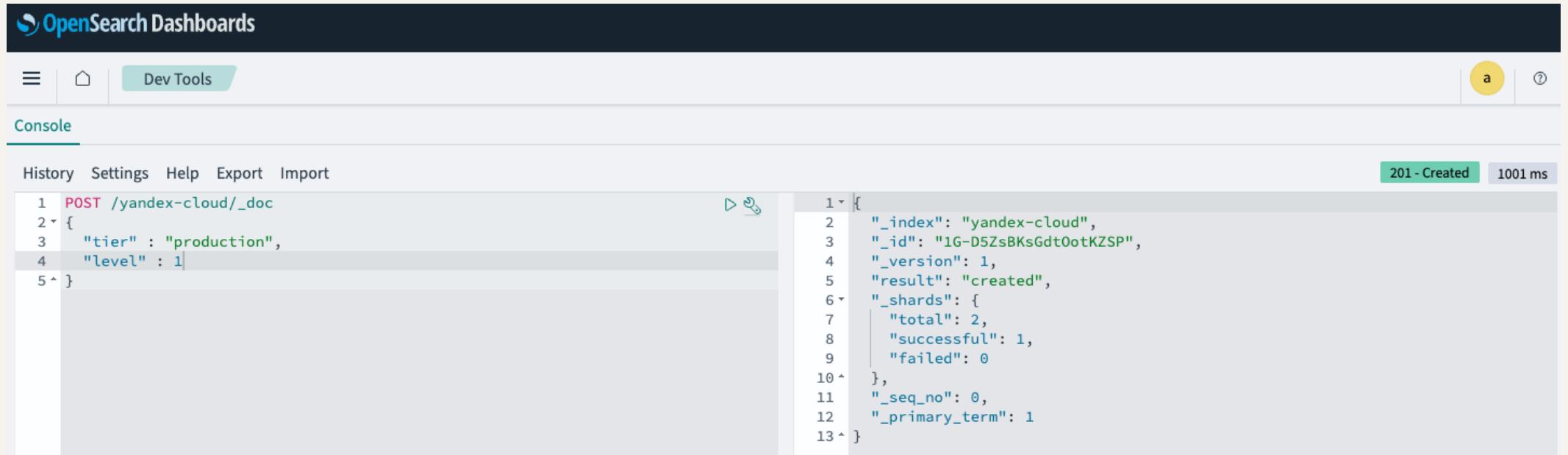
Программа курса

**КАК  
ХРАНЯТСЯ  
ДАННЫЕ**

# Как хранятся данные в OpenSearch



# Запишем документ в индекс



OpenSearch Dashboards

Dev Tools

Console

History Settings Help Export Import

201 - Created 1001 ms

```
1 POST /yandex-cloud/_doc
2 {
3   "tier" : "production",
4   "level" : 1
5 }
```

```
1 [
2   "_index": "yandex-cloud",
3   "_id": "1G-D5ZsBKsGdt0otKZSP",
4   "_version": 1,
5   "result": "created",
6   "_shards": {
7     "total": 2,
8     "successful": 1,
9     "failed": 0
10  },
11  "_seq_no": 0,
12  "_primary_term": 1
13 ]
```

Почему total = 2, successful = 1?

# Посмотрим на шарды и сегменты

OpenSearch Dashboards

Dev Tools

Console

History Settings Help Export Import 200 - OK 1264 ms

```
1 GET yandex-cloud/_settings
```

```
1 {  
2   "yandex-cloud": {  
3     "settings": {  
4       "index": {  
5         "replication": {  
6           "type": "DOCUMENT"  
7         },  
8           "number_of_shards": "1",  
9           "provided_name": "yandex-cloud",  
10          "creation_date": "1769083422133",  
11          "number_of_replicas": "1",  
12          "uuid": "KTCChhrtSJeQBjELUuujnA",  
13          "version": {  
14            "created": "137248027"  
15          }  
16        }  
17      }  
18    }  
19  }
```

```
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!

OpenSearch Dashboards

☰ ⌂ Dev Tools

Console

History Settings Help Export Import 200 - OK 908

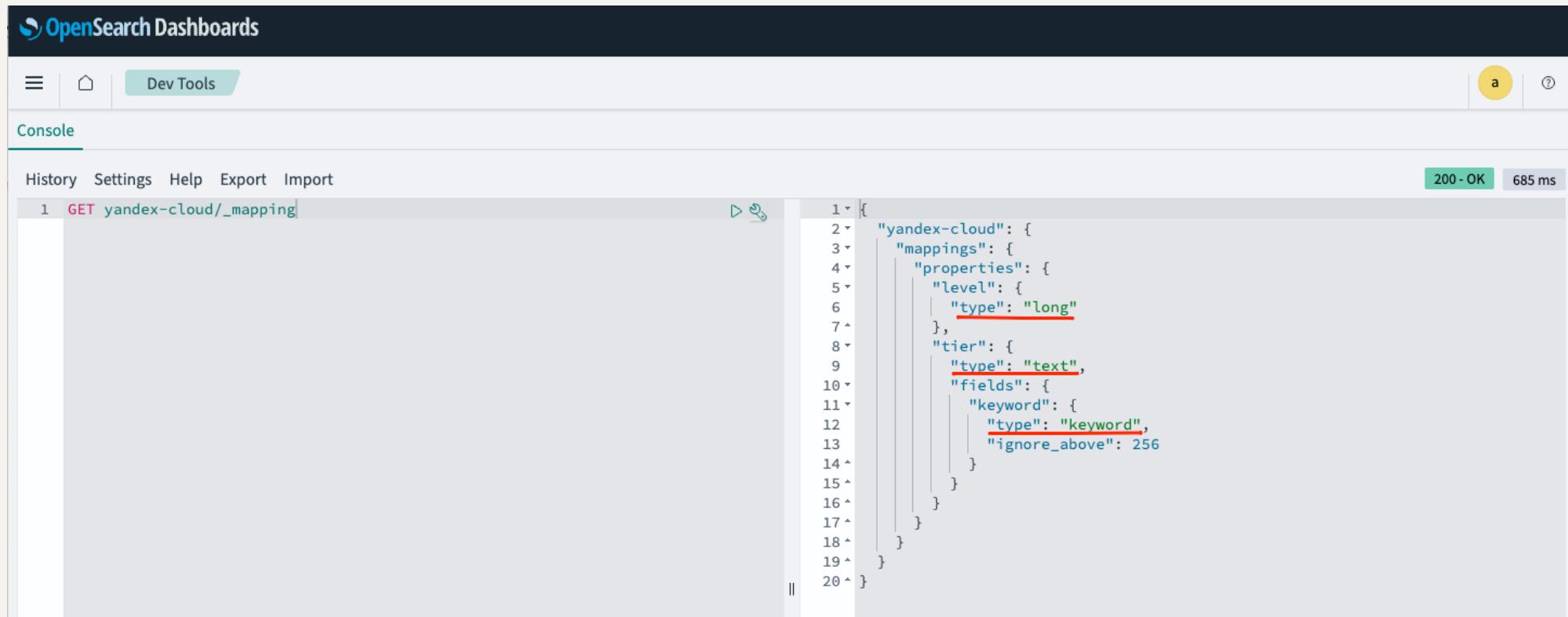
```
1 GET yandex-cloud/_segments
```

