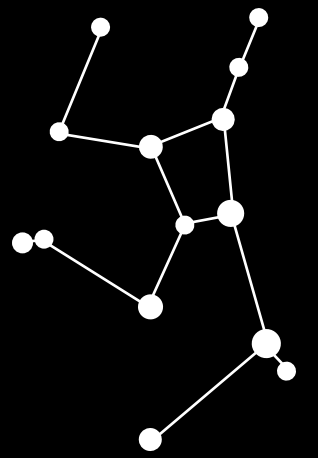
Григорий Кошелев
СКБ Контур



Нельзя просто так взять и отправить все логи в Elastic

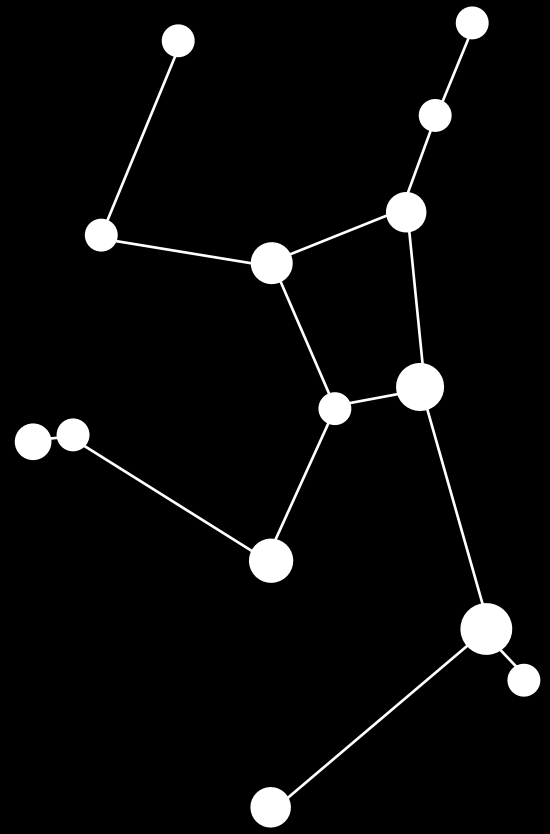
java.ural.Meetup @3, Екатеринбург, 2019

Hercules – транспорт для телеметрии

Логи

Метрики

Распределённые трассировки



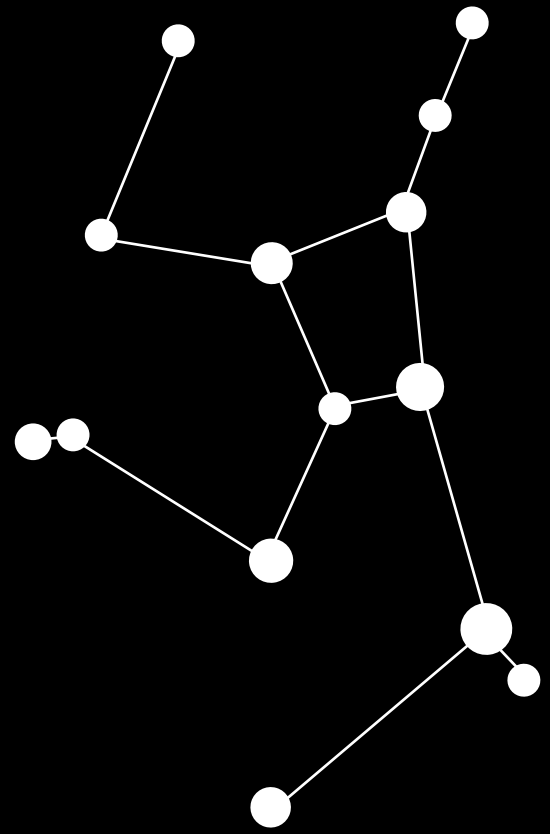
https://github.com/vostok/hercules

Hercules – транспорт для телеметрии

Логи

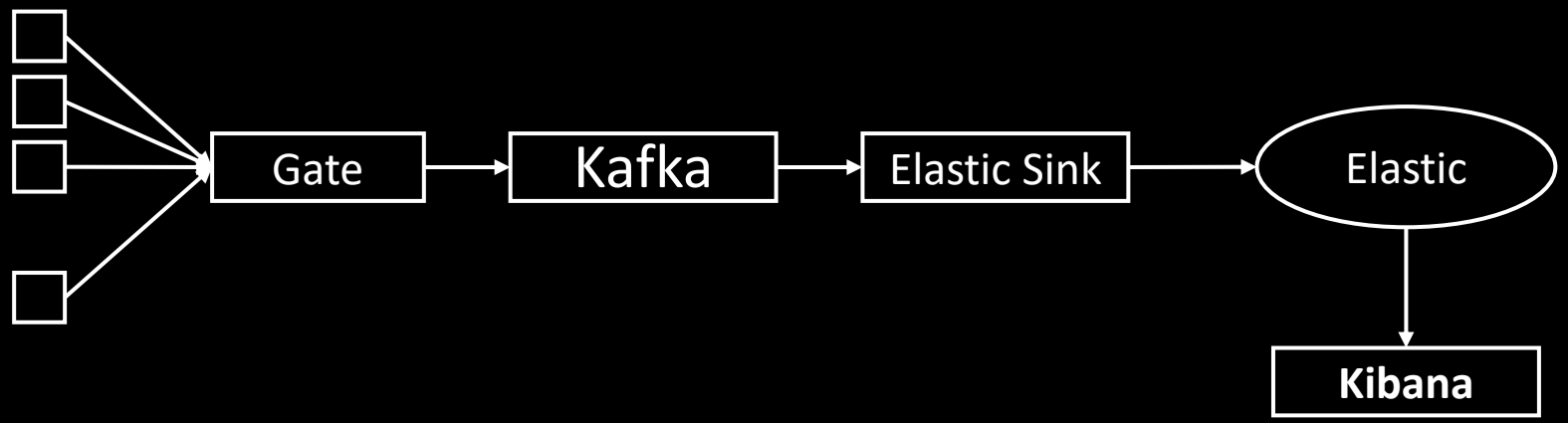
Метрики

