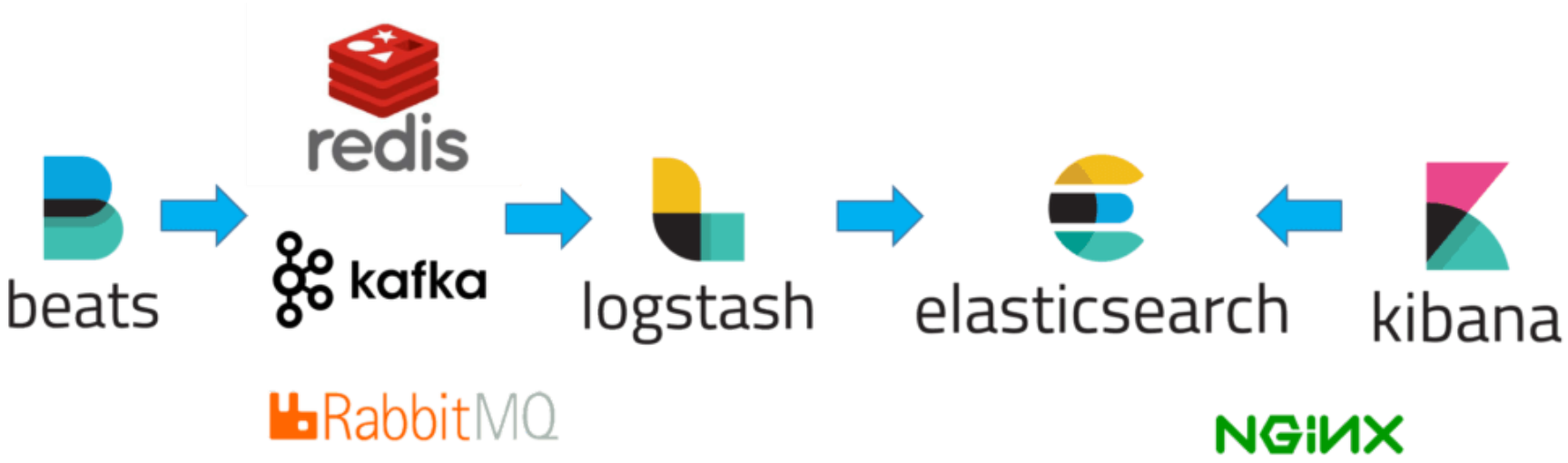
Сервисы централизованного логирования для компонентов Kubernetes и приложений



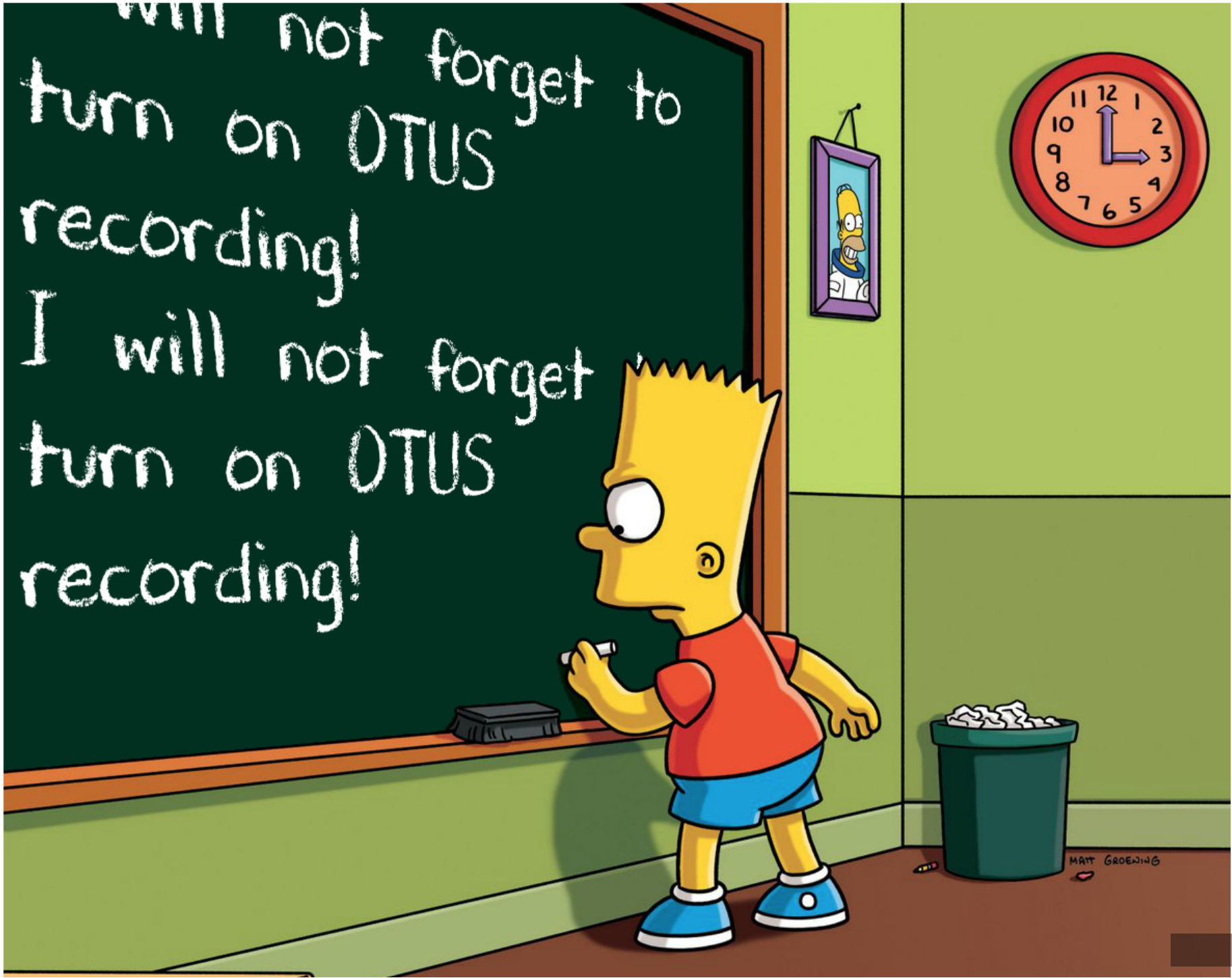
Data Collection	Buffering	Data Aggregation & Processing	Indexing & storage	Analysis & visualization

Меня хорошо слышно && видно?