```
1 {  
2   "_shards": {  
3     "total": 2,  
4     "successful": 2,  
5     "failed": 0  
6   },  
7   "indices": {  
8     "yandex-cloud": {  
9       "shards": {  
10        "0": [  
11          {  
12            "routing": {  
13              "state": "STARTED",  
14              "primary": true,  
15              "node": "8NzzGPHiTISGevc3g6p1NA"  
16            },  
17            "num_committed_segments": 0,  
18            "num_search_segments": 1,  
19            "segments": {  
20              "_0": {  
21                "generation": 0,  
22                "num_docs": 1,  
23                "deleted_docs": 0,  
24                "size_in_bytes": 4143,  
25                "memory_in_bytes": 0,  
26                "committed": false,  
27                "search": true,  
28                "version": "10.3.1",  
29                "compound": true,  
30                "attributes": {  
31                  "Lucene90StoredFieldsFormat.mode": "BEST_SPEED"  
32                }  
33              }  
34            },  
35          },  
36        },  
37      },  
38    },  
39  },  
40}
```

```
1 {  
2   "_shards": {  
3     "total": 2,  
4     "successful": 2,  
5     "failed": 0  
6   },  
7   "indices": {  
8     "yandex-cloud": {  
9       "shards": {  
10        "0": [  
11          {  
12            "routing": {  
13              "state": "STARTED",  
14              "primary": true,  
15              "node": "8NzzGPHiTISGevc3g6p1NA"  
16            },  
17            "num_committed_segments": 0,  
18            "num_search_segments": 1,  
19            "segments": {  
20              "_0": {  
21                "generation": 0,  
22                "num_docs": 1,  
23                "deleted_docs": 0,  
24                "size_in_bytes": 4143,  
25                "memory_in_bytes": 0,  
26                "committed": false,  
27                "search": true,  
28                "version": "10.3.1",  
29                "compound": true,  
30                "attributes": {  
31                  "Lucene90StoredFieldsFormat.mode": "BEST_SPEED"  
32                }  
33              }  
34            },  
35          },  
36        },  
37      },  
38    },  
39  },  
40}
```

## Обратите внимание на размер индекса

# Проверим созданный маппинг полей



OpenSearch Dashboards

Dev Tools

Console

History Settings Help Export Import

1 GET yandex-cloud/\_mapping

200 - OK 685 ms

```
1 {  
2   "yandex-cloud": {  
3     "mappings": {  
4       "properties": {  
5         "level": {  
6           "type": "long"  
7         },  
8         "tier": {  
9           "type": "text",  
10          "fields": {  
11            "keyword": {  
12              "type": "keyword",  
13              "ignore_above": 256  
14            }  
15          }  
16        }  
17      }  
18    }  
19  }  
20 }
```

# Удалим документ

OpenSearch Dashboards

Dev Tools

Console

History Settings Help Export Import

```
1 GET yandex-cloud/_search
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
```

