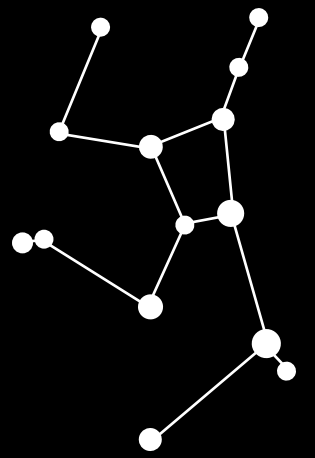
Григорий Кошелев
СКБ Контур



# Нельзя просто так взять и отправить все логи в Elastic

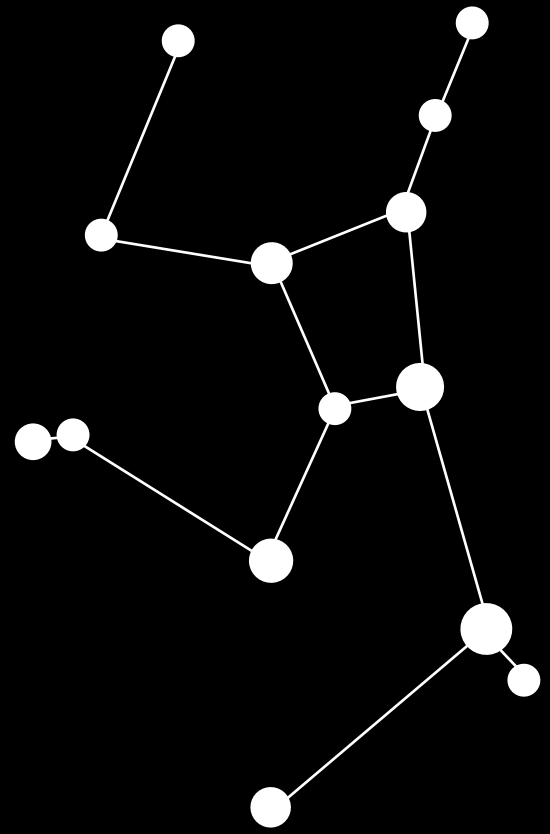
DUMP, Казань, 2019

# Hercules – транспорт для телеметрии

Логи

Метрики

Распределённые трассировки



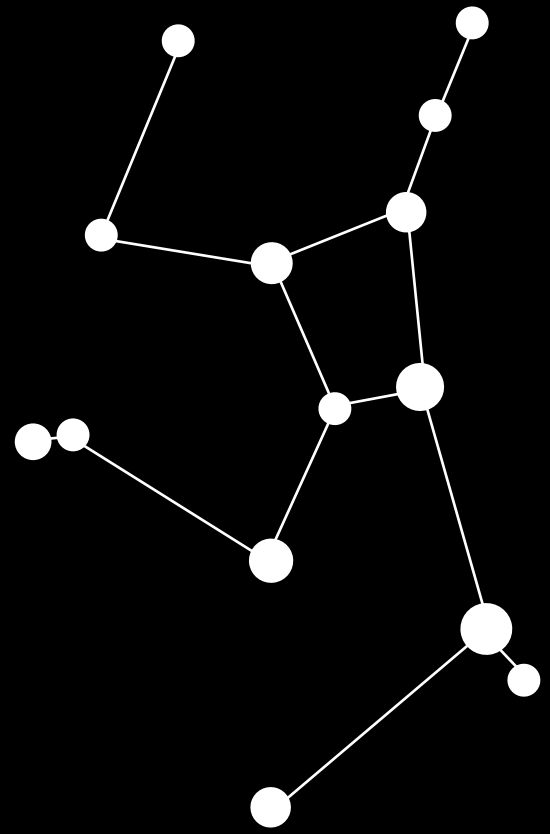
https://github.com/vostok/hercules

# Hercules – транспорт для телеметрии

**Логи**

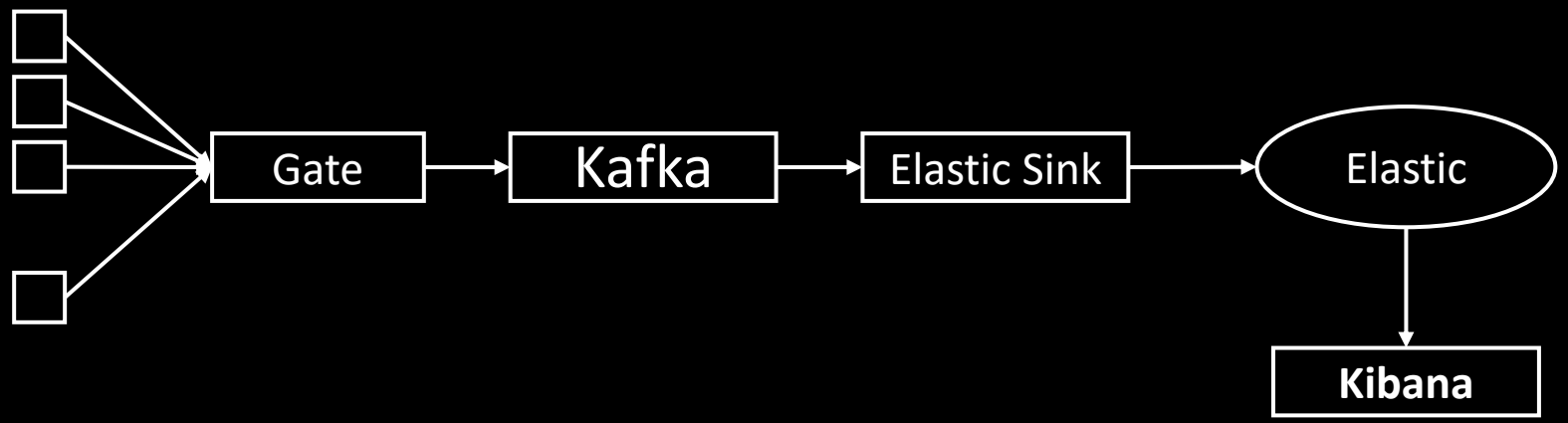
Метрики

Распределённые трассировки



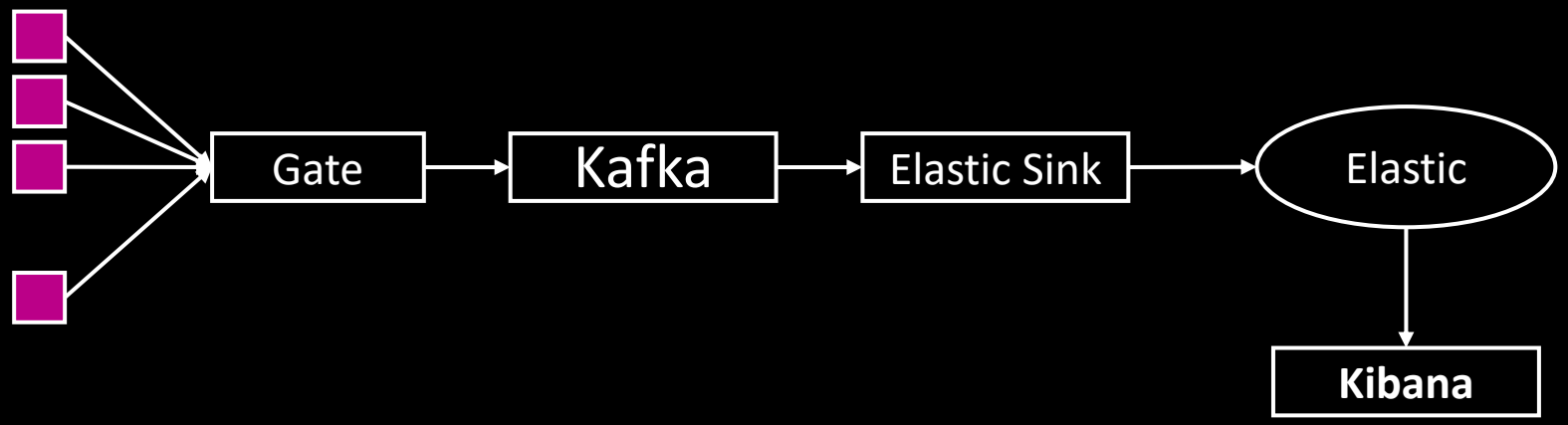
https://github.com/vostok/hercules

# Hercules – доставляем Логи



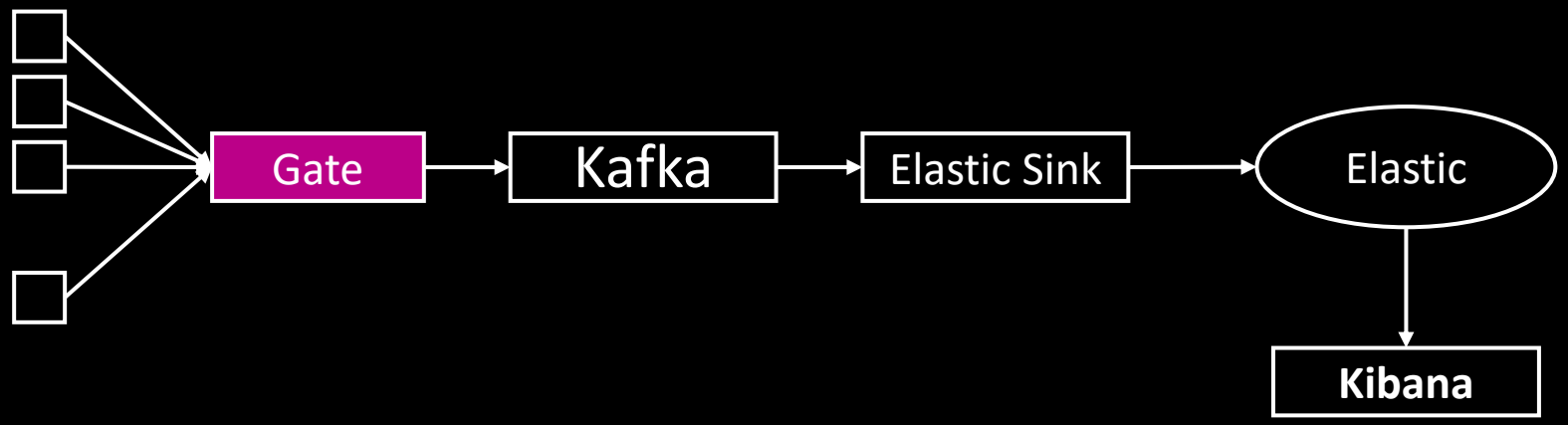

https://github.com/vostok/hercules

# Hercules – доставляем Логи



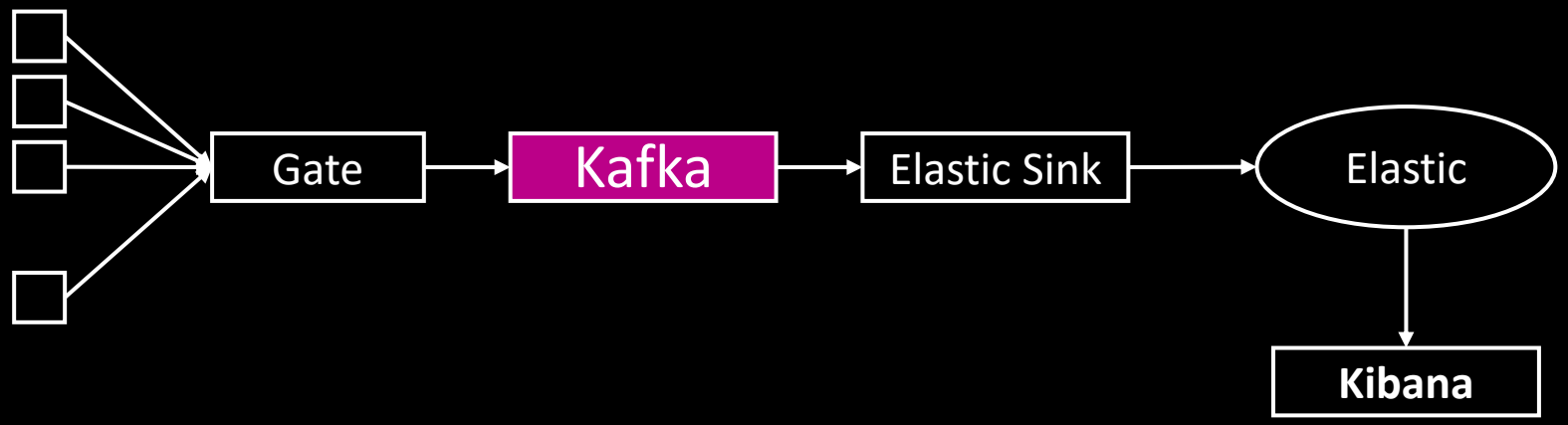

https://github.com/vostok/hercules

