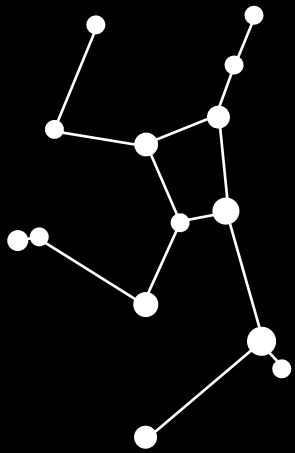


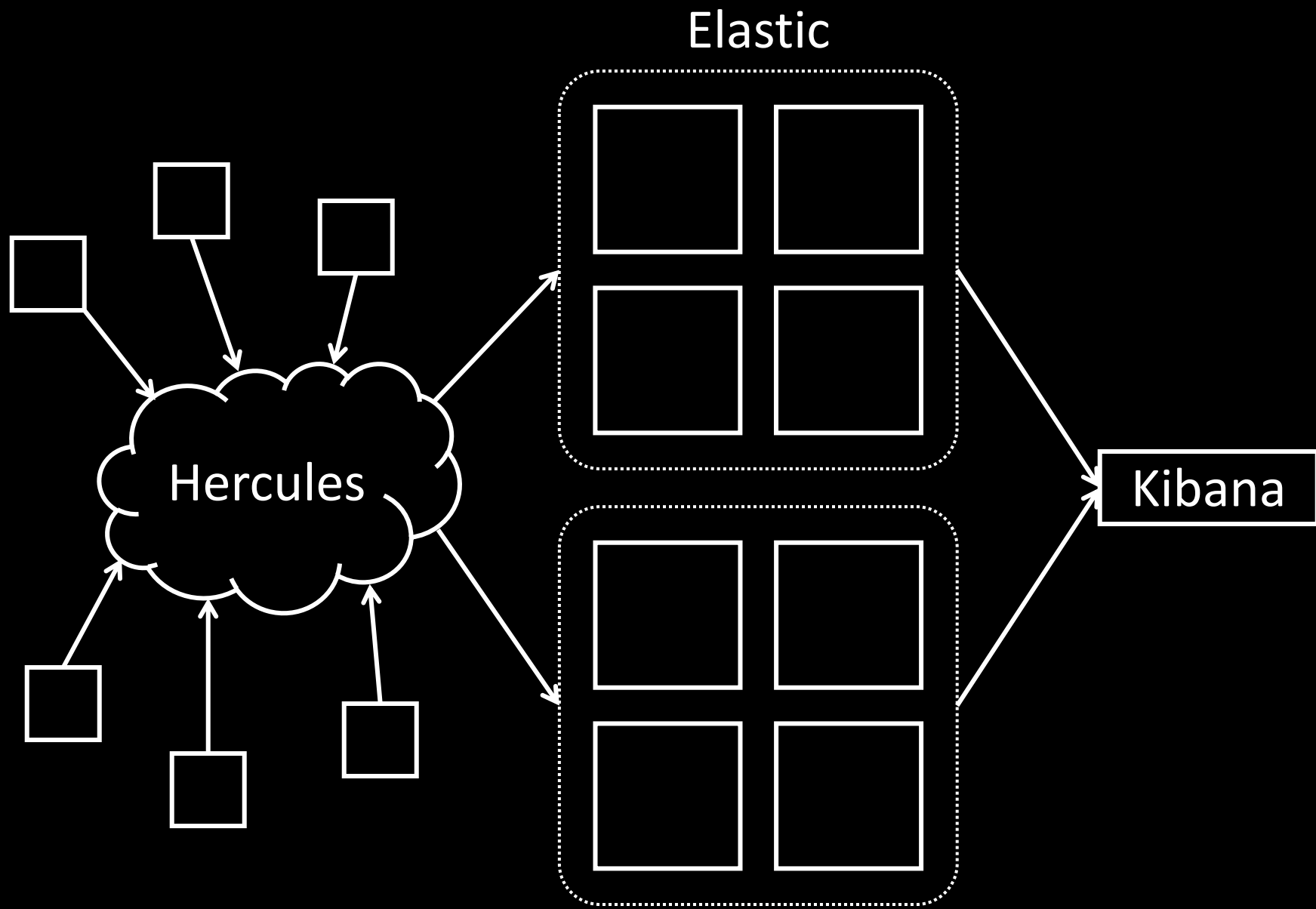
**НЕЛЬЗЯ ПРОСТО ТАК ВЗЯТЬ**

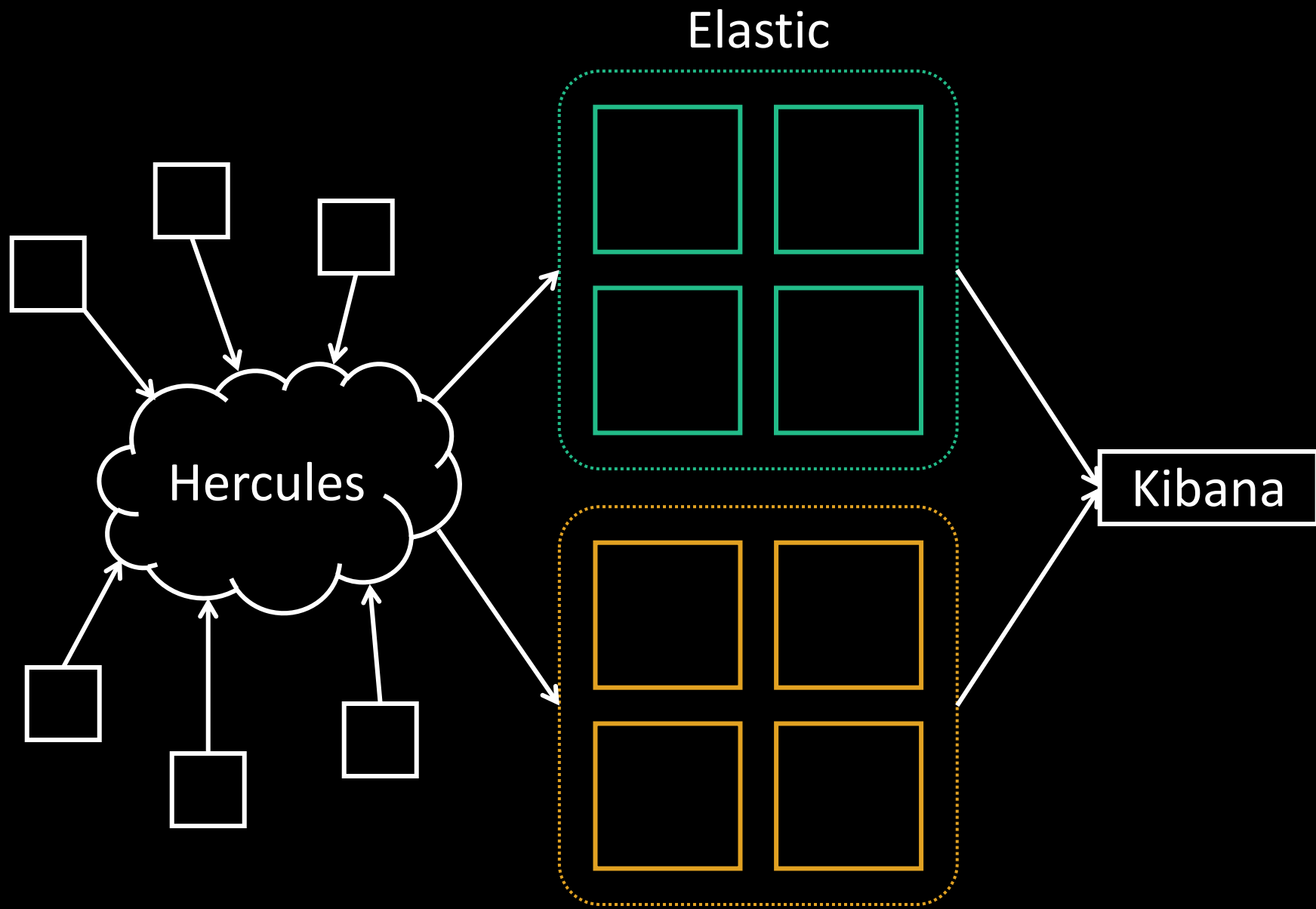
**И ОТПРАВИТЬ ВСЕ ЛОГИ В ELASTIC**



Григорий Кошелёв

Контур





# Hercules – транспорт для телеметрии

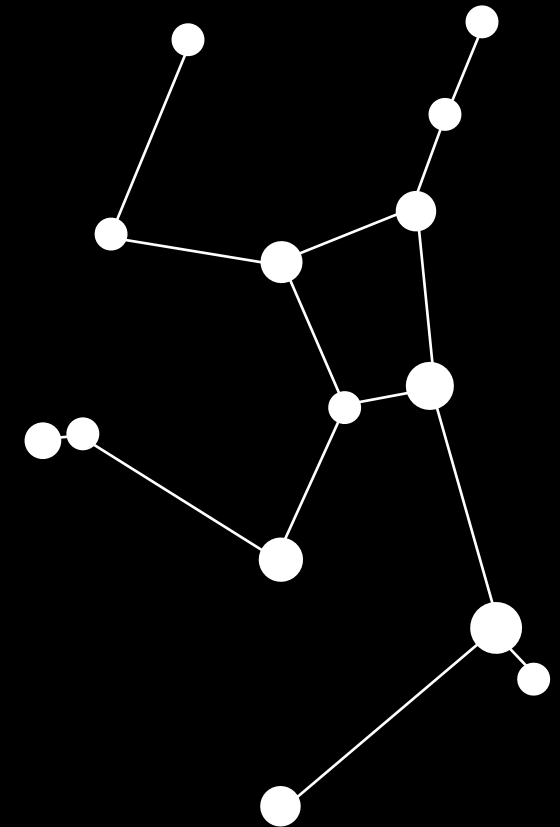
<https://github.com/vostok/hercules>

<https://youtu.be/qg3yRmV-fHs>

Vostok Hercules:

3 года доставляем телеметрию —  
полёт нормальный

(JPoint 2022)



<https://github.com/vostok/hercules>

Обычный Elastic vs Elastic для логов

# Обычный Elastic vs Elastic для логов

- Данные однородны в одном индексе

- Данные разнородны в одном индексе

# Обычный Elastic vs Elastic для логов

- Данные однородны в одном индексе
- Данные редко меняются

- Данные разнородны в одном индексе
- Данные иммутабельны



# Обычный Elastic vs Elastic для логов

- Данные однородны в одном индексе
- Данные редко меняются
- Низкая частота записи
- Данные разнородны в одном индексе
- Данные иммутабельны
- Высокая частота записи

# Обычный Elastic vs Elastic для логов

- Данные однородны в одном индексе
  - Данные редко меняются
  - Низкая частота записи
  - Сильный разброс в размере документов
- Данные разнородны в одном индексе
  - Данные иммутабельны
  - Высокая частота записи
  - Большинство документов небольшого размера

# Логи в Контуре

- 3 кластера Elastic для логов
- 250 000 событий в секунду
- более 100 команд разработки

# Логи в Контуре

<https://youtu.be/KN4Ia0uir8Y>

Эластик весом в петабайт

(Владимир Лила, DUMP 2019)

<https://youtu.be/KN4Ia0uir8Y>

# Как разделять Логи

# Как разделять Логи

— Дробление индекса по дате

*<префикс> - <YYYY.MM.DD>*

# Как разделять Логи

- Дробление индекса по дате

*<префикс>* - *<YYYY.MM.DD>*

- Index Lifecycle Management (с версии **6.7**)

*<префикс>* - *<00001>*

*<префикс>* - *<00002>*

...

Как разделять Логи



# Как разделять Логи

— Проект / команда

# Как разделять Логи

- Проект / команда
- Окружение (staging, production, ...)