```
1 {{"took": 3,
2 "timed_out": false,
3 "_shards": {
4   "total": 1,
5   "successful": 1,
6   "skipped": 0,
7   "failed": 0
8 },
9 "hits": {
10   "total": {
11     "value": 1,
12     "relation": "eq"
13   },
14   "max_score": 1,
15   "hits": [
16     {
17       "_index": "yandex-cloud",
18       "_id": "Cm-W5zsBKsGdt0otxppc",
19       "_score": 1,
20       "_source": {
21         "tier": "production",
22         "level": 1
23       }
24     }
25   ]
26 }
27 }
```

200 - OK 1141 ms

OpenSearch Dashboards

Dev Tools

Console

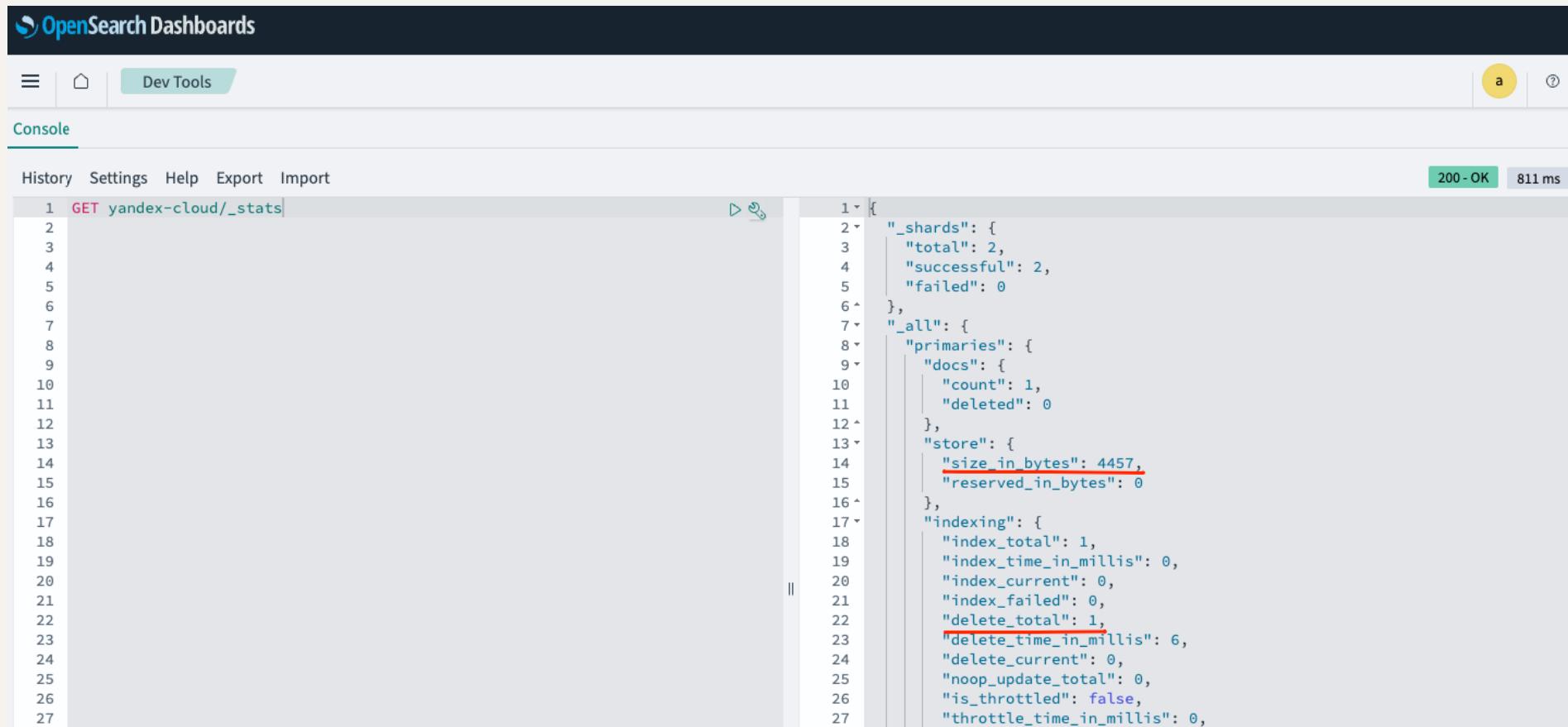
History Settings Help Export Import

```
1 DELETE yandex-cloud/_doc/Cm-W5zsBKsGdt0otxppc
2
3
4
5
6
7
8
9
10
11
12
13
14
```

```
1 {{"_index": "yandex-cloud",
2   "_id": "Cm-W5zsBKsGdt0otxppc",
3   "_version": 2,
4   "result": "deleted",
5   "_shards": {
6     "total": 2,
7     "successful": 2,
8     "failed": 0
9   },
10  "_seq_no": 1,
11  "_primary_term": 1
12 }
```

200 - OK 154 ms

# Смотрим размер индекса



OpenSearch Dashboards

Dev Tools

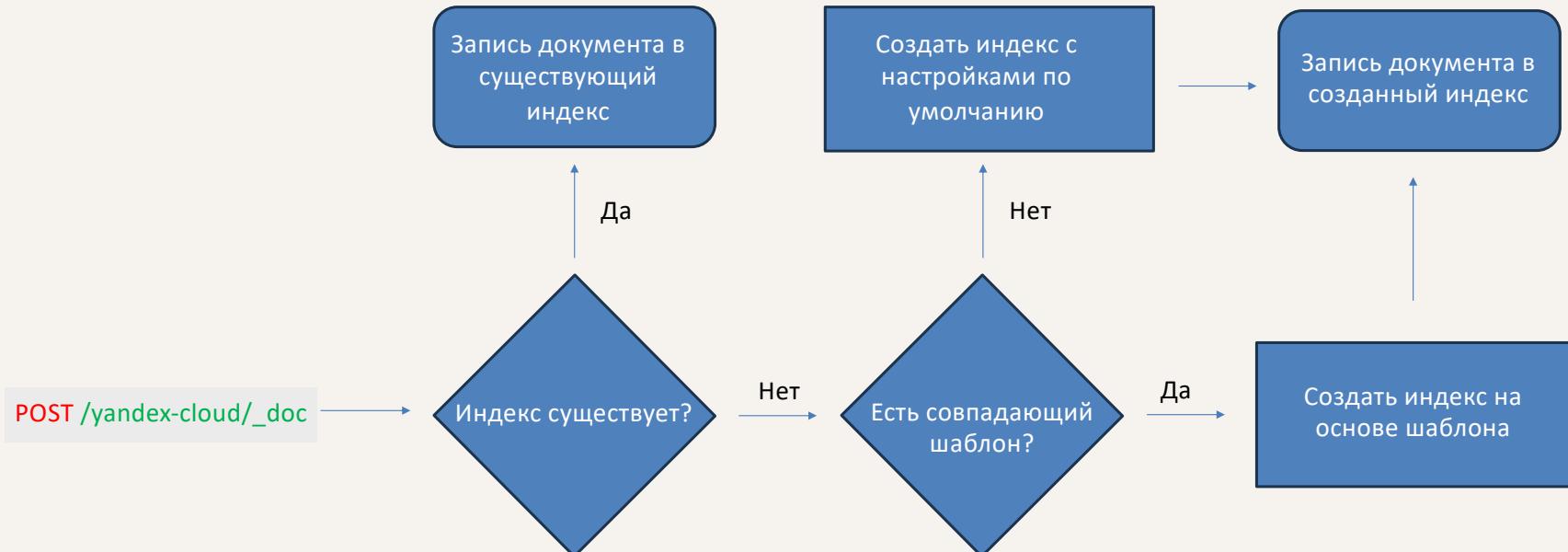
Console

History Settings Help Export Import 200 - OK 811 ms