Распределённые трассировки



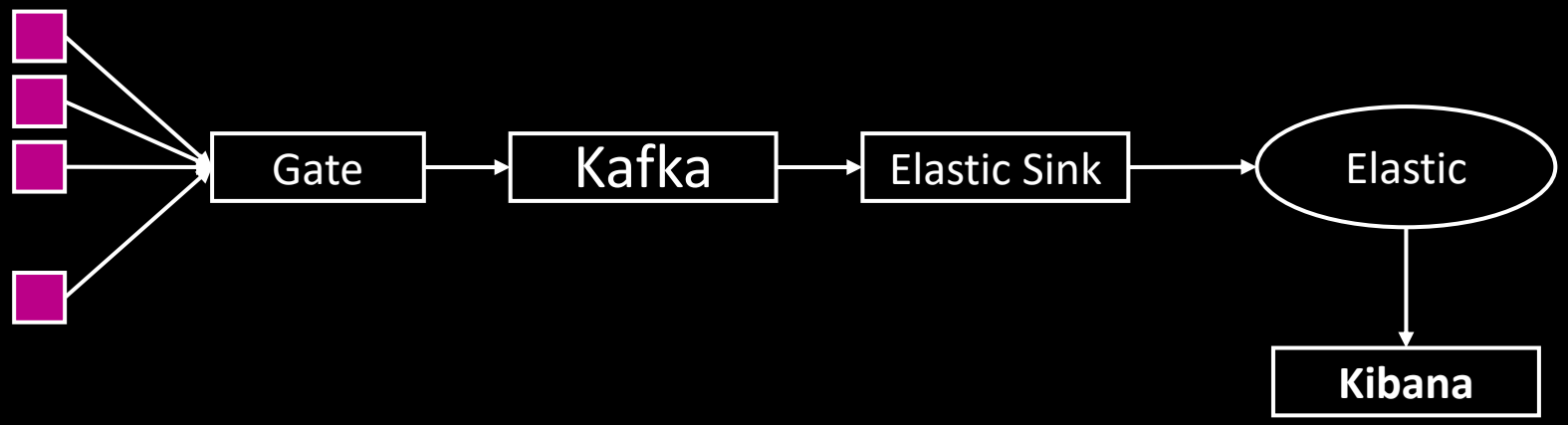
https://github.com/vostok/hercules

Hercules – доставляем Логи



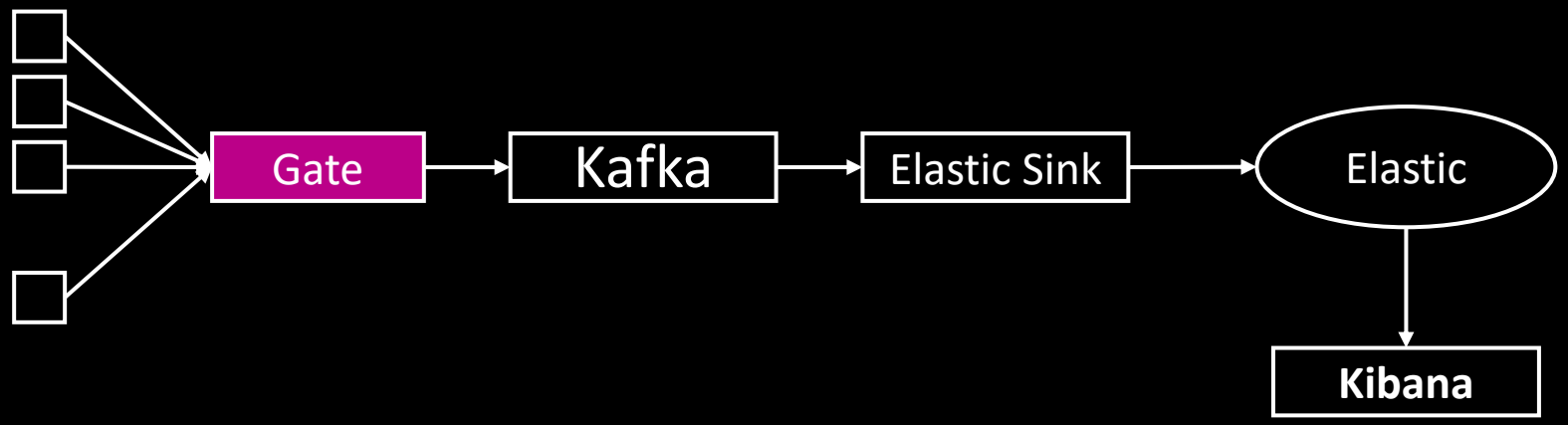
https://github.com/vostok/hercules

Hercules – доставляем Логи



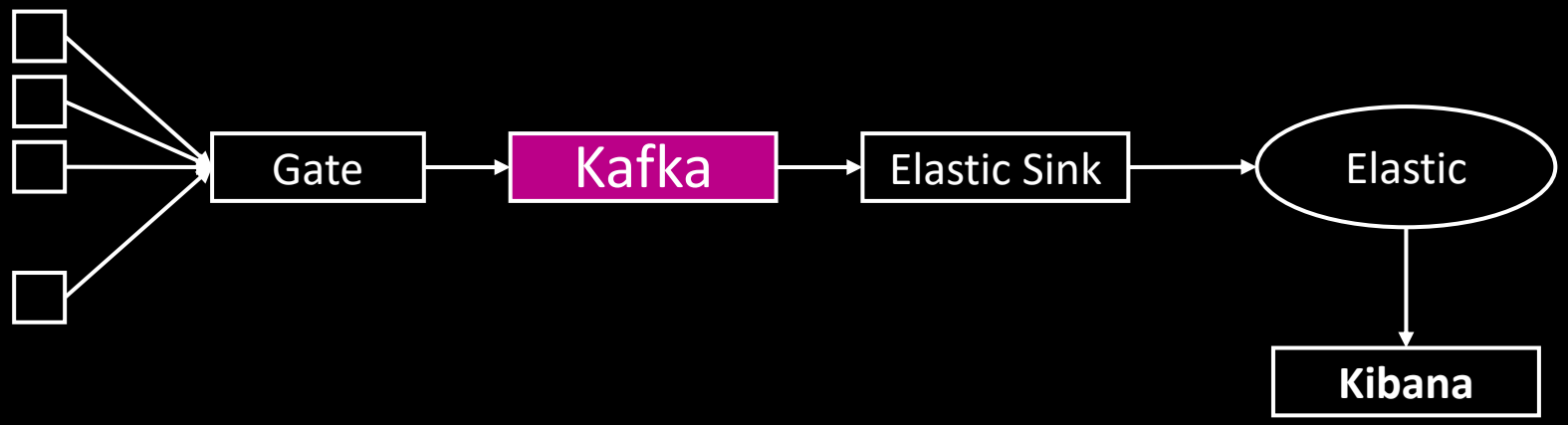
https://github.com/vostok/hercules

Hercules – доставляем Логи



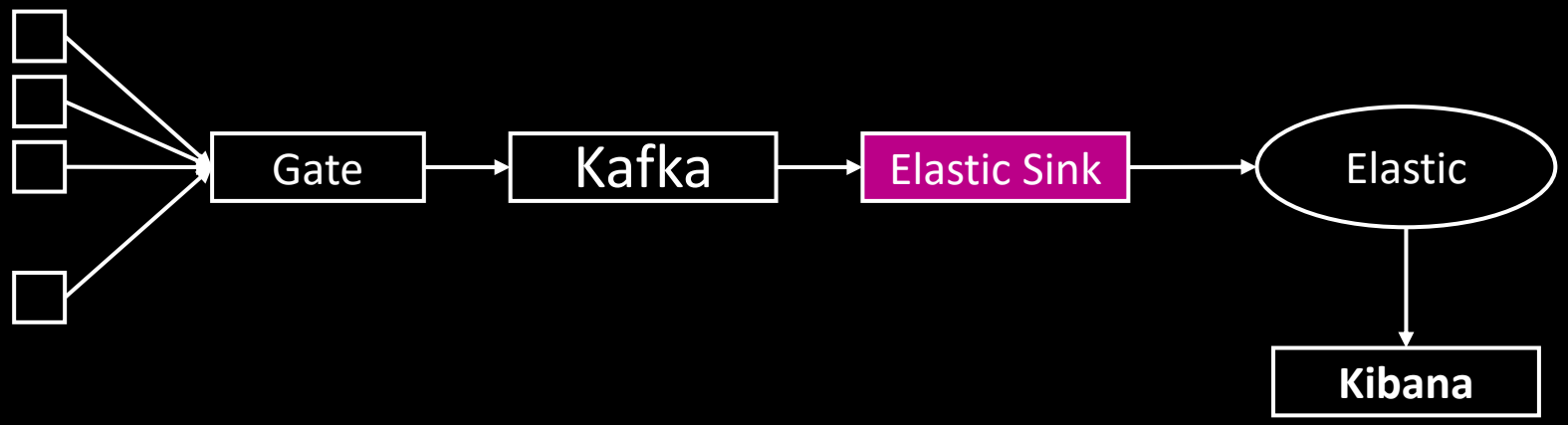
https://github.com/vostok/hercules

Hercules – доставляем Логи



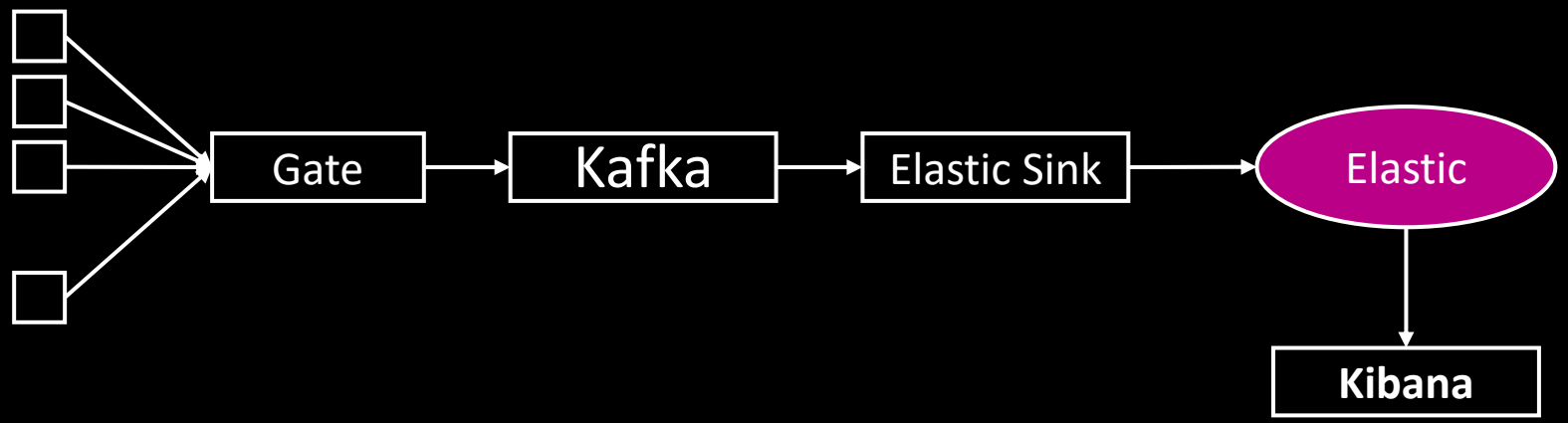