# Как разделять Логи

- Проект / команда
- Окружение (staging, production, ...)

*<проект> - <окружение> - <00XXX>*

# Как разделять Логи

- Проект / команда
- Окружение (staging, production, ...)

*<проект> - <окружение> - <00XXX>*

*<проект> - <окружение> - <сервис> - <00XXX>*

# Как устроены Логи

```
{  
  "@timestamp" : "2022-11-19T08:00:00.123456789Z",  
  "loggerName" : "joker.conf.ElasticTalk",  
  "level" : "INFO",  
  "message" : "Утро начинается с кофе",  
  "user" : "Gregory",  
  "timeoutMs" : 120000,  
}
```

# Как устроены Логи

```
{  
  "@timestamp" : "2022-11-19T08:00:00.123456789Z",  
  "loggerName" : "joker.conf.ElasticTalk",  
  "level" : "INFO",  
  "message" : "Утро начинается с кофе",  
  "user" : "Gregory",  
  "timeoutMs" : 120000,  
}
```

# Как устроены Логи

```
{  
  "@timestamp" : "2022-11-19T08:00:00.123456789Z",  
  "loggerName" : "joker.conf.ElasticTalk",  
  "level" : "INFO",  
  "message" : "Утро начинается с кофе",  
  "user" : "Gregory",  
  "timeoutMs" : 120000,  
}
```

# Как устроены Логи

```
{  
  "@timestamp" : "2022-11-19T08:00:00.123456789Z",  
  "loggerName" : "joker.conf.ElasticTalk",  
  "level" : "INFO",  
  "message" : "Утро начинается с кофе",  
  "user" : "Gregory",  
  "timeoutMs" : 120000,  
}
```



Конвенция полей в логах

# Конвенция полей в логах

Согласованность названий и типов значений  
в пределах одного проекта

# Конвенция полей в логах

Согласованность названий и типов значений  
в пределах одного проекта

+ общие библиотеки

# Конвенция полей в логах

Согласованность названий и типов значений  
в пределах одного проекта

+ общие библиотеки

```
index.mapping.total_fields.limit: 1000
```

количество полей в индексе

<https://www.elastic.co/guide/en/elasticsearch/reference/current/mapping-settings-limit.html>

# Особенности работы с API

# Особенности работы с API

Native Java Client: TransportClient

<https://www.elastic.co/guide/en/elasticsearch/client/java-api/7.17/transport-client.html>