```
1 GET yandex-cloud/_stats| 1  {
2 2   "_shards": { 2   "total": 2, 3   "successful": 2, 4   "failed": 0 5  },
6   "_all": { 6   "primaries": { 7   "docs": { 8   "count": 1, 9   "deleted": 0 10  },
11   "store": { 12   "size_in_bytes": 4457, 13   "reserved_in_bytes": 0 14  },
15   "indexing": { 16   "index_total": 1, 17   "index_time_in_millis": 0, 18   "index_current": 0, 19   "index_failed": 0, 20   "delete_total": 1, 21   "delete_time_in_millis": 6, 22   "delete_current": 0, 23   "noop_update_total": 0, 24   "is_throttled": false, 25   "throttle_time_in_millis": 0, 26 27 }
```

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



Screenshot of the GitHub repository for the OpenSearch Benchmark Workloads. The repository is named `opensearch-project/opensearch-benchmark-workloads` and is public. The `eventdata` branch is selected. The repository has 48 issues, 1 pull request, and 29 stars. The README.md file is visible, detailing the `EventData workload`. It states that the workload is based on 20 million Apache access log entries and is around 15GB in size. The repository contains several files and subdirectories, including `main`, `.github`, `big5`, `clickbench`, `common_operations`, `eventdata` (which is expanded to show `operations`, `test_procedures`, `README.md`, `files.txt`, `index.json`, `workload.json`), `geonames`, `geopoint`, `geopointshape`, `geoshape`, `http_logs`, `nested`, `neural_search`, `noaa`, and `noaa_semantic_search`.

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

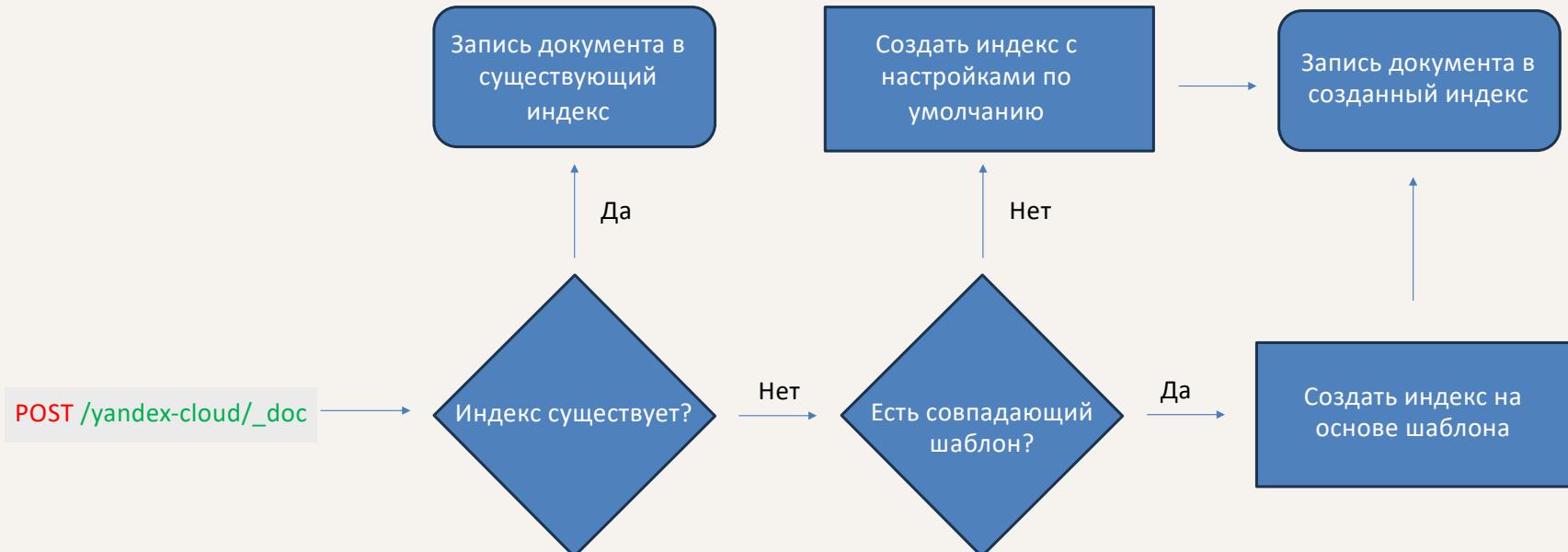
# Датасет eventlog

**Позже мы его загрузим в кластер и посмотрим...**

# **КАК ОПТИМИЗИРОВАТЬ ХРАНЕНИЕ**

# Работать с маппингом

# Как создается индекс



# Динамический маппинг полей (dynamic)

JSON	OPENSEARCH
string	<ul style="list-style-type: none"><li>• text + keyword</li><li>• date</li><li>• float или long</li></ul>
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**

**short**

**keyword**

**double**

**text**

**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": "мяких",  
      "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	

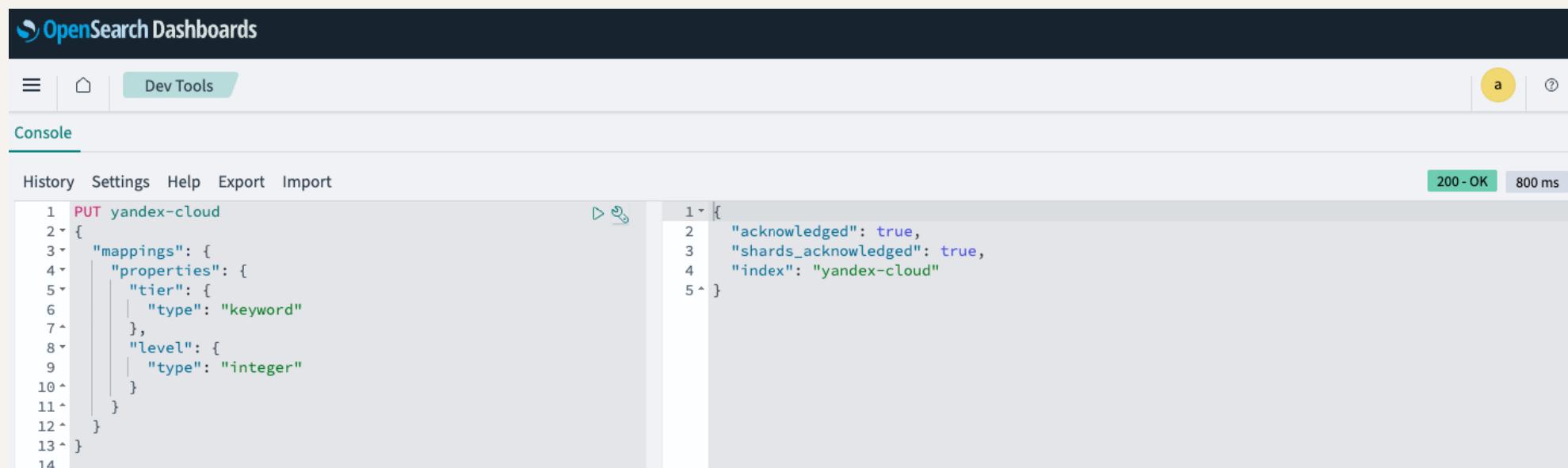
# Анализатор (keyword)

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



OpenSearch Dashboards

Dev Tools

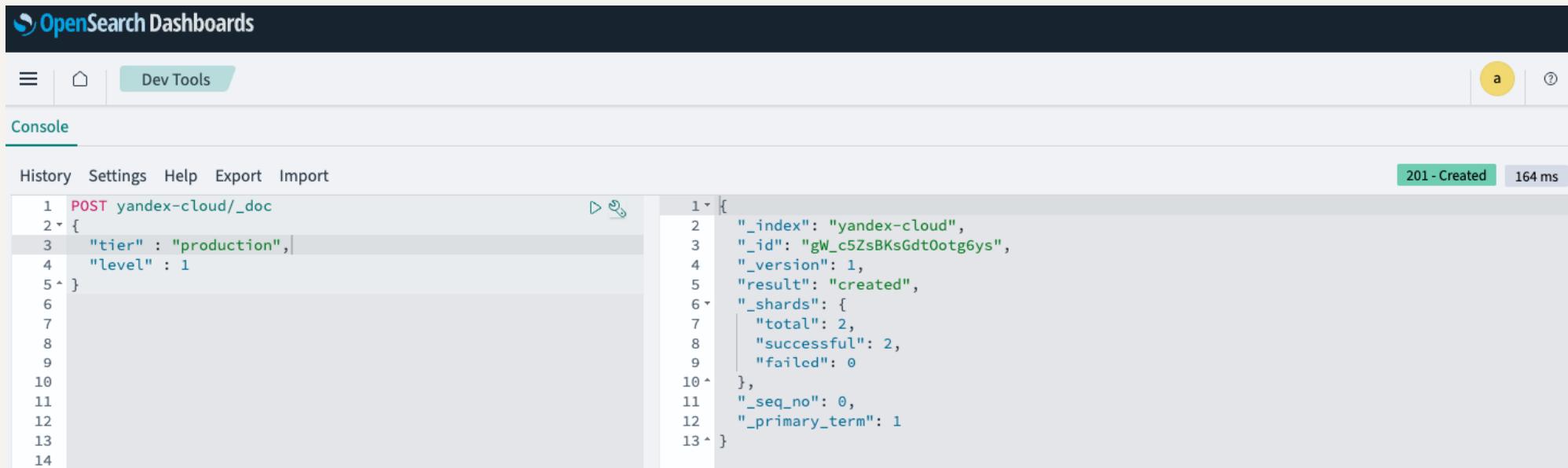
Console

History Settings Help Export Import 200 - OK 800 ms