https://github.com/vostok/hercules

Hercules – доставляем Логи



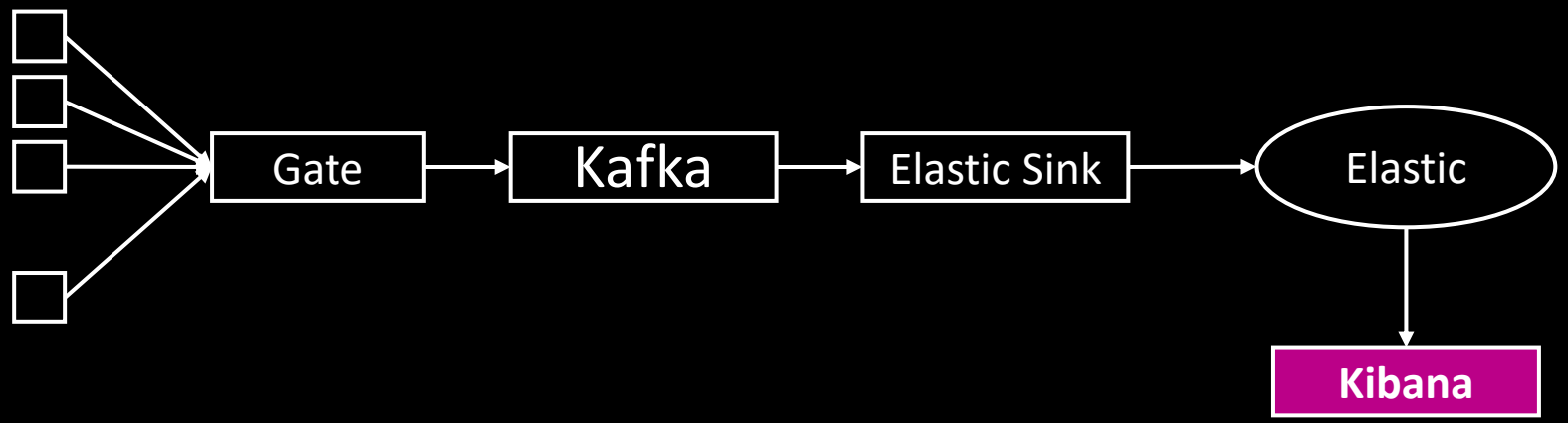
https://github.com/vostok/hercules

Hercules – доставляем Логи



https://github.com/vostok/hercules

Hercules – доставляем Логи



https://github.com/vostok/hercules

Как устроены Логи

```
{
  "@timestamp" : "2019-09-21T06:30:00.123456789Z",
  "loggerName" : "java.ural.Meetup",
  "level" : "INFO",
  "message" : "Утро начинается с кофе",
  "user" : "Gregory",