# Особенности работы с API

Native Java Client: TransportClient

— Высокоуровневые абстракции (Request, Response)

# Особенности работы с API

## Native Java Client: TransportClient

- Высокоуровневые абстракции (Request, Response)
- Отсутствует обратная совместимость между версиями



# Особенности работы с API

## Native Java Client: TransportClient

- Высокоуровневые абстракции (Request, Response)
- Отсутствует обратная совместимость между версиями
- Deprecated с версии Elasticsearch **7.0**

# Особенности работы с API

## Native Java Client: TransportClient

- Высокоуровневые абстракции (Request, Response)
- Отсутствует обратная совместимость между версиями
- Deprecated с версии Elasticsearch **7.0**
- Удалён в версии **8.0**

<https://www.elastic.co/guide/en/elasticsearch/client/java-api/7.17/transport-client.html>

# Особенности работы с API

HTTP REST API

<https://www.elastic.co/guide/en/elasticsearch/client/java-rest/current/index.html>

# Особенности работы с API

## HTTP REST API

– `elasticsearch-rest-client` (с версии **5.0**)

<https://www.elastic.co/guide/en/elasticsearch/client/java-rest/current/index.html>

# Особенности работы с API

## HTTP REST API

- `elasticsearch-rest-client` (с версии **5.0**)
- `elasticsearch-rest-high-level-client` (с версии **5.6** и **6.0**)

<https://www.elastic.co/guide/en/elasticsearch/client/java-rest/current/index.html>

# Особенности работы с API

## HTTP REST API

- `elasticsearch-rest-client` (с версии **5.0**)
- `elasticsearch-rest-high-level-client` (с версии **5.6** и **6.0**)

## Под капотом Apache HTTP Async Client

<https://www.elastic.co/guide/en/elasticsearch/client/java-rest/current/index.html>

# Особенности работы с API

## HTTP REST API

- `elasticsearch-rest-client` (с версии **5.0**)
- `elasticsearch-rest-high-level-client` (с версии **5.6** и **6.0**)

Под капотом Apache HTTP Async Client  
(синхронная и асинхронная работа с API)

<https://www.elastic.co/guide/en/elasticsearch/client/java-rest/current/index.html>

# Elasticsearch (Java) REST Client

Текущая версия клиента: **8.5**



# Elasticsearch (Java) REST Client

- Совместим с любой версией Elasticsearch

# Elasticsearch (Java) REST Client

- Совместим с любой версией Elasticsearch
- Балансировка по всем доступным нодам

# Elasticsearch (Java) REST Client

- Совместим с любой версией Elasticsearch
- Балансировка по всем доступным нодам
- Фэйловер (в некоторых случаях)

# Elasticsearch (Java) REST Client

- Совместим с любой версией Elasticsearch
- Балансировка по всем доступным нодам
- Фэйловер (в некоторых случаях)
- Постоянные коннекции к кластеру

# Elasticsearch (Java) REST Client

- Совместим с любой версией Elasticsearch
- Балансировка по всем доступным нодам
- Фэйловер (в некоторых случаях)
- Постоянные коннекции к кластеру
- Актуализация топологии кластера \*

# Elasticsearch (Java) REST Client

- Совместим с любой версией Elasticsearch
- Балансировка по всем доступным нодам
- Фэйловер (в некоторых случаях)
- Постоянные коннекции к кластеру
- Актуализация топологии кластера \*  
(ещё одна библиотека)

# Elasticsearch (Java) REST Client

```
HttpHost[] hosts = ... // elastic data nodes  
RestClient restClient = RestClient.builder(hosts).build();
```

# Elasticsearch (Java) REST Client

```
HttpHost[] hosts = ... // elastic data nodes
RestClient restClient = RestClient.builder(hosts).build();

Request request = new Request("POST", "_bulk");
request.setEntity(new ByteArrayEntity(data, ContentType.APPLICATION_JSON));
```



# Elasticsearch (Java) REST Client

```
HttpHost[] hosts = ... // elastic data nodes
RestClient restClient = RestClient.builder(hosts).build();

Request request = new Request("POST", "_bulk");
request.setEntity(new ByteArrayEntity(data, ContentType.APPLICATION_JSON));

Response response = restClient.performRequest(request);
```

# Elasticsearch (Java) REST Client

- Совместим с любой версией Elasticsearch
- Балансировка по всем доступным нодам
- Фэйловер (в некоторых случаях)
- Постоянные коннекции к кластеру
- Актуализация топологии кластера \*  
(ещё одна библиотека)

# Elasticsearch (Java) REST Client

```
Request request = new Request("POST", "/_bulk");
request.setEntity(new ByteArrayEntity(data, ContentType.APPLICATION_JSON));

Request request = new Request("POST", "/my_index/_doc/");
request.setEntity(new ByteArrayEntity(data, ContentType.APPLICATION_JSON));

Response response = restClient.performRequest(request);
```

# Elasticsearch (Java) REST Client

```
Request request = new Request("POST", "/_bulk");
request.setEntity(new ByteArrayEntity(data, ContentType.APPLICATION_JSON));

Request request = new Request("POST", "/my_index/_doc/");
request.setEntity(new ByteArrayEntity(data, ContentType.APPLICATION_JSON));

Response response = restClient.performRequest(request);
```

# Elasticsearch (Java) REST Client

Выводы

# Elasticsearch (Java) REST Client

## Выводы

- Можно долго не менять версию клиента

# Elasticsearch (Java) REST Client

## Выводы

- Можно долго не менять версию клиента
- Ответственность за **version specific** ложится на наш код

# Elasticsearch (Java) REST Client

## Выводы

- Можно долго не менять версию клиента
- Ответственность за **version specific** ложится на наш код (или **High Level REST client**)



# Elasticsearch (Java) REST Client

- Совместим с любой версией Elasticsearch
- **Балансировка по всем доступным нодам**
- Фэйловер (в некоторых случаях)
- Постоянные коннекции к кластеру
- Актуализация топологии кластера \*  
(ещё одна библиотека)

# Elasticsearch (Java) REST Client

# Elasticsearch (Java) REST Client

- Хост для каждого запроса выбирается по **round robin**

# Elasticsearch (Java) REST Client

- Хост для каждого запроса выбирается по **round robin**
- Если запрос завершился ошибкой (код  $\geq 300$ ), то хост попадает в **blacklist** на **1 минуту**

# Elasticsearch (Java) REST Client

- Хост для каждого запроса выбирается по **round robin**
- Если запрос завершился ошибкой (код  $\geq 300$ ), то хост попадает в **blacklist** на **1 минуту**
- Каждый следующий запрос с ошибкой увеличивает время нахождения в **blacklist** в  $\sqrt{2}$  раз (макс. **30 минут**)

# Elasticsearch (Java) REST Client

- Хост для каждого запроса выбирается по **round robin**
- Если запрос завершился ошибкой (код  $\geq 300$ ), то хост попадает в **blacklist** на **1 минуту**
- Каждый следующий запрос с ошибкой увеличивает время нахождения в **blacklist** в  $\sqrt{2}$  раз (макс. **30 минут**)
- Если все хосты попали в **blacklist** – достаётся ближайший по времени

# Elasticsearch (Java) REST Client

Выводы

# Elasticsearch (Java) REST Client

## Выводы

- Первая поднявшаяся нода Эластика будет страдать



# Elasticsearch (Java) REST Client

## Выводы

- Первая поднявшаяся нода Эластика будет страдать
- Балансировка нагрузки долго приходит в норму после шторма (или работ) в кластере

# Elasticsearch (Java) REST Client

- Совместим с любой версией Elasticsearch
- Балансировка по всем доступным нодам
- Фэйловер (в некоторых случаях)
- Постоянные коннекции к кластеру
- Актуализация топологии кластера \*  
(ещё одна библиотека)

# Elasticsearch (Java) REST Client

Фэйловер

# Elasticsearch (Java) REST Client

Фэйловер

— если код 502, 503 или 504

# Elasticsearch (Java) REST Client

## Фэйловер

- если код 502, 503 или 504
- если доступных хостов > 1 (blacklist!)