Напишите в чат, если есть проблемы!

Ставьте + если все хорошо

Не забудь включить запись!


not forget to
turn on OTUS
recording!
I will not forget
turn on OTUS
recording!
MATT GROENING

ELK

Что такое логи

Log = Metadata + Timestamp + Data

Какие логи бывают

- основной файл лога
- лог загрузки системы
- логи веб-сервера
- логи сервера баз данных
- аудит
- логи планировщика задач
- логи компонентов kubernetes

Когда есть логи мы

- понимаем, как работают системы
- можем найти ошибки и их причины
- имеем контекст для метрик событий или процессов
- знаем, что происходит у наших клиентов

Логи используются для:

- Устранение неполадок.
- Понимание поведения системы/приложения
- Аудит
- Предиктивная аналитика
- Сбор бизнес-метрик

Некоторые особенности, которые необходимо учитывать

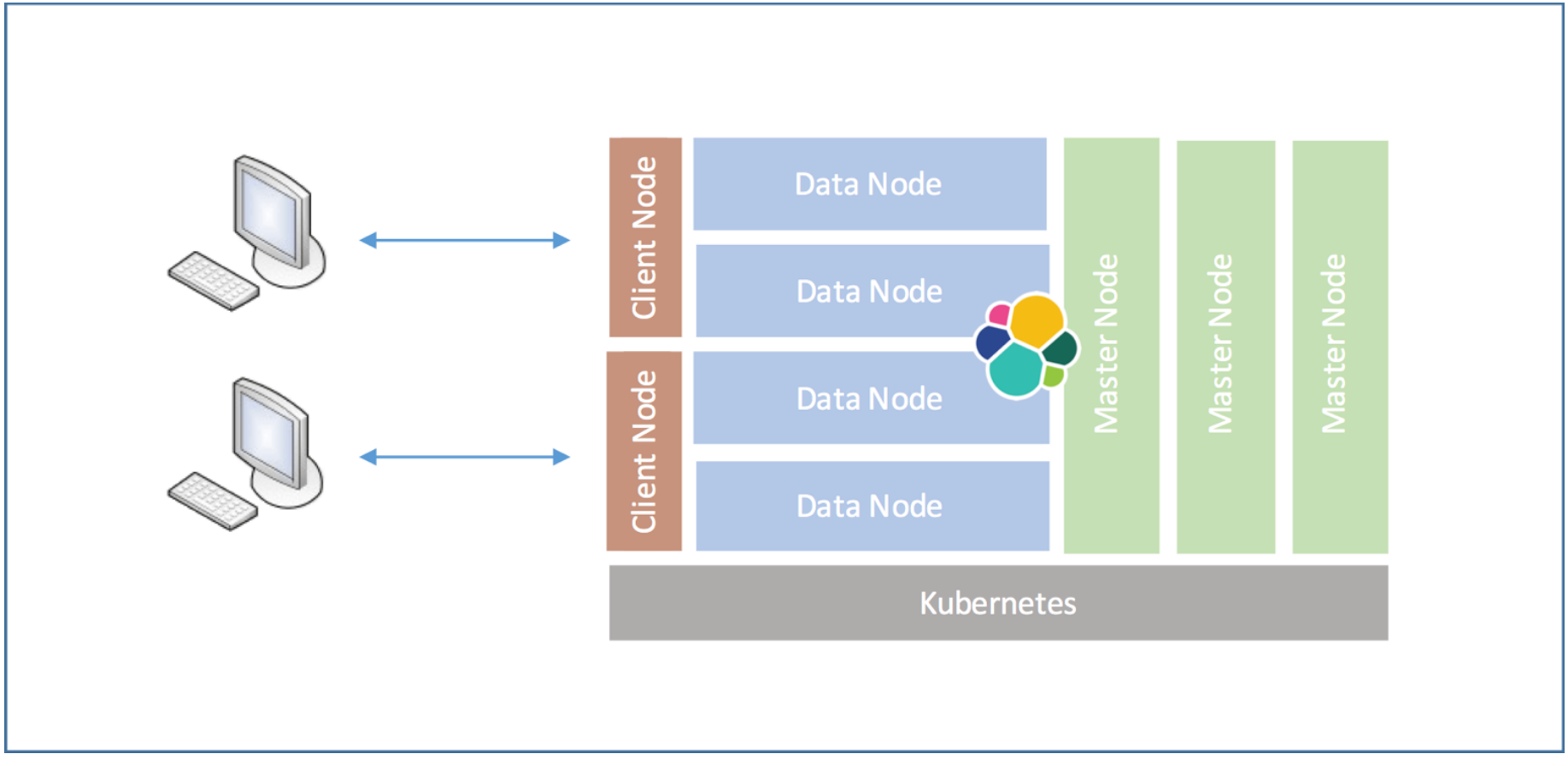
- Нестандартный/несовместимый формат
- Децентрализованные логи
- Несовместимый формат времени
- Неструктурированные данные

Какие ещё есть проблемы и задачи при анализе логов

- В kubernetes мы не можем хранить логи локально
- Инстансы могут быть ограничены жизненным циклом
- Командное взаимодействие при отслеживании проблемы
- kubectl logs
- Ротация логов

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

Централизованная система логирования



Elasticsearch

Основные понятия Elasticsearch.