  "timeoutMs" : 120000,
}
```

Как устроены Логи

```
{
  "@timestamp" : "2019-09-21T06:30:00.1234567890Z",
  "loggerName" : "java.ural.Meetup",
  "level" : "INFO",
  "message" : "Утро начинается с кофе",
  "user" : "Gregory",
  "timeoutMs" : 120000,
}
```

Как устроены Логи

```
{
  "@timestamp" : "2019-09-21T06:30:00.123456789Z",
  "loggerName" : "java.ural.Meetup",
  "level" : "INFO",
  "message" : "Утро начинается с кофе",
  "user" : "Gregory",
  "timeoutMs" : 120000,
}
```

Как устроены Логи

```
{
  "@timestamp" : "2019-09-21T06:30:00.123456789Z",
  "loggerName" : "java.ural.Meetup",
  "level" : "INFO",
  "message" : "Утро начинается с кофе",
  "user" : "Gregory",
  "timeoutMs" : 120000,
}
```

Как устроены Логи

```
{
  "@timestamp" : "2019-09-21T06:30:00.123456789Z",
  "loggerName" : "java.ural.Meetup",
  "level" : "INFO",
  "message" : "Утро начинается с кофе",
  "user" : "Gregory",
  "timeoutMs" : 120000,
}
```