# Elasticsearch (Java) REST Client

## Фэйловер

- если код 502, 503 или 504
- если доступных хостов > 1 (blacklist!)
- если осталось время (до версии 7.0)

# Elasticsearch (Java) REST Client

## Фэйловер

- если код **502, 503** или **504**
- если доступных хостов **> 1** (**blacklist!**)
- если осталось время (до версии **7.0**)

```
RestClient restClient = RestClient.builder(hosts)
    .setMaxRetryTimeoutMillis(90_000)
    .build();
```

# Elasticsearch (Java) REST Client

## Фэйловер

- если код 502, 503 или 504
- если доступных хостов > 1 (blacklist!)
- ~~— если осталось время (до версии 7.0)~~

```
RestClient restClient = RestClient.builder(hosts)
    .setMaxRetryTimeoutMillis(90_000)
    .build();
```



# Elasticsearch (Java) REST Client

Выводы

# Elasticsearch (Java) REST Client

## Выводы

- Поведение зависит от версии клиента

# Elasticsearch (Java) REST Client

## Выводы

- Поведение зависит от версии клиента
- Реализация полагается на настройки Apache HTTP Async Client

# Elasticsearch (Java) REST Client

- Совместим с любой версией Elasticsearch
- Балансировка по всем доступным нодам
- Фэйловер (в некоторых случаях)
- **Постоянные коннекции к кластеру**
- Актуализация топологии кластера \*  
(ещё одна библиотека)

# Elasticsearch (Java) REST Client

ApacheHttpClient же!

# Elasticsearch (Java) REST Client

ApacheHttpClient же!

– Keep-Alive (HTTP 1.1)

# Elasticsearch (Java) REST Client

ApacheHttpClient жє!

- Keep-Alive (HTTP 1.1)
- PoolingNHttpClientConnectionManager

# Elasticsearch (Java) REST Client

- Совместим с любой версией Elasticsearch
- Балансировка по всем доступным нодам
- Фэйловер (в некоторых случаях)
- Постоянные коннекции к кластеру
- Актуализация топологии кластера \*  
(ещё одна библиотека)



# Elasticsearch (Java) REST Client

```
RestClient restClient = RestClient.builder(hosts).build();
```

# Elasticsearch (Java) REST Client

```
RestClient restClient = RestClient.builder(hosts).build();
```

```
Sniffer sniffer = Sniffer.builder(restClient)  
    .setSniffIntervalMillis(5 * 60 * 1000) // 5 минут  
    .build();
```

# Elasticsearch (Java) REST Client

```
RestClient restClient = RestClient.builder(hosts).build();
```

```
Sniffer sniffer = Sniffer.builder(restClient)  
    .setSniffIntervalMillis(5 * 60 * 1000) // 5 минут  
    .build();
```



# Elasticsearch (Java) REST Client

```
RestClient restClient = RestClient.builder(hosts).build();
```

```
Sniffer sniffer = Sniffer.builder(restClient)  
    .setSniffIntervalMillis(5 * 60 * 1000) // 5 минут  
    .build();
```

↓

```
Request request = new Request("GET", "_nodes/http");
```

# Elasticsearch (Java) REST Client

```
RestClient restClient = RestClient.builder(hosts).build();
```

```
Sniffer sniffer = Sniffer.builder(restClient)  
    .setSniffIntervalMillis(5 * 60 * 1000) // 5 минут  
    .build();
```

↓

```
Request request = new Request("GET", "_nodes/http");  
Response response = restClient.performRequest(request);
```

# Elasticsearch (Java) REST Client

```
RestClient restClient = RestClient.builder(hosts).build();
```

```
Sniffer sniffer = Sniffer.builder(restClient)  
    .setSniffIntervalMillis(5 * 60 * 1000) // 5 минут  
    .build();
```

↓

```
Request request = new Request("GET", "_nodes/http");  
Response response = restClient.performRequest(request);  
List<Node> sniffedNodes = readHosts(response.getEntity, ...);
```

# Elasticsearch (Java) REST Client

```
RestClient restClient = RestClient.builder(hosts).build();
```

```
Sniffer sniffer = Sniffer.builder(restClient)
    .setSniffIntervalMillis(5 * 60 * 1000) // 5 минут
    .build();
```

↓

```
Request request = new Request("GET", "/_nodes/http");
Response response = restClient.performRequest(request);
List<Node> sniffedNodes = readHosts(response.getEntity, ...);
if (!sniffedNodes.isEmpty()) {
    restClient.setNodes(sniffedNodes);
}
```

# Elasticsearch (Java) REST Client

```
RestClient restClient = RestClient.builder(hosts).build();
```

```
Sniffer sniffer = Sniffer.builder(restClient)  
    .setSniffIntervalMillis(5 * 60 * 1000) // 5 минут  
    .build();
```

↓

```
Request request = new Request("GET", "/_nodes/http");  
Response response = restClient.performRequest(request);  
List<Node> sniffedNodes = readHosts(response.getEntity, ...);  
if (!sniffedNodes.isEmpty()) {  
    restClient.setNodes(sniffedNodes);  
}
```



# Elasticsearch (Java) REST Client

Выводы

# Elasticsearch (Java) REST Client

## Выводы

— `sniffer.close()` --> `restClient.close()`

# Elasticsearch (Java) REST Client

## Выводы

- `sniffer.close()` --> `restClient.close()`
- Sniffer очищает **blacklist**

# Elasticsearch (Java) REST Client

## Выводы

- `sniffer.close()` --> `restClient.close()`
- Sniffer очищает **blacklist**
- А что будет, если на запрос топологии ответит нода 1. Выведенная из кластера?

# Elasticsearch (Java) REST Client

## Выводы

- `sniffer.close()` --> `restClient.close()`
- Sniffer очищает **blacklist**
- А что будет, если на запрос топологии ответит нода
  1. Выведенная из кластера?
  2. Изолированная по сети?