- индекс;
- тип;
- документ;
- кластер;
- узел;
- шарды и копии;
- разметку и типы данных;
- обратный индекс.

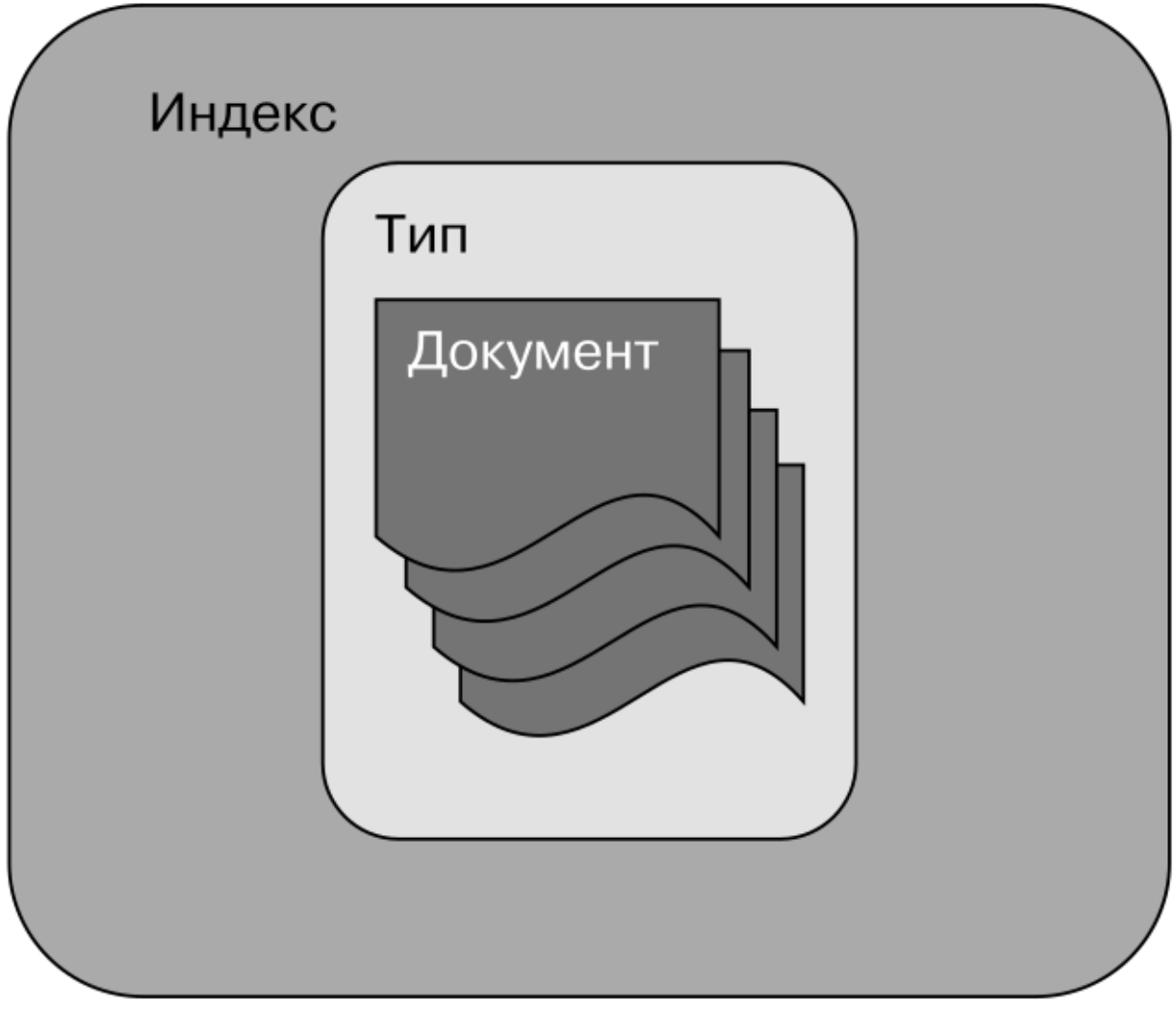
Index, type, document

```
PUT /catalog/product/1
{
    "sku": "SP000001",
    "title": "Elasticsearch",
    "description": "Elasticsearch for kubernetes",
    "author": "Otus",
    "ISBN": "1785288997",
    "price": 26.99
}
```

Index, type, document

```
curl -XPUT http://localhost:9200/catalog/product/1 -d '{ "sku": "SP000001", "title":
"Elasticsearch", "description": "Elasticsearch for kubernetes", "author": "Otus",
"ISBN": "1785288997", "price": 26.99}'
```