Как устроены Логи

```
{
  "@timestamp" : "2019-09-21T06:30:00.123456789Z",
  "loggerName" : "java.ural.Meetup",
  "level" : "INFO",
  "message" : "Утро начинается с кофе",
  "user" : "Gregory",
  "timeoutMs" : 120000,
}
```

Как устроены Логи

```
{
  "@timestamp" : "2019-09-21T06:30:00.123456789Z",
  "loggerName" : "java.ural.Meetup",
  "level" : "INFO",
  "message" : "Утро начинается с кофе",
  "user" : "Gregory",
  "timeoutMs" : 120000,
}
```

Как разделять Логи

Как разделять Логи

- Проект / команда

Как разделять Логи

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

Как разделять Логи

- Проект / команда
- Окружение (staging, production, …)
- Время (день, неделя, месяц, …)

Как разделять Логи

- Проект / команда
- Окружение (staging, production, …)
- Время (день, неделя, месяц, …)

Индекс: *<проект>-<окружение>-<YYYY.MM.DD>*

Ок, в чём проблема?

Ок, в чём проблема?

- 70 000 лог записей в секунду

Ок, в чём проблема?

- 70 000 лог записей в секунду
- 200 000+ в пике

Ок, в чём проблема?

- 70 000 лог записей в секунду
- 200 000+ в пике
- 3+ ТВ логов в сутки

Ок, в чём проблема?

- 70 000 лог записей в секунду
- 200 000+ в пике
- 3+ ТВ логов в сутки

И это всё в Elastic

Особенности работы с API

Особенности работы с API

Native Java Client: TransportClient

https://www.elastic.co/guide/en/elasticsearch/client/java-api/7.3/transport-client.html

Особенности работы с API

Native Java Client: TransportClient

— Высокоуровневые абстракции (Request, Response)

https://www.elastic.co/guide/en/elasticsearch/client/java-api/7.3/transport-client.html

Особенности работы с API

Native Java Client: TransportClient

— Высокоуровневые абстракции (Request, Response)

— Отсутствует обратная совместимость между версиями

https://www.elastic.co/guide/en/elasticsearch/client/java-api/7.3/transport-client.html

Особенности работы с API

Native Java Client: TransportClient

— Высокоуровневые абстракции (Request, Response)

— Отсутствует обратная совместимость между версиями

— Deprecated с версии Elasticsearch 7.0

https://www.elastic.co/guide/en/elasticsearch/client/java-api/7.3/transport-client.html

Особенности работы с API

Native Java Client: TransportClient

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

https://www.elastic.co/guide/en/elasticsearch/client/java-api/7.3/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 REST Client

Версия клиента: 7.3 (последняя на данный момент)