# Elasticsearch (Java) REST Client

Сжатие контента при отправке в Elastic

# Elasticsearch (Java) REST Client

Сжатие контента при отправке в Elastic

— Поддерживается с версии **8.0**

# Elasticsearch (Java) REST Client

Сжатие контента при отправке в Elastic

- Поддерживается с версии **8.0**
- Портировано в **7.10+**



# Elasticsearch (Java) REST Client

Сжатие контента при отправке в Elastic

- Поддерживается с версии **8.0**
- Портировано в **7.10+**

Реализуется на уровне HTTP

# Apache HTTP Async Client

# Apache HTTP Async Client

```
RestClient.builder(hosts)
    .setHttpClientConfigCallback(httpClientBuilder -> httpClientBuilder
        .setMaxConnPerRoute(10)
        .setMaxConnTotal(30)
    )
    .setRequestConfigCallback(requestConfigBuilder -> requestConfigBuilder
        .setConnectTimeout(1_000)
        .setSocketTimeout(30_000)
    )
    .build();
```

# Apache HTTP Async Client

```
RestClient.builder(hosts)
    .setHttpClientConfigCallback(httpClientBuilder -> httpClientBuilder
        .setMaxConnPerRoute(10)
        .setMaxConnTotal(30)
    )
    .setRequestConfigCallback(requestConfigBuilder -> requestConfigBuilder
        .setConnectTimeout(1_000)
        .setSocketTimeout(30_000)
    )
    .build();
```

# Apache HTTP Async Client

```
RestClient.builder(hosts)
    .setHttpClientConfigCallback(httpClientBuilder -> httpClientBuilder
        .setMaxConnPerRoute(10)
        .setMaxConnTotal(30)
    )
    .setRequestConfigCallback(requestConfigBuilder -> requestConfigBuilder
        .setConnectTimeout(1_000)
        .setSocketTimeout(30_000)
    )
    .build();
```

# Apache HTTP Async Client

```
RestClient.builder(hosts)
    .setHttpClientConfigCallback(httpClientBuilder -> httpClientBuilder
        .setMaxConnPerRoute(10)
        .setMaxConnTotal(30)
    )
    .setRequestConfigCallback(requestConfigBuilder -> requestConfigBuilder
        .setConnectTimeout(1_000)
        .setSocketTimeout(30_000)
    )
    .build();
```

# Apache HTTP Async Client

```
RestClient.builder(hosts)
    .setHttpClientConfigCallback(httpClientBuilder -> httpClientBuilder
        .setMaxConnPerRoute(10)
        .setMaxConnTotal(30)
    )
    .setRequestConfigCallback(requestConfigBuilder -> requestConfigBuilder
        .setConnectTimeout(1_000)
        .setSocketTimeout(30_000)
        /* ??? */
    )
    .build();
```

# Apache HTTP Async Client

```
RestClient.builder(hosts)
    .setHttpClientConfigCallback(httpClientBuilder -> httpClientBuilder
        .setMaxConnPerRoute(10)
        .setMaxConnTotal(30)
    )
    .setRequestConfigCallback(requestConfigBuilder -> requestConfigBuilder
        .setConnectTimeout(1_000)
        .setSocketTimeout(30_000)
        .setConnectionRequestTimeout(500)
    )
    .build();
```



# Apache HTTP Async Client

```
RestClient.builder(hosts)
    .setHttpClientConfigCallback(httpClientBuilder -> httpClientBuilder
        .setMaxConnPerRoute(10)
        .setMaxConnTotal(30)
    )
    .setRequestConfigCallback(requestConfigBuilder -> requestConfigBuilder
        .setConnectTimeout(1_000)
        .setSocketTimeout(30_000)
        .setConnectionRequestTimeout(500) // До версии 6.3
    )
    .build();
```

# Elasticsearch REST Client

Выводы

# Elasticsearch REST Client

## Выводы

- Важно понимать, как работает `HttpClient` (и его настройки)

# Elasticsearch REST Client

## Выводы

- Важно понимать, как работает `HttpAsyncClient` (и его настройки)
- Настройки таймаутов на клиенте и сервере должны учитывать тяжёлые операции

# Elasticsearch REST Client

## Выводы

- Важно понимать, как работает `HttpAsyncClient` (и его настройки)
- Настройки таймаутов на клиенте и сервере должны учитывать тяжёлые операции
- Нельзя полагаться на поведение по умолчанию

# Elasticsearch REST High Level Client

# Elasticsearch REST High Level Client

- Высокоуровневые абстракции (Request, Response)

# Elasticsearch REST High Level Client

- Высокоуровневые абстракции (Request, Response)
- Замена для TransportClient



# Elasticsearch REST High Level Client

## high rest client has low performance #30385

✓ Closed

kervin521 opened this issue on May 4, 2018 · 5 comments



kervin521 commented on May 4, 2018 · edited ▾



data size : 1MB  
use api : bulk api

Rest Client	HighRest client
use time:5s	use time:20s

code example:

<https://github.com/elastic/elasticsearch/issues/30385>

# Elasticsearch REST High Level Client

  **javanna** added the `feedback_needed` label on May 7, 2018



**javanna** commented on Jul 3, 2018

Member ...

Closing this for lack of feedback. Would be nice to have some of the info we asked for, so we can better understand what caused the performance issue. Feel free to reopen if/once you get back to this with the needed info.

  **javanna** closed this as completed on Jul 3, 2018

<https://github.com/elastic/elasticsearch/issues/30385>

# Index API vs Bulk API

# Index API vs Bulk API

```
byte[] data = jsonMapper.writeValueAsBytes(logEvent);

Request request = new Request(
    "POST",
    "/joker/_doc/");
request.setEntity(new ByteArrayEntity(data, ContentType.APPLICATION_JSON));

Response response = restClient.performRequest(request);
```

# Index API vs Bulk API

```
byte[] data = jsonMapper.writeValueAsBytes(logEvent);
```

```
Request request = new Request(  
    "POST",  
    "/joker/_doc/");  
request.setEntity(new ByteArrayEntity(data, ContentType.APPLICATION_JSON));
```

```
Response response = restClient.performRequest(request);
```

# Index API vs Bulk API

```
byte[] data = jsonMapper.writeValueAsBytes(logEvent);

Request request = new Request(
    "POST",
    "/joker/_doc/");
request.setEntity(new ByteArrayEntity(data, ContentType.APPLICATION_JSON));

Response response = restClient.performRequest(request);
```

# Index API vs Bulk API

```
byte[] data = jsonMapper.writeValueAsBytes(logEvent);

Request request = new Request(
    "POST",
    "/joker/_doc/");
request.setEntity(new ByteArrayEntity(data, ContentType.APPLICATION_JSON));

Response response = restClient.performRequest(request);
```

# Index API vs Bulk API

```
byte[] data = jsonMapper.writeValueAsBytes(logEvent);

Request request = new Request(
    "POST",
    "/joker/_doc/");
request.setEntity(new ByteArrayEntity(data, ContentType.APPLICATION_JSON));

Response response = restClient.performRequest(request);
```



# Index API vs Bulk API

```
byte[] data = jsonMapper.writeValueAsBytes(logEvent);

Request request = new Request(
    "POST",
    "/joker/_doc/");
request.setEntity(new ByteArrayEntity(data, ContentType.APPLICATION_JSON));

Response response = restClient.performRequest(request);
```