```
1 PUT yandex-cloud
2 {
3   "mappings": {
4     "properties": {
5       "tier": {
6         "type": "keyword"
7       },
8       "level": {
9         "type": "integer"
10      }
11    }
12  }
13 }
14 }
```

1 {  
2 "acknowledged": true,  
3 "shards\_acknowledged": true,  
4 "index": "yandex-cloud"  
5 }

# Записываем документ



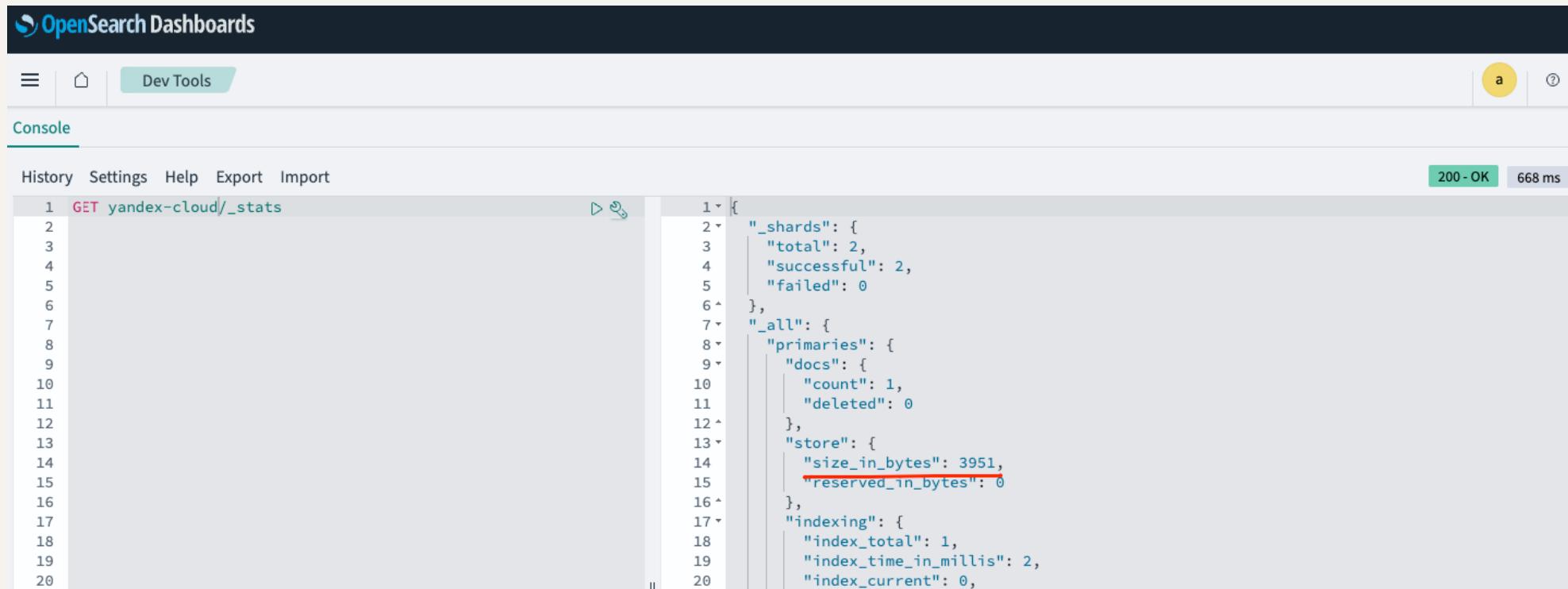
The screenshot shows the OpenSearch Dashboards Dev Tools Console. The URL bar at the top says "OpenSearch Dashboards". The main area is titled "Console". The "History" tab is selected. The "Dev Tools" button is highlighted in green. The "201 - Created" status and "164 ms" latency are shown on the right. The console shows a POST request to "yandex-cloud/\_doc" with the following JSON payload:

```
1 POST yandex-cloud/_doc
2 {
3   "tier" : "production",
4   "level" : 1
5 }
```

The response is a JSON object:

```
1 [
2   "_index": "yandex-cloud",
3   "_id": "gW_c5ZsBKsGdt0otg6ys",
4   "_version": 1,
5   "result": "created",
6   "_shards": {
7     "total": 2,
8     "successful": 2,
9     "failed": 0
10  },
11  "_seq_no": 0,
12  "_primary_term": 1
13 }
```

# Получаем результат



OpenSearch Dashboards

Dev Tools

Console

History Settings Help Export Import 200 - OK 668 ms

```
1 GET yandex-cloud/_stats
```

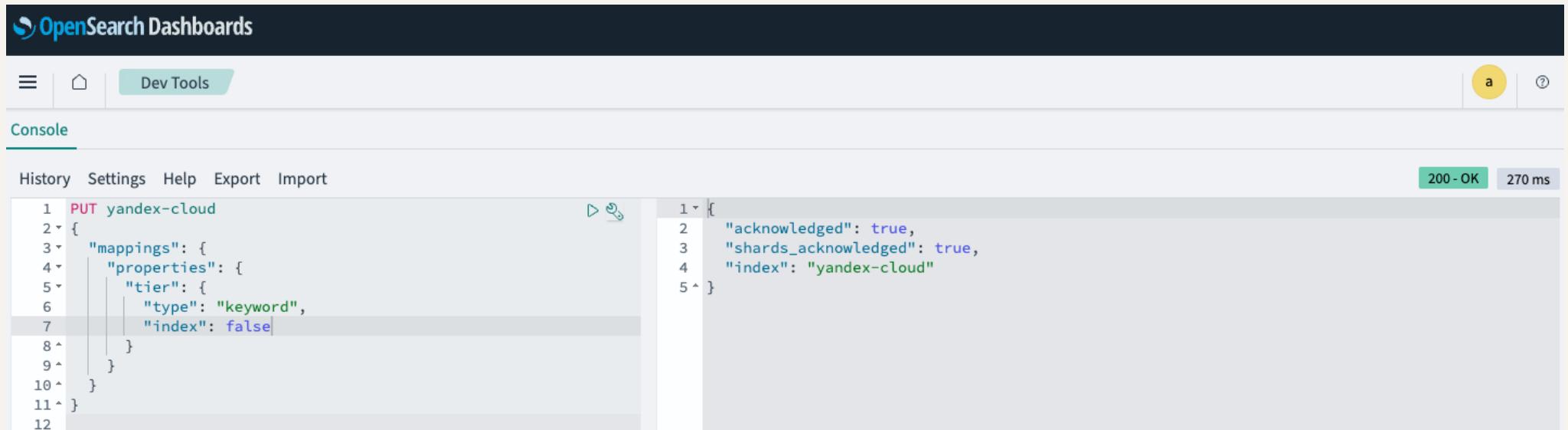
```
1 [ {  
2   "_shards": {  
3     "total": 2,  
4     "successful": 2,  
5     "failed": 0  
6   },  
7   "_all": {  
8     "primaries": {  
9       "docs": {  
10        "count": 1,  
11        "deleted": 0  
12      },  
13      "store": {  
14        "size_in_bytes": 3951,  
15        "reserved_in_bytes": 0  
16      },  
17      "indexing": {  
18        "index_total": 1,  
19        "index_time_in_millis": 2,  
20        "index_current": 0,  
21      }  
22    }  
23  }  
24 }  
25 }
```

\* 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



The screenshot shows the OpenSearch Dashboards Dev Tools Console. The URL bar says "OpenSearch Dashboards". The top navigation bar includes "Dev Tools", "Console" (which is underlined), "History", "Settings", "Help", "Export", and "Import". The status bar on the right shows "200 - OK" and "270 ms". The main area is a code editor with line numbers 1 to 12. Line 1: "PUT yandex-cloud". Lines 2-11 define an index mapping for a "tier" property: "mappings": {"properties": {"tier": {"type": "keyword", "index": false}}}. Line 12: "}" (closing brace). To the right of the code editor is the response JSON: {"acknowledged": true, "shards\_acknowledged": true, "index": "yandex-cloud"}. The "index": "yandex-cloud" part is highlighted in blue.

```
1 PUT yandex-cloud
2 {
3   "mappings": {
4     "properties": {
5       "tier": {
6         "type": "keyword",
7         "index": false
8       }
9     }
10  }
11 }
12
```

```
1 {
2   "acknowledged": true,
3   "shards_acknowledged": true,
4   "index": "yandex-cloud"
5 }
```

\* отключение записи в инвертированный индекс

# Смотрим размер

OpenSearch Dashboards

☰ ⌂ Dev Tools ⌂ a ⓘ

Console

History Settings Help Export Import 200 - OK 691 ms

```
1 GET yandex-cloud/_stats| ↗ a ⓘ
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
```