Index



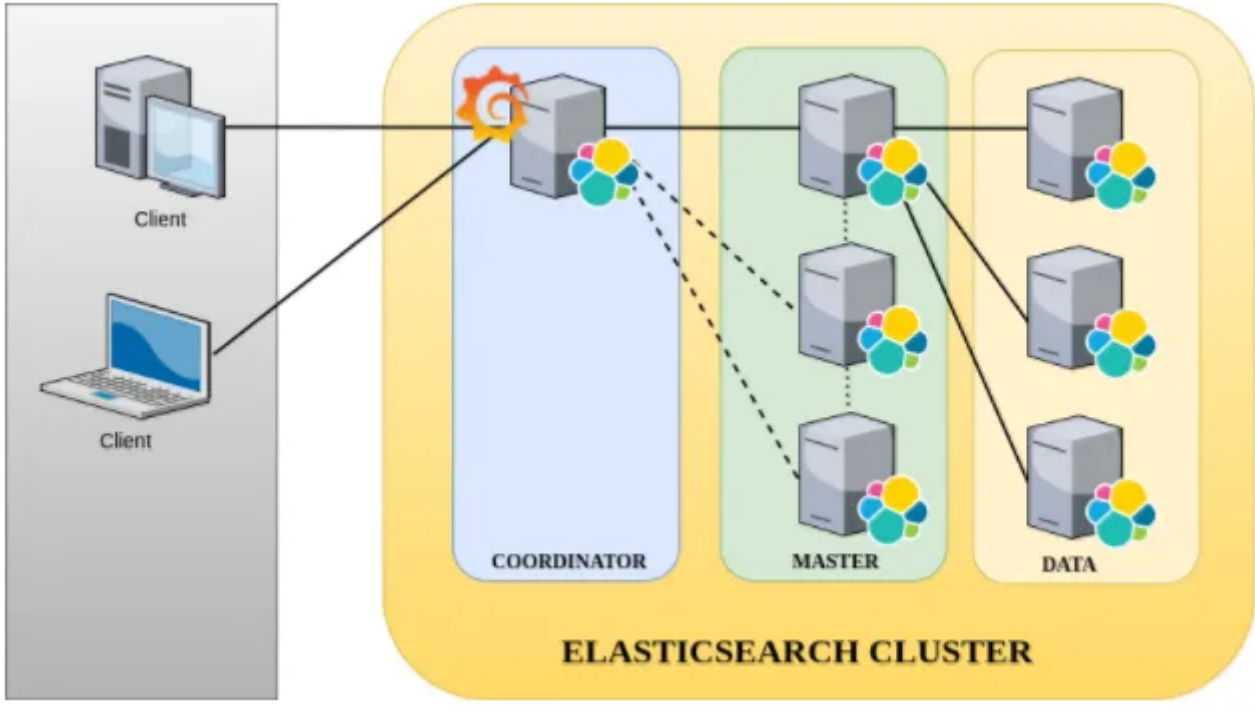
Тип

```
{
    "sku": "SP000001",
    "title": "Elasticsearch",
    "description": "Elasticsearch for kubernetes",
    "author": "Otus",
    "ISBN": "1785288997",
    "price": 26.99
}
```

Документ

- '_id' — уникальный идентификатор документа внутри типа по аналогии с пер- вичным ключом в таблице базы данных. Он может генерироваться автомати- чески или выбираться пользователем;
- '_type' — это поле содержит тип документа;
- '_index'— хранит имя индекса документа.

Кластер


Client
Client
COORDINATOR
MASTER
DATA
ELASTICSEARCH CLUSTER

Zen Discovery

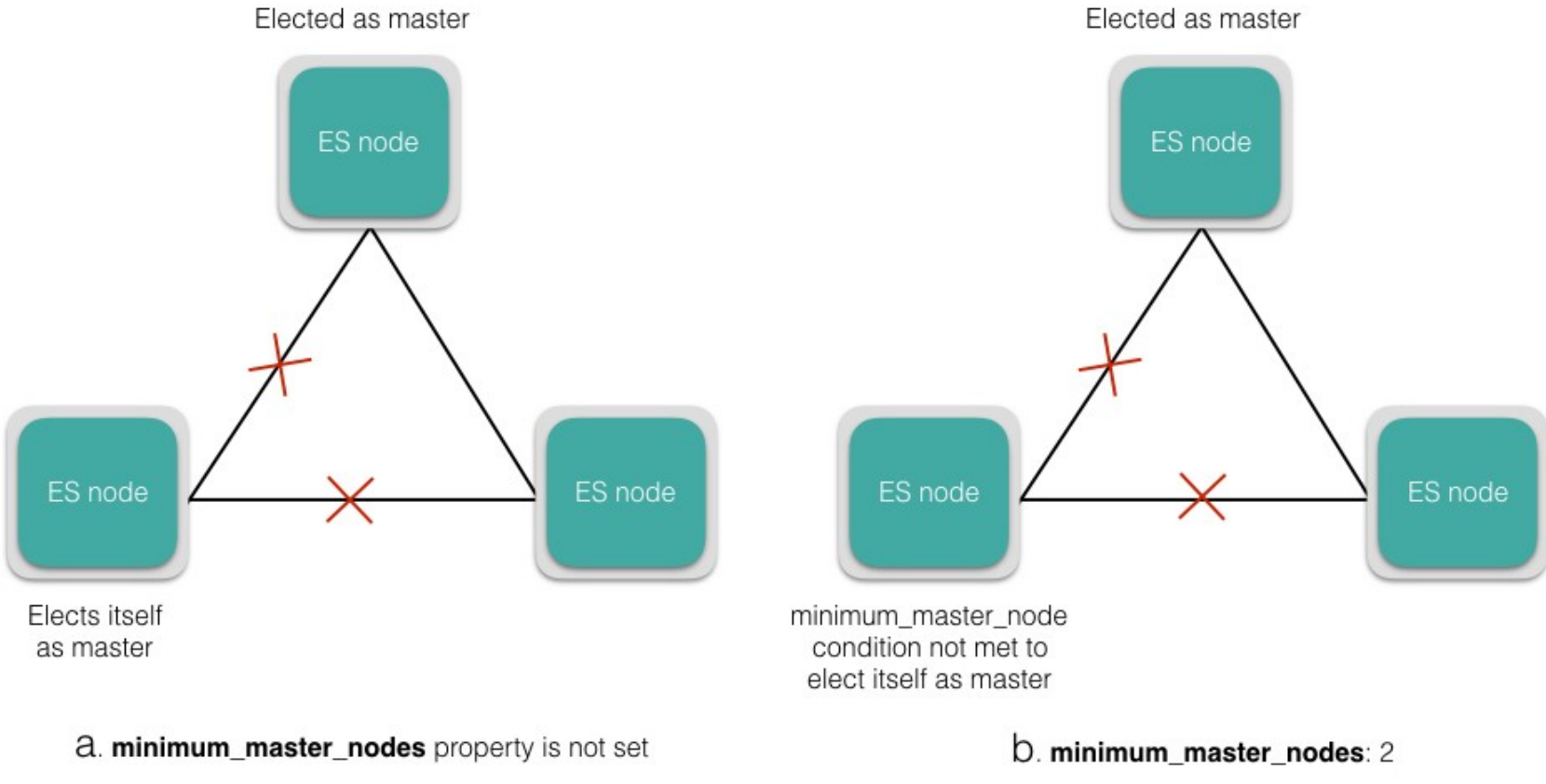
- Ping
- Unicast
- Master Election
- Fault Detection
- Cluster State Updates
- No Master Block