# Index API vs Bulk API

- Bulk Index API
- Bulk Bulk API

# Bulk Index API

```
ByteArrayOutputStream baos = new ByteArrayOutputStream();
for (LogEvent logEvent : logEvents) {
    baos.write("{\"index\":{}}".getBytes(StandardCharsets.UTF8));
    baos.write('\n');
    jsonMapper.writeValue(baos, logEvent);
    baos.write('\n');
}
byte[] data = baos.toByteArray();

Request request = new Request(
    "POST",
    "/joker/_bulk");
request.setEntity(new ByteArrayEntity(data, ContentType.APPLICATION_JSON));

Response response = restClient.performRequest(request);
```

# Bulk Index API

```
ByteArrayOutputStream baos = new ByteArrayOutputStream();
for (LogEvent logEvent : logEvents) {
    baos.write("{\"index\":{}}".getBytes(StandardCharsets.UTF8));
    baos.write('\n');
    jsonMapper.writeValue(baos, logEvent);
    baos.write('\n');
}
byte[] data = baos.toByteArray();

Request request = new Request(
    "POST",
    "/joker/_bulk");
request.setEntity(new ByteArrayEntity(data, ContentType.APPLICATION_JSON));

Response response = restClient.performRequest(request);
```

# Bulk Index API

```
ByteArrayOutputStream baos = new ByteArrayOutputStream();
for (LogEvent logEvent : logEvents) {
    baos.write("{\"index\":{}}".getBytes(StandardCharsets.UTF8));
    baos.write('\n');
    jsonMapper.writeValue(baos, logEvent);
    baos.write('\n');
}
byte[] data = baos.toByteArray();

Request request = new Request(
    "POST",
    "/joker/_bulk");
request.setEntity(new ByteArrayEntity(data, ContentType.APPLICATION_JSON));

Response response = restClient.performRequest(request);
```

# Bulk Index API

```
ByteArrayOutputStream baos = new ByteArrayOutputStream();
for (LogEvent logEvent : logEvents) {
    baos.write("{\"index\":{\"}}".getBytes(StandardCharsets.UTF8));
    baos.write('\n');
    jsonMapper.writeValue(baos, logEvent);
    baos.write('\n');
}
byte[] data = baos.toByteArray();

Request request = new Request(
    "POST",
    "/joker/_bulk");
request.setEntity(new ByteArrayEntity(data, ContentType.APPLICATION_JSON));

Response response = restClient.performRequest(request);
```

# Bulk Index API

```
ByteArrayOutputStream baos = new ByteArrayOutputStream();
for (LogEvent logEvent : logEvents) {
    baos.write("{\"index\":{}}".getBytes(StandardCharsets.UTF8));
    baos.write('\n');
    jsonMapper.writeValue(baos, logEvent);
    baos.write('\n');
}
byte[] data = baos.toByteArray();

Request request = new Request(
    "POST",
    "/joker/_bulk");
request.setEntity(new ByteArrayEntity(data, ContentType.APPLICATION_JSON));

Response response = restClient.performRequest(request);
```

# Bulk Index API

<https://github.com/elastic/elasticsearch/issues/25673>

```
ByteArrayOutputStream baos = new ByteArrayOutputStream();
for (LogEvent logEvent : logEvents) {
    baos.write("{\"index\":{}}".getBytes(StandardCharsets.UTF8));
    baos.write('\n');
    jsonMapper.writeValue(baos, logEvent);
    baos.write('\n');
}
byte[] data = baos.toByteArray();

Request request = new Request(
    "POST",
    "/joker/_bulk");
request.setEntity(new ByteArrayEntity(data, ContentType.APPLICATION_JSON));

Response response = restClient.performRequest(request);
```



# Bulk Bulk API

```
ByteArrayOutputStream baos = new ByteArrayOutputStream();
for (LogEvent logEvent : logEvents) {
    writeIndex(baos, logEvent);
    baos.write('\n');
    jsonMapper.writeValue(baos, logEvent);
    baos.write('\n');
}
byte[] data = baos.toByteArray();

Request request = new Request(
    "POST",
    "/_bulk");
request.setEntity(new ByteArrayEntity(data, ContentType.APPLICATION_JSON));

Response response = restClient.performRequest(request);
```

# Bulk Bulk API

```
ByteArrayOutputStream baos = new ByteArrayOutputStream();
for (LogEvent logEvent : logEvents) {
    writeIndex(baos, logEvent);
    baos.write('\n');
    jsonMapper.writeValue(baos, logEvent);
    baos.write('\n');
}
byte[] data = baos.toByteArray();

Request request = new Request(
    "POST",
    "/_bulk");
request.setEntity(new ByteArrayEntity(data, ContentType.APPLICATION_JSON));

Response response = restClient.performRequest(request);
```

# Bulk Bulk API

```
ByteArrayOutputStream baos = new ByteArrayOutputStream();
for (LogEvent logEvent : logEvents) {
    writeIndex(baos, logEvent);
    baos.write('\n');
    jsonMapper.writeValue(baos, logEvent);
    baos.write('\n');
}
byte[] data = baos.toByteArray();

Request request = new Request(
    "POST",
    "/_bulk");
request.setEntity(new ByteArrayEntity(data, ContentType.APPLICATION_JSON));

Response response = restClient.performRequest(request);
```

# Bulk Bulk API

```
ByteArrayOutputStream baos = new ByteArrayOutputStream();
for (LogEvent logEvent : logEvents) {
    writeIndex(baos, logEvent);
    baos.write('\n');
    jsonMapper.writeValue(baos, logEvent);
    baos.write('\n');
}
byte[] data = baos.toByteArray();

Request request = new Request(
    "POST",
    "/_bulk");
request.setEntity(new ByteArrayEntity(data, ContentType.APPLICATION_JSON));

Response response = restClient.performRequest(request);
```

# Bulk Bulk API

```
ByteArrayOutputStream baos = new ByteArrayOutputStream();
for (LogEvent logEvent : logEvents) {
    writeIndex(baos, logEvent);
    baos.write('\n');
    jsonMapper.writeValue(baos, logEvent);
    baos.write('\n');
}
byte[] data = baos.toByteArray();

Request request = new Request(
    "POST",
    "/_bulk");
request.setEntity(new ByteArrayEntity(data, ContentType.APPLICATION_JSON));

Response response = restClient.performRequest(request);
```

# Bulk Bulk API

```
/* writeIndex - добавление такого JSON: */  
{  
  "index" : {  
    "_index" : "joker"  
  }  
}
```

# Bulk Bulk API

```
/* writeIndex - добавление такого JSON: */  
{  
  "index" : {  
    "_index" : "joker"  
  }  
}
```

# Идемпотентная запись



# Идемпотентная запись

- По умолчанию – запись не идемпотентна

# Идемпотентная запись

- По умолчанию – запись не идемпотентна
- Каждому документу Elastic даёт уникальный ID

# Идемпотентная запись

- По умолчанию – запись не идемпотентна
- Каждому документу Elastic даёт уникальный ID
- Но можно передавать свой ID

# Идепотентная запись

```
/* writeIndex – добавление такого JSON: */  
{  
  "index" : {  
    "_index" : "joker",  
    "_id" : "<eventId>"  
  }  
}
```