# Hercules – доставляем Логи



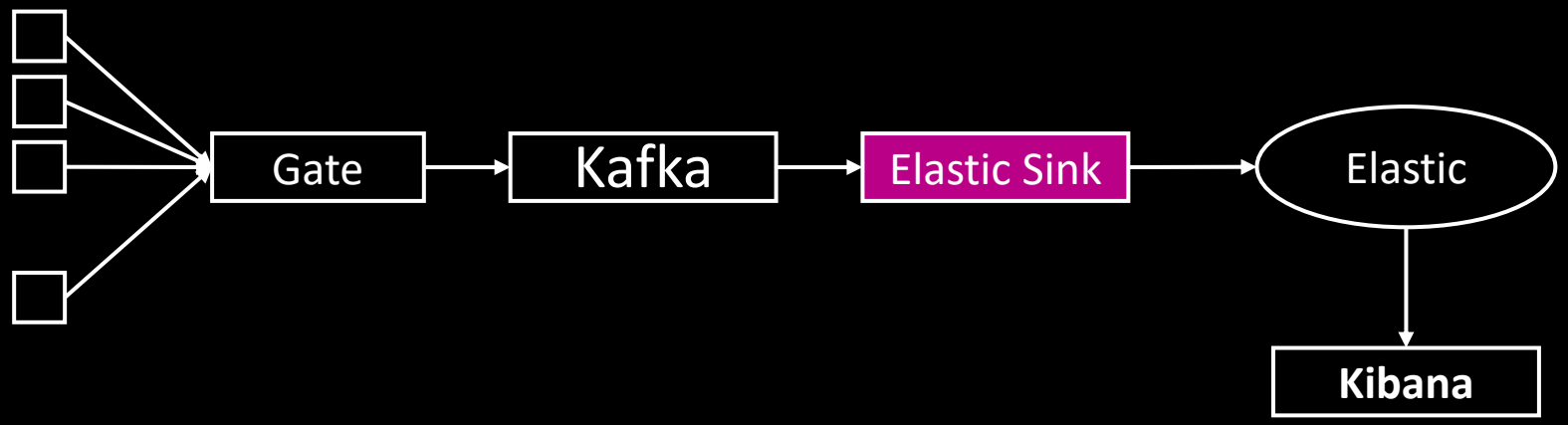

https://github.com/vostok/hercules

# Hercules – доставляем Логи



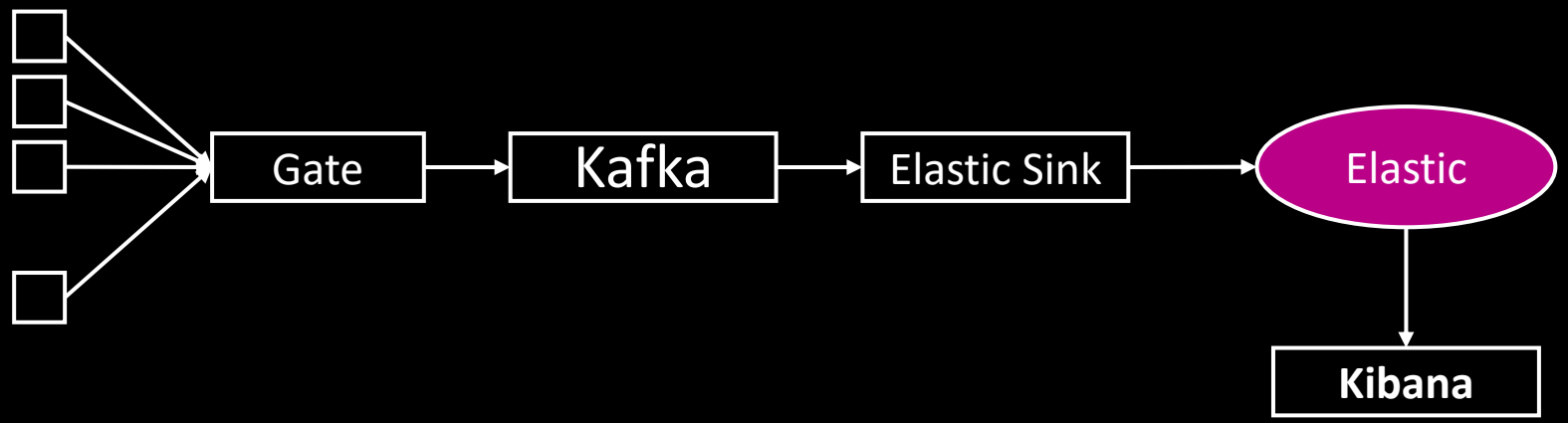

https://github.com/vostok/hercules

# Hercules – доставляем Логи



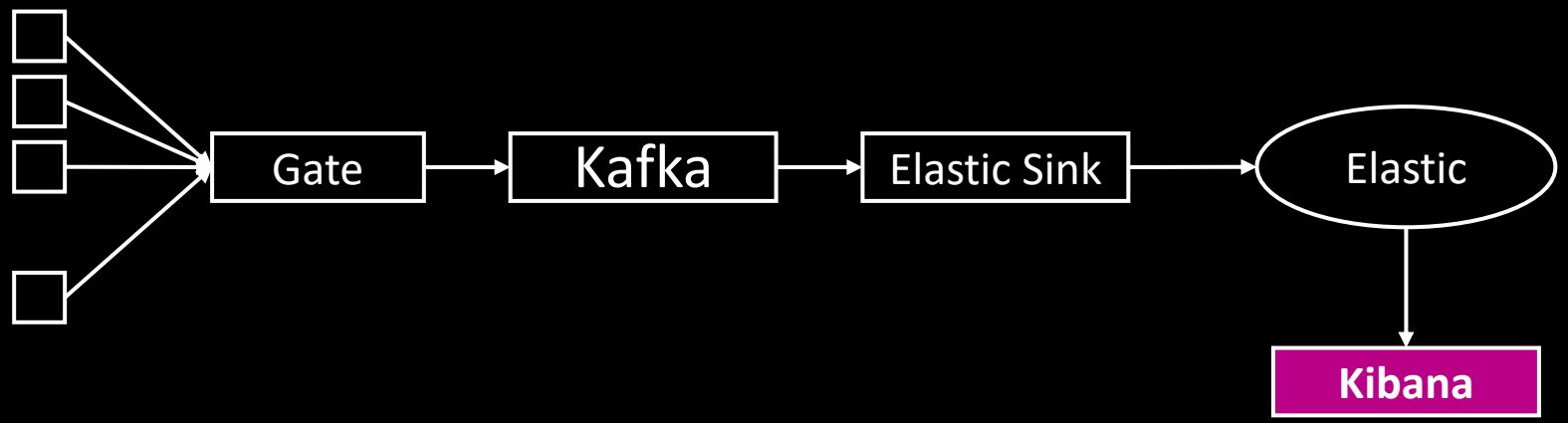

https://github.com/vostok/hercules

# 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
- ~ 5 ТВ логов в сутки

# Ок, в чём проблема?

- 70 000 лог записей в секунду
- 200 000+ в пике
- ~ 5 ТВ логов в сутки

И это всё в Elastic

# Особенности работы с API

# Особенности работы с API

Elasticsearch-клиенты

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

# Особенности работы с API

Elasticsearch-клиенты

– Java (Native, REST client, High Level REST client)

# Особенности работы с API

Elasticsearch-клиенты