https://www.elastic.co/guide/en/elasticsearch/reference/6.2/modules-discovery-zen.html

Master Election

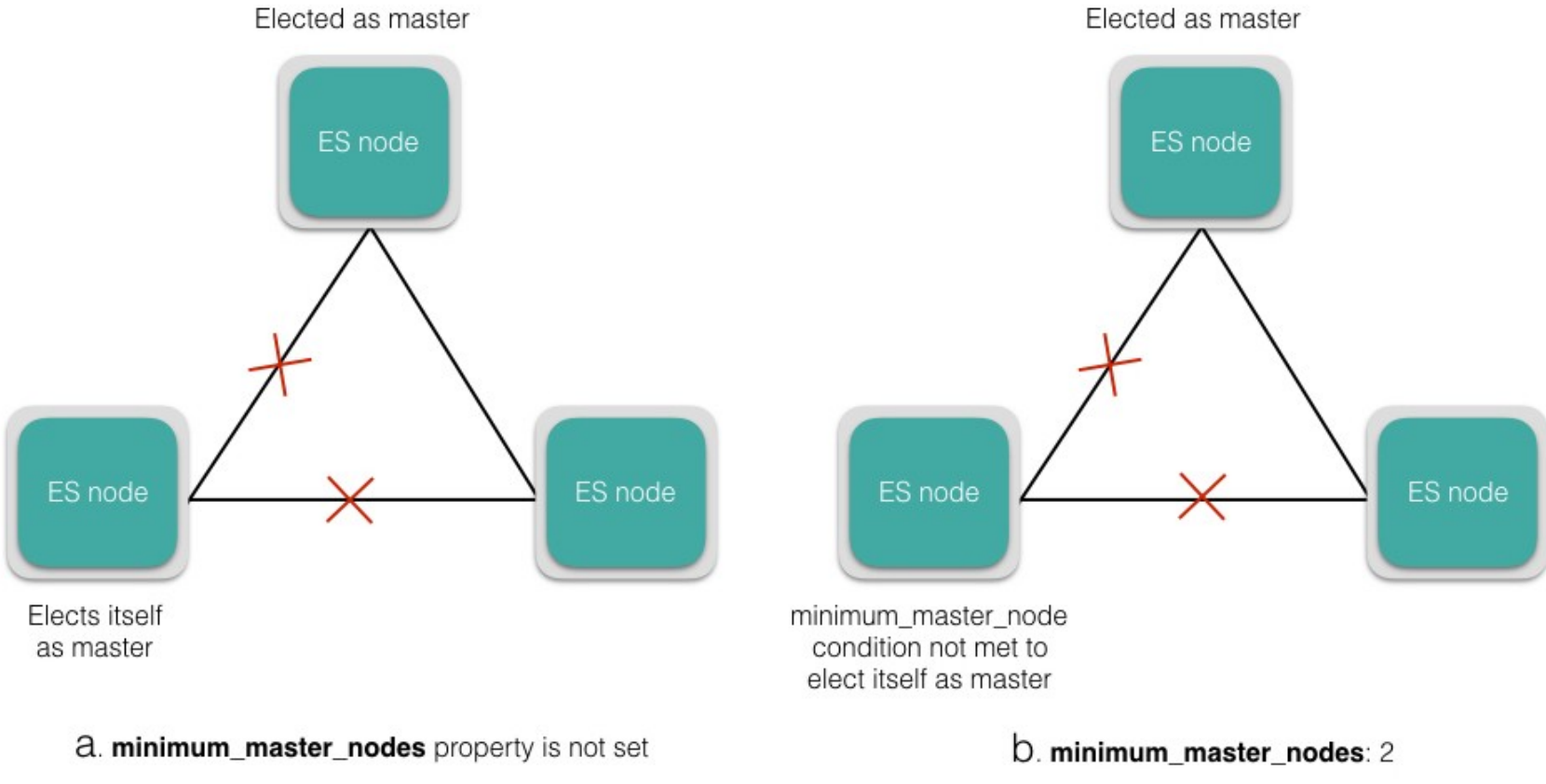
- discovery.zen.ping_timeout (default 3s)
- discovery.zen.join_timeout (default 20xping_time)
- discovery.zen.master_election.ignore_non_master_pings and node.master
- discovery.zen.minimum_master_nodes