```
1 {
2   "_shards": {
3     "total": 2,
4     "successful": 2,
5     "failed": 0
6   },
7   "_all": {
8     "primaries": {
9       "docs": {
10         "count": 1,
11         "deleted": 0
12     },
13     "store": {
14       "size_in_bytes": 3733,
15       "reserved_in_bytes": 0
16     },
17     "indexing": {
18       "index_total": 1,
19       "index_time_in_millis": 1,
20       "index_current": 0,
21       "index_failed": 0,
22       "delete_total": 0,
23       "delete_time_in_millis": 0,
24       "delete_current": 0,
25       "noop_update_total": 0,
26       "is_throttled": false,
```

# 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": "\u2022",
  "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
<input type="checkbox"/>	Index 	Health	Managed by policy	Status	Total size	Size of primaries	Total documents	Deleted documents	Primaries	Replicas	<input type="checkbox"/> Show data stream indexes	
<input type="checkbox"/>	 eventdata-compressed-6	 Green	No	Open	3gb	3gb	20000000	0	5	0		
<input type="checkbox"/>	 eventdata	 Green	No	Open	4.9gb	4.9gb	20000000	0	5	0		
Rows per page: 20 										 1 		

# Смотрим на маппинг

OpenSearch Dashboards

Dev Tools

Console

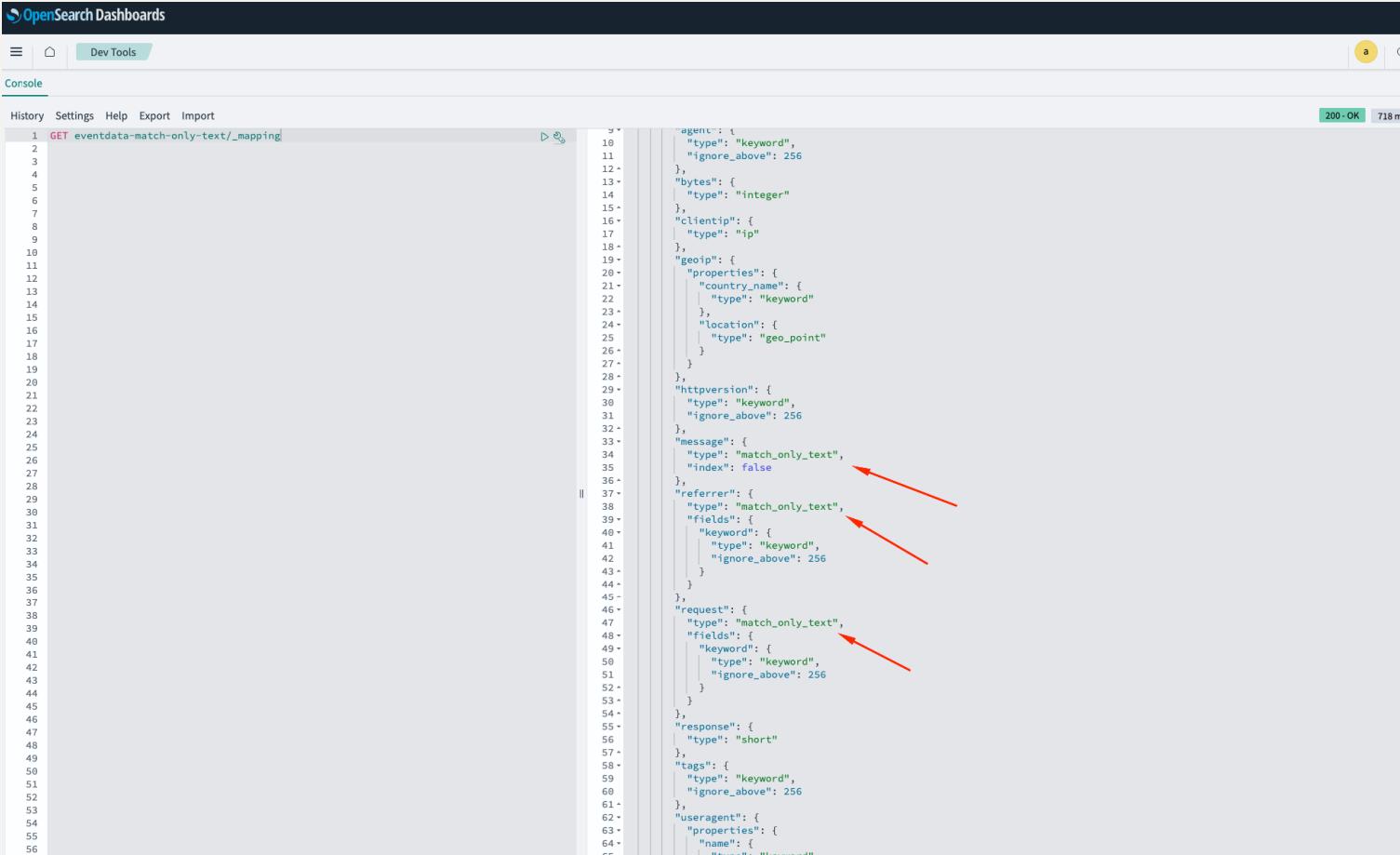
History Settings Help Export Import

1 GET eventdata-compressed-6/\_mapping

```
1 GET eventdata-compressed-6/_mapping
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
"index": false
"agent": {
  "type": "keyword",
  "ignore_above": 256
},
"bytes": {
  "type": "integer"
},
"clientip": {
  "type": "ip"
},
"geoip": {
  "properties": {
    "country_name": {
      "type": "keyword"
    },
    "location": {
      "type": "geo_point"
    }
  }
},
"httpversion": {
  "type": "keyword",
  "ignore_above": 256
},
"message": {
  "type": "text",
  "index": false
},
"referrer": {
  "type": "text",
  "fields": {
    "keyword": {
      "type": "keyword",
      "ignore_above": 256
    }
  }
},
"norms": false
},
"request": {
  "type": "text",
  "fields": {
    "keyword": {
      "type": "keyword",
      "ignore_above": 256
    }
  }
},
"norms": false
},
"response": {
  "type": "short"
},
"tags": {
  "type": "keyword",
  "ignore_above": 256
}
```

200 - OK 867 ms

# Вспоминаем про match\_only\_text



```
1 GET eventdata-match-only-text/_mapping
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
```