- Java (Native, REST client, High Level REST client)
- .NET (REST client, High Level REST client)

# Особенности работы с API

Elasticsearch-клиенты

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

# Особенности работы с API

Elasticsearch-клиенты

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

Native Java client – deprecated с версии 7.0

# Особенности работы с API

Elasticsearch-клиенты

– Java (Native, REST client, High Level REST client)

– .NET (REST client, High Level REST client)

– Node.JS, Ruby, Go, PHP, Perl, Python

High Level REST client – обёртка над REST client

# Elasticsearch (Java) REST Client

Версия клиента: 7.4

8.0 – выйдет *когда-нибудь* в будущем

# 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/my_type/");
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/my_type/");
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)

# А как в других клиентах?

Совместимость с версиями Elasticsearch

– .NET – аналогично Java

# А как в других клиентах?

Совместимость с версиями Elasticsearch

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

# 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 >=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 >=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 >=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

# А как в других клиентах?

Балансировка по всем доступным нодам

— .NET

# А как в других клиентах?

Балансировка по всем доступным нодам

- .NET
- Node.JS, Go, Python –
  exponential backoff (1, 2, 4, 8, 16, 32 минут)

# А как в других клиентах?

Балансировка по всем доступным нодам

- .NET
- Node.JS, Go, Python –
  exponential backoff (1, 2, 4, 8, 16, 32 минут)
- PHP, Perl – exponential backoff (до 1 часа)

# А как в других клиентах?

Балансировка по всем доступным нодам

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

# 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
- Важно корректно настроить таймауты в HTTP client

# А как в других клиентах?

Фэйловер

– .NET, Node.JS, Ruby, Go, Python

# А как в других клиентах?

Фэйловер

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

# 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

Keep-Alive (HTTP 1.1)

# А как в других клиентах?

Постоянные коннекции к кластеру

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

# 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?

# А как в других клиентах?

Актуализация топологии кластера

– .NET, Node.JS, Ruby, PHP, Perl, Python

# 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

# HTTP Client

# HTTP Client

Замечания

# HTTP Client

Замечания

- Важно понимать, как работает HTTP client (и его настройки)

# HTTP Client

Замечания

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

# HTTP Client

Замечания

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

# 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

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 HTTP client

# Многопоточность

Цена

- Тюнить HTTP client
- Тестировать оптимальное соотношение размера bulk-запроса и уровня параллелизации

# Многопоточность

Цена

- Тюнить HTTP 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!