Master node



- discovery.zen.minimum_master_nodes

No master block



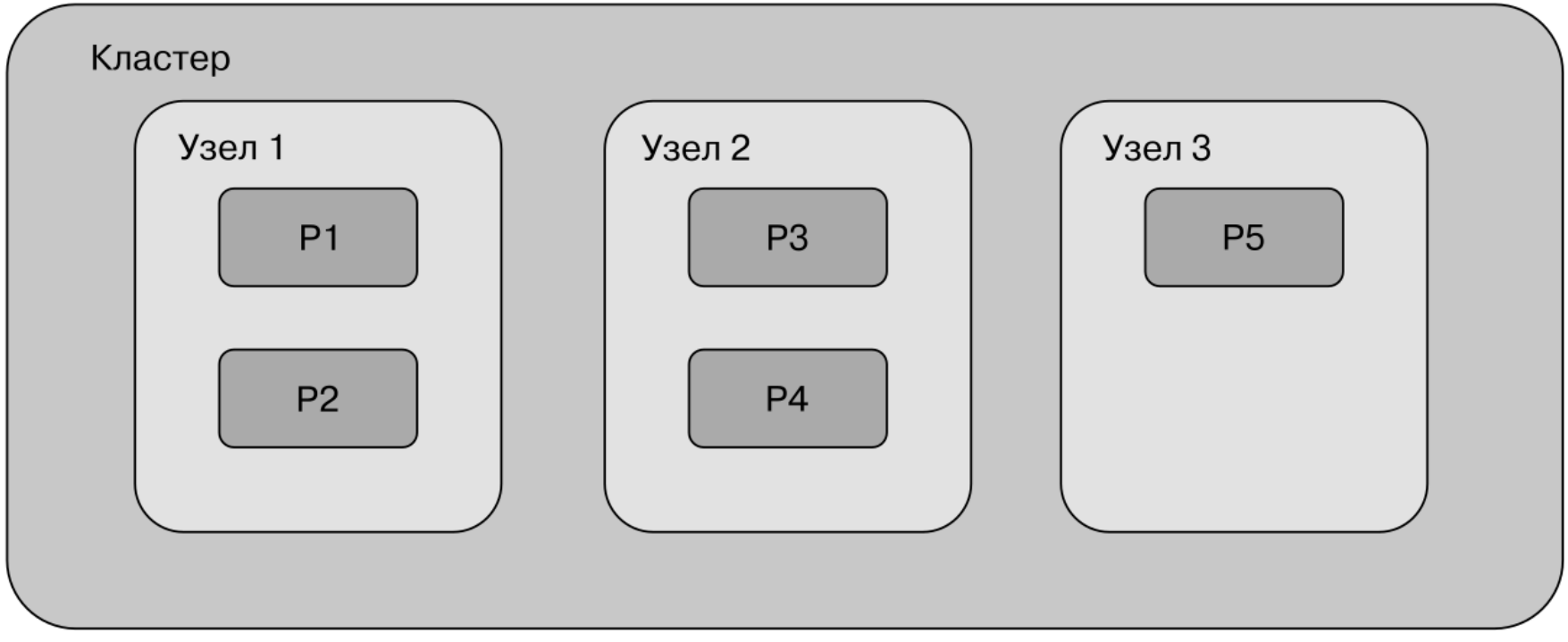
- discovery.zen.minimum_master_nodes
- discovery.zen.no_master_block

Шарды и копии

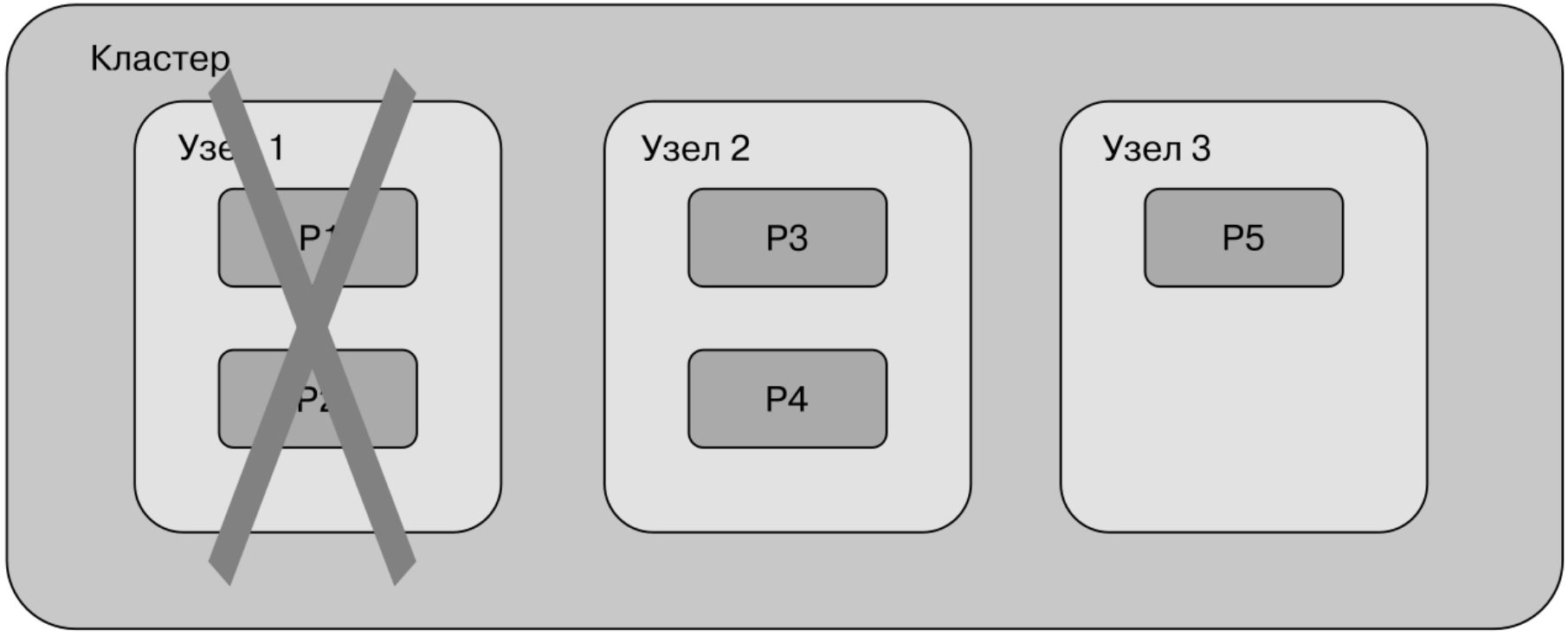
Процесс разделения данных по шардам называется шардированием. Это неотъемлемая часть Elasticsearch, необходимая для масштабируемой и параллельной работы с выполнением оптимизации:

- дискового пространства по разным узлам кластера;
- вычислительной мощности по разным узлам кластера.