Elasticsearch REST Client

- Совместим с любой версией Elasticsearch

Elasticsearch REST Client

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

Elasticsearch REST Client

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

Elasticsearch REST Client

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

Elasticsearch REST Client

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

Elasticsearch 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 REST Client

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

Elasticsearch REST Client

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


// Ещё один способ с версии 6.4
Node[] nodes = ... // elastic nodes with metadata
RestClient restClient = RestClient.builder(nodes).build();
```

Elasticsearch REST Client

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

Elasticsearch 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 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 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 REST Client

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

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

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

Elasticsearch REST Client

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

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

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

Elasticsearch REST Client

Выводы

Elasticsearch REST Client

Выводы

– Обновление Elastic независимо от клиентов

Elasticsearch REST Client

Выводы

- Обновление Elastic независимо от клиентов
- Можно долго не менять версию клиента

Elasticsearch 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 REST Client

Elasticsearch REST Client

- Хост для каждого запроса выбирается по round robin

Elasticsearch REST Client

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

Elasticsearch REST Client

- Хост для каждого запроса выбирается по round robin
- Если запрос завершился ошибкой (код >=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 REST Client

- Хост для каждого запроса выбирается по round robin
- Если запрос завершился ошибкой (код >=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 REST Client

Выводы

Elasticsearch REST Client

Выводы

– Первая поднявшаяся нода Эластика будет страдать

Elasticsearch REST Client

Выводы

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

Elasticsearch 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 REST Client

Фэйловер

Elasticsearch REST Client

Фэйловер

– если код 502, 503 или 504

Elasticsearch REST Client

Фэйловер

— если код 502, 503 или 504

— если доступных хостов >1 (blacklist!)

Elasticsearch REST Client

Фэйловер

— если код 502, 503 или 504

— если доступных хостов >1 (blacklist!)

— если осталось время (до версии 7.0)

Elasticsearch 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 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 REST Client

Выводы

Elasticsearch REST Client

Выводы

– Поведение зависит от версии клиента

Elasticsearch REST Client

Выводы

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

Elasticsearch 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 REST Client

ApacheHttpAsyncClient же!

Elasticsearch REST Client

ApacheHttpAsyncClient же!

– Keep-Alive (HTTP 1.1)

Elasticsearch REST Client

ApacheHttpAsyncClient же!

- Keep-Alive (HTTP 1.1)
- PoolingNHttpClientConnectionManager

Elasticsearch 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 REST Client

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

Elasticsearch REST Client

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

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

Elasticsearch REST Client

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

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

⇩

Elasticsearch 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 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 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 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 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 REST Client

Выводы

Elasticsearch REST Client

Выводы

– sniffer.close() --> restClient.close()

Elasticsearch REST Client

Выводы

– sniffer.close() --> restClient.close()

– Sniffer очищает blacklist

Elasticsearch REST Client

Выводы

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

Elasticsearch 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?

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 HttpAsyncClient (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

Index API vs Bulk API

Index API vs Bulk API

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

Request request = new Request(
                    "POST",
                    "/meetup_prod_2019.09.21/LogEvent/");
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",
                    "/meetup_prod_2019.09.21/LogEvent/");
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",
                    "/meetup_prod_2019.09.21/LogEvent/");
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",
                    "/meetup_prod_2019.09.21/LogEvent/");
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",
                    "/meetup_prod_2019.09.21/LogEvent/");
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",
                    "/meetup_prod_2019.09.21/LogEvent/");
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",
                    "/meetup_prod_2019.09.21/LogEvent/_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",
                    "/meetup_prod_2019.09.21/LogEvent/_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",
                    "/meetup_prod_2019.09.21/LogEvent/_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",
                    "/meetup_prod_2019.09.21/LogEvent/_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",
                    "/meetup_prod_2019.09.21/LogEvent/_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",
                    "/meetup_prod_2019.09.21/LogEvent/_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",
                    "/meetup_prod_2019.09.21/LogEvent/_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" : "meetup_prod_2019.09.21",
    "_type" : "LogEvent"
  }
}
```

Bulk Bulk API

```
/* writeIndex – добавление такого JSON: */
{
  "index" : {
    "_index" : "meetup_prod_2019.09.21",
    "_type" : "LogEvent"
  }
}
```

Bulk Bulk API

```
/* writeIndex – добавление такого JSON: */
{
  "index" : {
    "_index" : "meetup_prod_2019.09.21",
    "_type" : "LogEvent"
  }
}
```

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

Идепотентая запись

```
/* writeIndex – добавление такого JSON: */
{
  "index" : {
    "_index" : "meetup_prod_2019.09.21",
    "_type" : "LogEvent"",
    "_id" : "<eventId>"
  }
}
```

Идепотентая запись

```
/* writeIndex – добавление такого JSON: */
{
  "index" : {
    "_index" : "meetup_prod_2019.09.21",
    "_type" : "LogEvent"",
    "_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

http://blog.mikemccandless.com/2014/05/choosing-fast-unique-identifier-uuid.html

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 OK

Обработка ошибок

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

Когда 200 OK – не OK


```
{
  "took" : 30000, "errors" : true,
  "items" : [ { "index" : {
    "_index" : "meetup_prod_2019.09.21", "_type" : "LogEvent", "_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" : "meetup_prod_2019.09.21", "_type" : "LogEvent", "_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" : "meetup_prod_2019.09.21", "_type" : "LogEvent", "_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" : "meetup_prod_2019.09.21", "_type" : "LogEvent", "_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" : "meetup_prod_2019.09.21", "_type" : "LogEvent", "_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" : "meetup_prod_2019.09.21", "_type" : "LogEvent", "_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" : "meetup_prod_2019.09.21", "_type" : "LogEvent", "_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" : "meetup_prod_2019.09.21", "_type" : "LogEvent", "_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" : "meetup_prod_2019.09.21", "_type" : "LogEvent", "_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" : "meetup_prod_2019.09.21", "_type" : "LogEvent", "_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" : "meetup_prod_2019.09.21", "_type" : "LogEvent", "_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" : "meetup_prod_2019.09.21", "_type" : "LogEvent", "_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" : "meetup_prod_2019.09.21", "_type" : "LogEvent", "_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" : "meetup_prod_2019.09.21", "_type" : "LogEvent", "_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