# Идепотентная запись

```
/* writeIndex - добавление такого JSON: */  
{  
  "index" : {  
    "_index" : "joker",  
    "_id" : "<eventId>"  
  }  
}
```

# Идемпотентная запись

Цена

# Идемпотентная запись

Цена

- Вынужденный поиск документа по ID в Lucene

# Идемпотентная запись

## Цена

- Вынужденный поиск документа по ID в Lucene
- ID влияет на распределение по шардам



# Идемпотентная запись

## Цена

- Вынужденный поиск документа по ID в Lucene
- ID влияет на распределение по шардам
- UUID v4 снижает производительность Lucene

# Идемпотентная запись

Примеры хороших ID

# Идемпотентная запись

Примеры хороших ID

- Последовательные ID, выровненные слева нулями

# Идемпотентная запись

## Примеры хороших ID

- Последовательные ID, выровненные слева нулями
- UUID v1

# Идемпотентная запись

## Примеры хороших ID

- Последовательные ID, выровненные слева нулями
- UUID v1
- nanotime

МНОГОПОТОЧНОСТЬ

# МНОГОПОТОЧНОСТЬ

- Много индексов

# МНОГОПОТОЧНОСТЬ

- Много индексов
- Много шардов



# МНОГОПОТОЧНОСТЬ

- Много индексов
- Много шардов
- Много предобработки

# МНОГОПОТОЧНОСТЬ

- Много индексов
- Много шардов
- Много предобработки

Можно делать параллельно!

МНОГОПОТОЧНОСТЬ

Цена

# МНОГОПОТОЧНОСТЬ

Цена

— Тюнить Apache HTTP Async Client

# МНОГОПОТОЧНОСТЬ

## Цена

- Тюнить Apache HTTP Async Client
- Тестировать оптимальное соотношение размера bulk-запроса и уровня параллелизации

# МНОГОПОТОЧНОСТЬ

## Цена

- Тюнить Apache HTTP Async Client
- Тестировать оптимальное соотношение размера bulk-запроса и уровня параллелизации
- Если кластеру Elastic уже плохо, то ему будет ещё хуже

# Обработка ошибок

# Обработка ошибок

- Код ответа не 200 ОК



# Обработка ошибок

- Код ответа не 200 ОК
- Но BulkResponse всегда возвращает 200 ОК!

# Обработка ошибок

- Код ответа не 200 ОК
- Но BulkResponse всегда возвращает 200 ОК!
- **Спойлер:** встроенный фэйловер бесполезен 😞

# Когда 200 OK – не OK

```
{
  "took" : 30000, "errors" : true,
  "items" : [ { "index" : {
    "_index" : "joker", "_id" : "<eventId>",
    "_status" : 400,
    "error" : {
      "type" : "mapper_parsing_exception",
      "reason" : "failed to parse",
      "caused_by" : {
        "type" : "i_o_exception",
        "reason" : "Duplicate field 'message'\n at [Source:...; line: 1, column: 123]"
      }
    }
  } } , ... ]
}
```

# Когда 200 OK – не OK

```
{
  "took" : 30000, "errors" : true,
  "items" : [ { "index" : {
    "_index" : "joker", "_id" : "<eventId>",
    "_status" : 400,
    "error" : {
      "type" : "mapper_parsing_exception",
      "reason" : "failed to parse",
      "caused_by" : {
        "type" : "i_o_exception",
        "reason" : "Duplicate field 'message'\n at [Source:...; line: 1, column: 123]"
      }
    }
  } } , ... ]
}
```

# Когда 200 OK – не OK

```
{
  "took" : 30000, "errors" : true,
  "items" : [ { "index" : {
    "_index" : "joker", "_id" : "<eventId>",
    "_status" : 400,
    "error" : {
      "type" : "mapper_parsing_exception",
      "reason" : "failed to parse",
      "caused_by" : {
        "type" : "i_o_exception",
        "reason" : "Duplicate field 'message'\n at [Source:...; line: 1, column: 123]"
      }
    }
  } } , ... ]
}
```

# Когда 200 OK – не OK

```
{
  "took" : 30000, "errors" : true,
  "items" : [ { "index" : {
    "_index" : "joker", "_id" : "<eventId>",
    "_status" : 400,
    "error" : {
      "type" : "mapper_parsing_exception",
      "reason" : "failed to parse",
      "caused_by" : {
        "type" : "i_o_exception",
        "reason" : "Duplicate field 'message'\n at [Source:...; line: 1, column: 123]"
      }
    }
  } } , ... ]
}
```

# Когда 200 OK – не OK

```
{
  "took" : 30000, "errors" : true,
  "items" : [ { "index" : {
    "_index" : "joker", "_id" : "<eventId>",
    "_status" : 400,
    "error" : {
      "type" : "mapper_parsing_exception",
      "reason" : "failed to parse",
      "caused_by" : {
        "type" : "i_o_exception",
        "reason" : "Duplicate field 'message'\n at [Source:...; line: 1, column: 123]"
      }
    }
  } } , ... ]
}
```

# Когда 200 OK – не OK

```
{
  "took" : 30000, "errors" : true,
  "items" : [ { "index" : {
    "_index" : "joker", "_id" : "<eventId>",
    "_status" : 400,
    "error" : {
      "type" : "mapper_parsing_exception",
      "reason" : "failed to parse",
      "caused_by" : {
        "type" : "i_o_exception",
        "reason" : "Duplicate field 'message'\n at [Source:...; line: 1, column: 123]"
      }
    }
  } } , ... ]
}
```



# Когда 200 ОК – не ОК

```
{
  "took" : 30000, "errors" : true,
  "items" : [ { "index" : {
    "_index" : "joker", "_id" : "<eventId>",
    "_status" : 400,
    "error" : {
      "type" : "mapper_parsing_exception",
      "reason" : "failed to parse",
      "caused_by" : {
        "type" : "i_o_exception",
        "reason" : "Duplicate field 'message'\n at [Source:...; line: 1, column: 123]"
      }
    }
  } } , ... ]
}
```

# Когда 200 OK – не OK

```
{
  "took" : 30000, "errors" : true,
  "items" : [ { "index" : {
    "_index" : "joker", "_id" : "<eventId>",
    "_status" : 400,
    "error" : {
      "type" : "mapper_parsing_exception",
      "reason" : "failed to parse",
      "caused_by" : {
        "type" : "i_o_exception",
        "reason" : "Duplicate field 'message'\n at [Source:...; line: 1, column: 123]"
      }
    }
  } } , ... ]
}
```

# Когда 200 OK – не OK

```
{
  "took" : 30000, "errors" : true,
  "items" : [ { "index" : {
    "_index" : "joker", "_id" : "<eventId>",
    "_status" : 400,
    "error" : {
      "type" : "mapper_parsing_exception",
      "reason" : "failed to parse",
      "caused_by" : {
        "type" : "i_o_exception",
        "reason" : "Duplicate field 'message'\n at [Source:...; line: 1, column: 123]"
      }
    }
  } } , ... ]
}
```