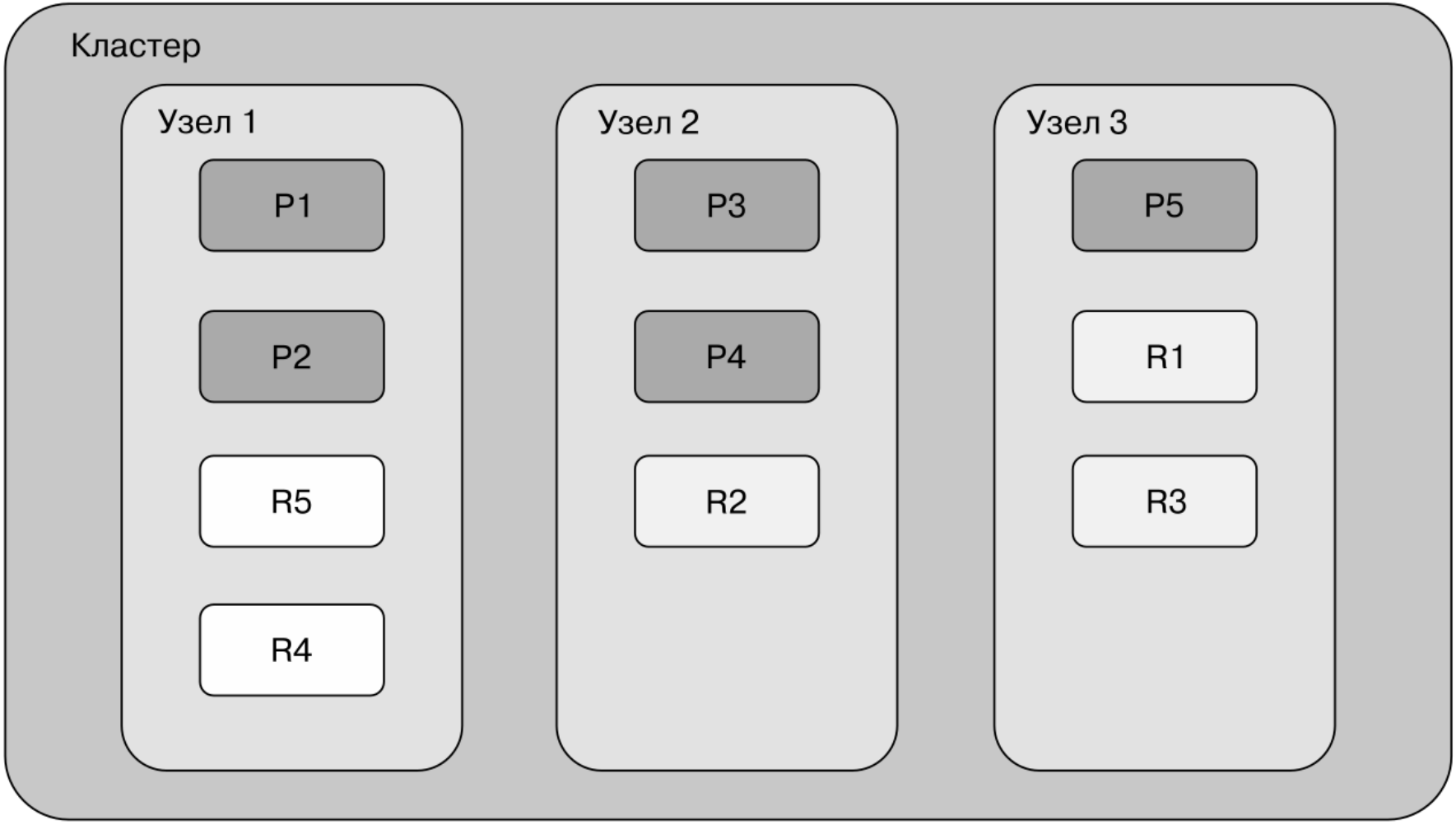
Шарды...