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

Исходники на GitHub

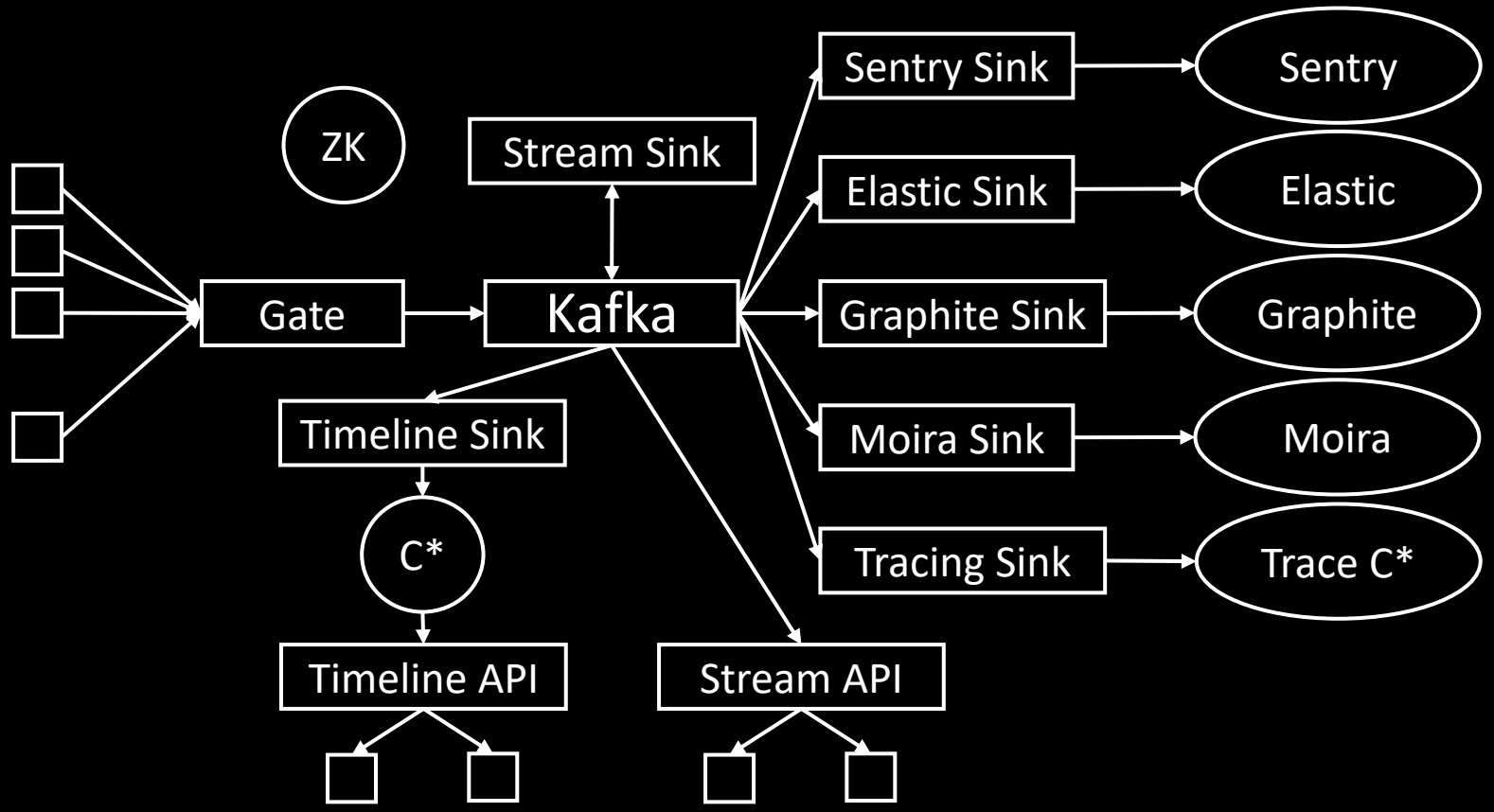
https://github.com/vostok

https://github.com/vostok/hercules

 https://t.me/java_ural_Meetup

Make telemetry great again!

Hercules under the hood



https://github.com/vostok/hercules