# Когда 200 OK – не OK

```
{
  "took" : 30000, "errors" : true,
  "items" : [ { "index" : {
    "_index" : "joker", "_id" : "<eventId>",
    "_status" : 400,
    "error" : {
      "type" : "mapper_parsing_exception",
      "reason" : "failed to parse",
      "caused_by" : {
        "type" : "i_o_exception",
        "reason" : "Duplicate field 'message'\n at [Source:...; line: 1, column: 123]"
      }
    }
  } } , ... ]
}
```

# Когда 200 OK – не OK

```
{
  "took" : 30000, "errors" : true,
  "items" : [ { "index" : {
    "_index" : "joker", "_id" : "<eventId>",
    "_status" : 400,
    "error" : {
      "type" : "mapper_parsing_exception",
      "reason" : "failed to parse",
      "caused_by" : {
        "type" : "i_o_exception",
        "reason" : "Duplicate field 'message'\n at [Source:...; line: 1, column: 123]"
      }
    }
  } } , ... ]
}
```

# Когда 200 OK – не OK

```
{
  "took" : 30000, "errors" : true,
  "items" : [ { "index" : {
    "_index" : "joker", "_id" : "<eventId>",
    "_status" : 400,
    "error" : {
      "type" : "mapper_parsing_exception",
      "reason" : "failed to parse",
      "caused_by" : {
        "type" : "i_o_exception",
        "reason" : "Duplicate field 'message'\n at [Source:...; line: 1, column: 123]"
      }
    }
  } } , ... ]
}
```

# Когда 200 OK – не OK

```
{
  "took" : 30000, "errors" : true,
  "items" : [ { "index" : {
    "_index" : "joker", "_id" : "<eventId>",
    "_status" : 400,
    "error" : {
      "type" : "mapper_parsing_exception",
      "reason" : "failed to parse",
      "caused_by" : {
        "type" : "i_o_exception",
        "reason" : "Duplicate field 'message'\n at [Source:...; line: 1, column: 123]"
      }
    }
  } } , ... ]
}
```

Длинный путь – Exceptions



# Длинный путь – Exceptions

– Всего 150+ исключений

# Длинный путь – Exceptions

- Всего 150+ исключений

## Разруха

- в кластере

- в данных

# Длинный путь – Exceptions

- Всего 150+ исключений

## Разруха

- в кластере – retry + backoff

- в данных

# Длинный путь – Exceptions

- Всего 150+ исключений

## Разруха

- в кластере – `retry + backoff`

- в данных – лепрозорий

Лепрозорий – Leprosery

Специальный индекс в Elastic  
для «плохих» сообщений

# Лепрозорий – Leprosery

Специальный индекс в Elastic  
для «плохих» сообщений

– Ошибки маппинга (разные типы значений в поле)

# Лепрозорий – Leprosery

Специальный индекс в Elastic  
для «плохих» сообщений

- Ошибки маппинга (разные типы значений в поле)
- Запись в закрытый индекс (логи из прошлого)

# Лепрозорий – Leprosery

Специальный индекс в Elastic  
для «плохих» сообщений

- Ошибки маппинга (разные типы значений в поле)
- Запись в закрытый индекс (логи из прошлого)
- Некорректное название индекса



# Лепрозорий – Leprosery

Специальный индекс в Elastic  
для «плохих» сообщений

- Ошибки маппинга (разные типы значений в поле)
- Запись в закрытый индекс (логи из прошлого)
- Некорректное название индекса
- ...

# Вместо выводов

“Lucene — отличный движок  
для полнотекстового поиска

Elastic — плохой сетевой код,  
обмазанный вокруг Lucene”

— коллега про Elasticsearch 2.0

# Вместо выводов

“Lucene — отличный движок  
для полнотекстового поиска

Elastic — плохой сетевой код,  
обмазанный вокруг Lucene”

— коллега про Elasticsearch 2.0

# Вместо выводов

“Lucene — отличный движок  
для полнотекстового поиска

Elastic — плохой сетевой код,  
обмазанный вокруг Lucene”

— коллега про Elasticsearch 2.0

# Вместо выводов

Иногда изменяется публичный интерфейс клиента при минорных обновлениях

# Вместо выводов

Иногда изменяется публичный интерфейс клиента при минорных обновлениях

```
public final class RestClientBuilder {  
    /* ... */  
    public static final int DEFAULT_CONNECTION_REQUEST_TIMEOUT_MILLIS = 500;  
    /* ... */  
}
```

# Вместо выводов

Иногда изменяется публичный интерфейс клиента при минорных обновлениях

```
public final class RestClientBuilder {  
    /* ... */  
    public static final int DEFAULT_CONNECTION_REQUEST_TIMEOUT_MILLIS = 500;  
    /* ... */  
}
```

Выпилили в **6.3**

<https://github.com/elastic/elasticsearch/pull/30384>

# Вместо выводов

Иногда RestClient ломается

<https://github.com/elastic/elasticsearch/issues/42133>



# Вместо выводов

Иногда RestClient ломается

```
/**
 * check client running status
 * @return client running status
 */
public boolean isRunning() {
    return client.isRunning();
}
```

<https://github.com/elastic/elasticsearch/pull/57973>

# Вместо выводов

Иногда RestClient ломается

```
/**
 * check client running status
 * @return client running status
 */
public boolean isRunning() {
    return client.isRunning();
}
```

С версии **8.0** и **7.9+**

<https://github.com/elastic/elasticsearch/pull/57973>

Исходники на GitHub

<https://github.com/vostok/hercules>

Make telemetry great again!

# Elasticsearch-клиенты

- Java (Native, REST client, High Level REST client)
- .NET (REST client, High Level REST client)
- Node.JS, Ruby, Go, PHP, Perl, Python

<https://www.elastic.co/guide/en/elasticsearch/client/index.html>

# Elasticsearch-клиенты

- Java (Native, REST client, High Level REST client)
- .NET (REST client, High Level REST client)
- Node.JS, Ruby, Go, PHP, Perl, Python
- Rust (альфа-версия, с версии 8.0)

# А как в других клиентах?

## Совместимость с версиями Elasticsearch

- .NET – аналогично Java
- Node.JS, Ruby, Go, PHP, Perl, Python, Rust –  
**version specific** клиент

# А как в других клиентах?

Балансировка по всем доступным нодам

- .NET – аналогично Java
- Node.js, Go, Python – exponential backoff (1, 2, 4, 8, 16, 32 минуты)
- PHP, Perl – exponential backoff (до 1 часа)
- Ruby – exponential backoff (без ограничения сверху)

# А как в других клиентах?

## Фэйловер

- .NET, Node.JS, Ruby, Go, Python
- PHP, Perl – только connection-ошибки  
(connection refused/timeout, DNS lookup timeout, ...)



# А как в других клиентах?

Постоянные коннекциии к кластеру

- .NET, Node.JS, Ruby, Go, PHP, Perl, Python –  
все HTTP-клиенты умеют в **Keep-Alive**

# А как в других клиентах?

Актуализация топологии кластера

- .NET, Node.JS, Ruby, PHP, Perl, Python
- Go – не реализовано,  
но есть альтернативный клиент, который всё умеет

<https://github.com/olivere/elastic>