```
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
```

```
  "agent": {
    "type": "keyword",
    "ignore_above": 256
  },
  "bytes": {
    "type": "integer"
  },
  "clientip": {
    "type": "ip"
  },
  "geoip": {
    "properties": {
      "country_name": {
        "type": "keyword"
      },
      "location": {
        "type": "geo_point"
      }
    }
  },
  "httpversion": {
    "type": "keyword",
    "ignore_above": 256
  },
  "message": {
    "type": "match_only_text",
    "index": false
  },
  "referrer": {
    "type": "match_only_text",
    "fields": [
      "keyword": {
        "type": "keyword",
        "ignore_above": 256
      }
    ]
  },
  "request": {
    "type": "match_only_text",
    "fields": [
      "keyword": {
        "type": "keyword",
        "ignore_above": 256
      }
    ]
  },
  "response": {
    "type": "short"
  },
  "tags": {
    "type": "keyword",
    "ignore_above": 256
  },
  "useragent": {
    "properties": {
      "name": {
        "type": "keyword"
      }
    }
  }
}
```

# Загрузим eventdata снова

## text vs match\_only\_text

Indexes (3)										 Refresh	 Actions	 + Create Index
 Index	↓	Health	Managed by policy	Status	Total size	Size of primaries	Total documents	Deleted documents	Primaries	Replicas	 Show data stream indexes	
  eventdata-match-only-text		 Green	No	Open	2.9gb	2.9gb	20000000	0	5	0		
  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 		

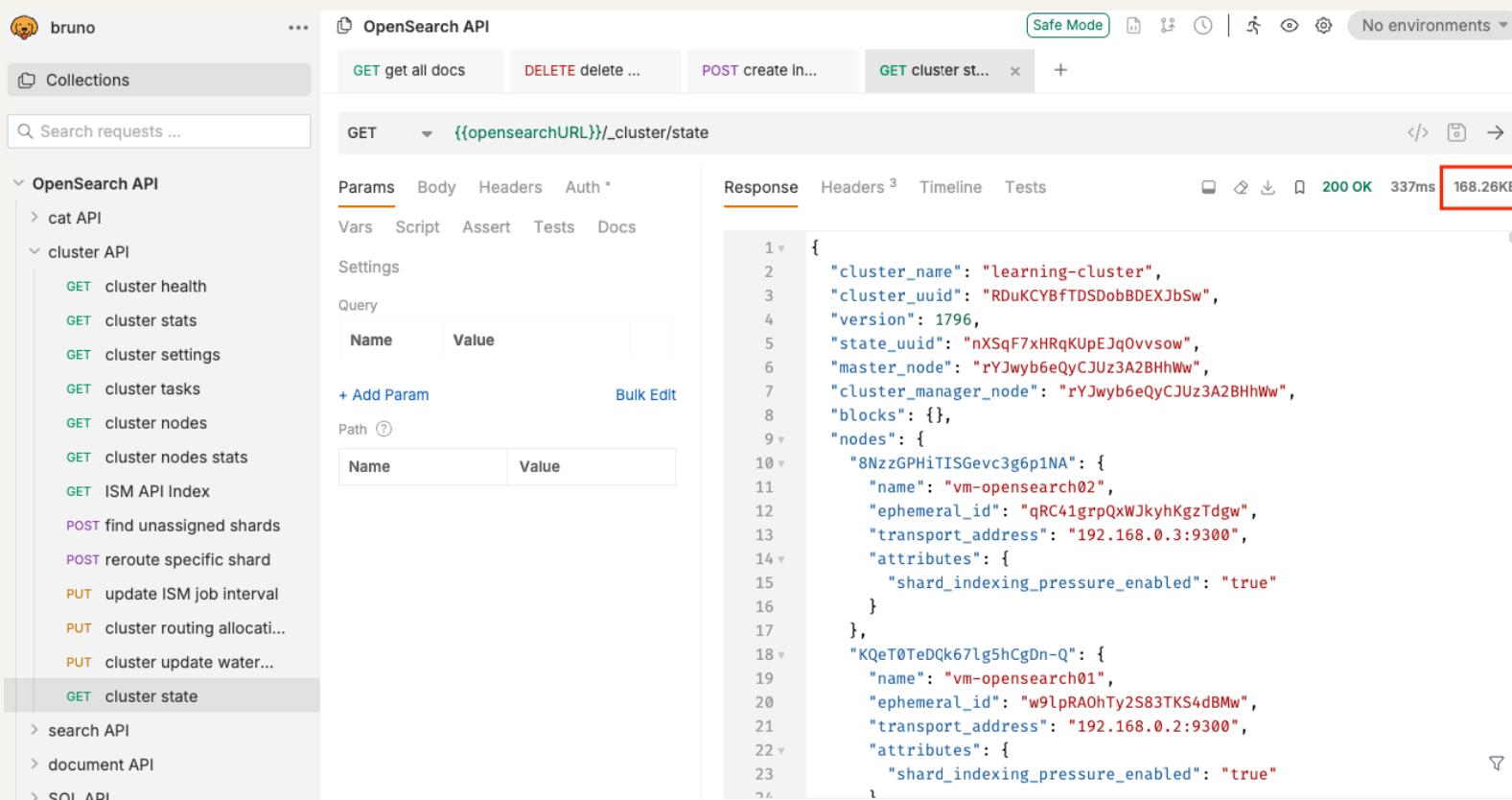
# Работать с `cluster_state`

# 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`



The screenshot shows the OpenSearch API interface with the following details:

- Header:** Bruno
- Collection:** Collections
- Search:** Search requests ...
- API:** OpenSearch API
- Endpoint:** GET `{{opensearchURL}}/_cluster/state`
- Params:** Params tab selected. Fields: Vars, Script, Assert, Tests, Docs.
- Response:** Headers: 3, Timeline, Tests. Status: 200 OK, 337ms, 168.26KB. The response body is a JSON object representing the cluster state, showing two nodes: `8NzZGPHiTISGevc3g6p1NA` and `KQeT0TeDQk67lg5hCgDn-Q`, each with their respective names, ephemeral IDs, transport addresses, and attributes including `shard_indexing_pressure_enabled: true`.

Если  OpenSearch, то 

Делаем как для себя.

# **gals.software**