# Обработка ошибок

- Код ответа не 200 OK
- Но BulkResponse всегда возвращает 200 OK!
- Спойлер: встроенный фэйловер бесполезен ☹

# 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
- …

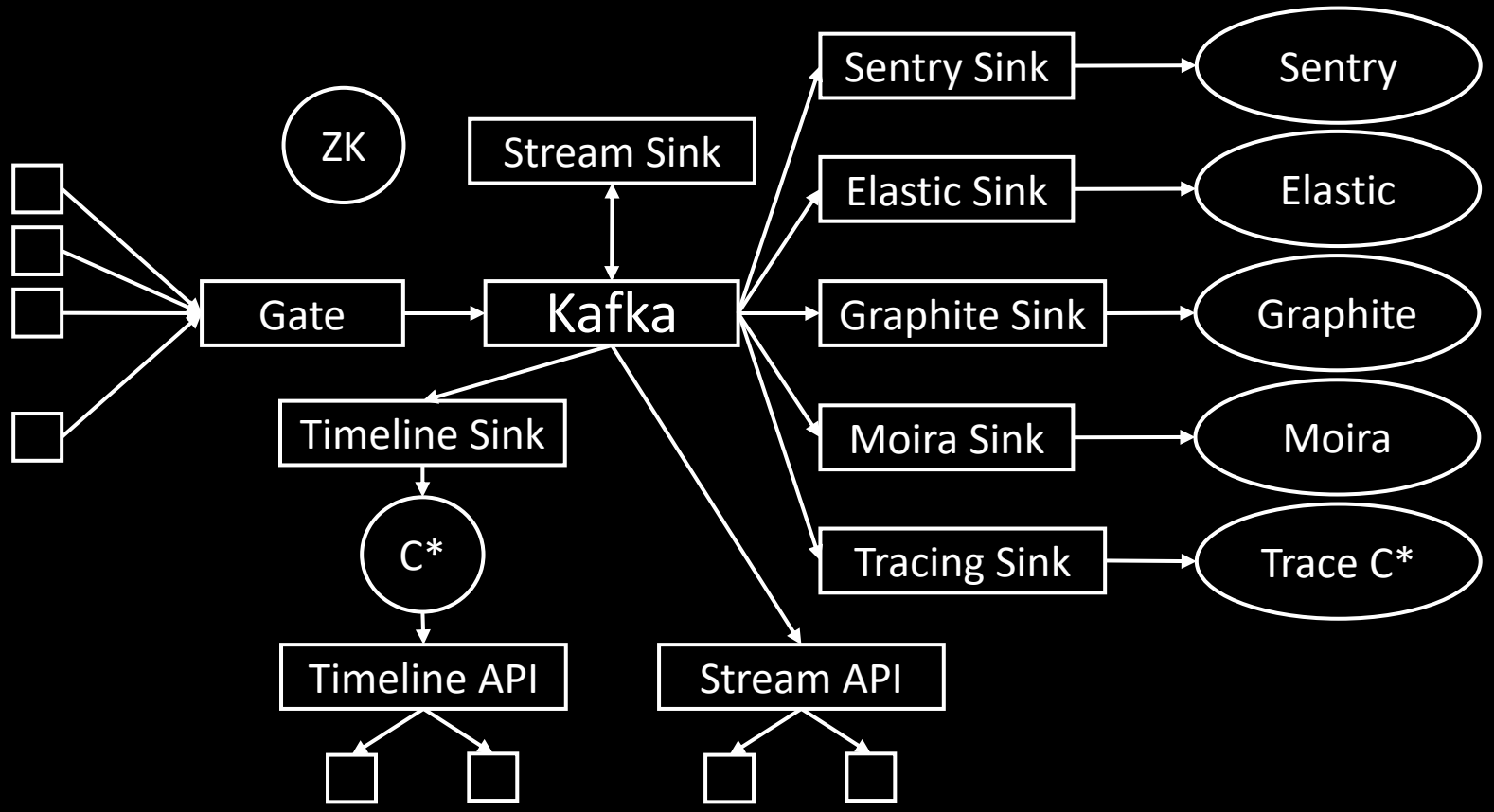
Исходники на GitHub

https://github.com/vostok

https://github.com/vostok/hercules

Make telemetry great again!

# Hercules under the hood




https://github.com/vostok/hercules