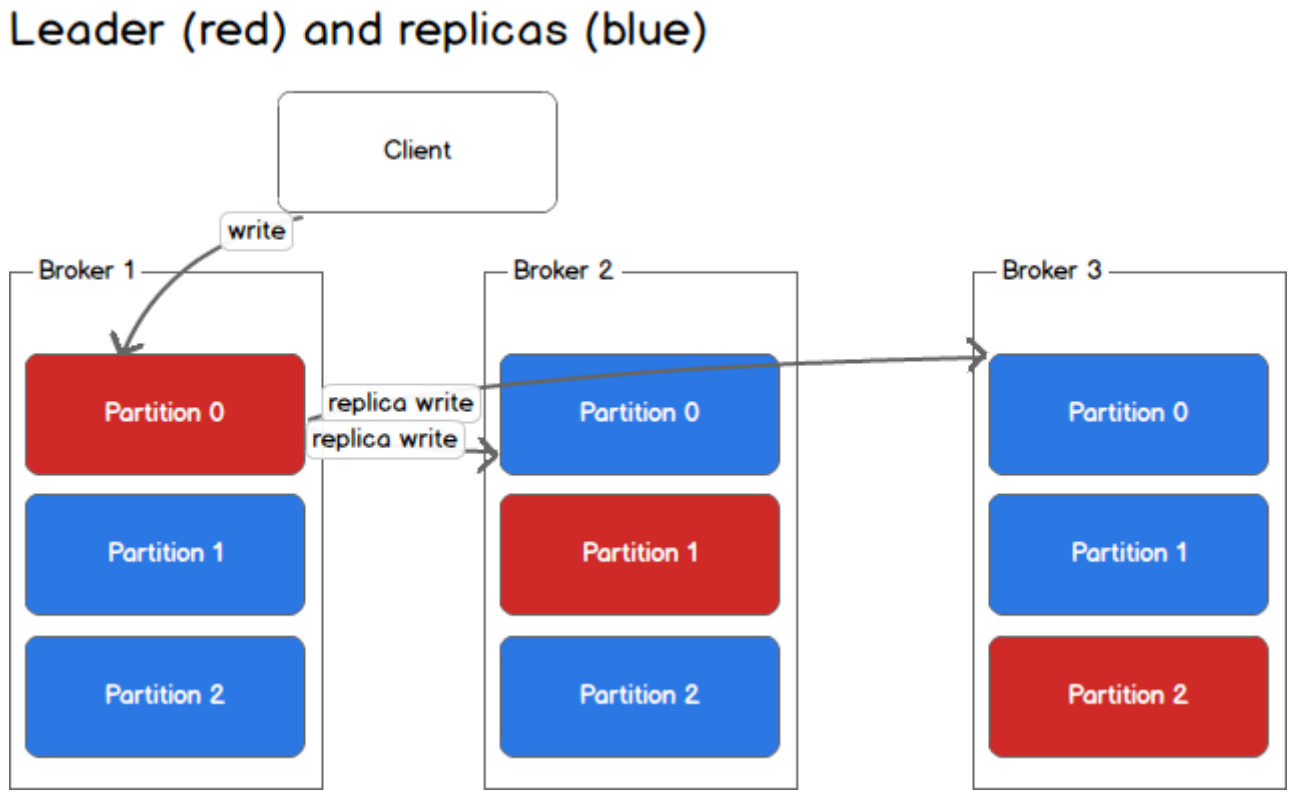
Что-то может пойти не так



...и копии



Ничего не напоминает? :)



Типы данных

- Text
- Keyword
- Date
- Numeric
- Boolean
- Binary
- Массив
- Объект
- Вложенный тип

Основные типы данных | Строковые:

- `text` — полезен для полнотекстового поиска по полям, которые содержат длинные текстовые значения. Эти поля анализируются до индексирования для поддержки полнотекстового поиска;
- `keyword` — позволяет выполнять анализ строковых полей. Поля такого типа поддерживают сортировку, фильтрацию и агрегацию.

Основные типы данных | Числовые:

- byte , short , integer и long
- float и double
- half-float
- scaled_float

Основные типы данных

- Даты: date .
- Булевы типы: boolean .
- Бинарный тип данных: binary .
- Диапазон: integer_range , float_range , long_range , double_range и data_range .

Хозяйке на заметку

- scaled_float — очень полезный тип данных для хранения таких значений, как цены, которые всегда имеют ограниченное количество знаков после десятичной точки. Цена может указываться с коэффициентоммасштабирования 100, следовательно, ценник $10.98 может храниться как 1098 центов и обрабатываться как целое число. Тип scaled_float гораздо эффективнее для хранения, чем другие типы, поскольку целые числа лучше сжимаются.

Комплексные типы данных

- Массив — определяет массивы одного типа. Например, массивы строковых данных, массивы целых чисел и т. д. В массивах не разрешается смешивать типы данных.
- Объект — допускает использование внутренних объектов в документах JSON.
- Вложенный тип данных — каждый объект в массиве индексируется как новый вложенный документ. Поскольку объекты обрабатываются внутри как отдель- ные документы, следует использовать специальный тип запроса для запроса вложенных документов.

Другие типы данных

- Геоточка — делает возможным хранение геоточек по широте и долготе. Напри- мер, тип данных «геоточка» полезен для таких запросов, как поиск по всем банкоматам в радиусе 2 км от указанной точки.
- Геоформа — используется для хранения геометрических форм, таких как по- лигоны, карты и пр. Тип данных «геоформа» позволяет выполнять запросы по поиску всех предметов определенной формы.
- Тип данных IP — используется для хранения адресов IPv4 и IPv6.

Разметка

```
PUT /catalog/product/2
{
    "sku": "SP000002",
    "title": "Google Pixel Phone 32GB - 5 inch display",
    "description": "Google Pixel Phone 32GB - 5 inch display (Factory Unlocked US
Version)",
    "price": 400.00,
    "resolution": "1440 x 2560 pixels",
    "os": "Android 7.1"
}
```

После того как первый документ типа «продукт» проиндексирован в каталоге индексов, Elasticsearch выполняет следующие действия:

- создает индекс с именем каталога;
- обозначает разметку для типа продукта.

Принцип работы разметки

- Создание индекса
- Установка разметки (mapping) для типа продукта

```
GET /catalog/_mapping/product
```

https://www.elastic.co/guide/en/elasticsearch/reference/current/indices.h
https://www.elastic.co/guide/en/elasticsearch/reference/current/indices-get-mapping.html

Вывод ответа на запрос

```
{
"catalog": {                    # индекс, для которого была запрошена разметка
  "mappings": {
    "product": {                # эта разметка – для типа продуктов
      "properties": {
        "ISBN": {               # поле, установленное строкой, получило тип данных text
          "type": "text"
        }
      },
      "author": {
        "type": "text"
      }
    },
    "description": {
      "type": "text"
    }
  }
},
  "price": {
    "type": "float"             # float задан только для цены
  },
    "sku": {
      "type": "text"
    },
  "title": {
      "type": "text"
```

```
    }
}
```

Обратный индекс

Идентификатор документа	Документ
1	Завтра воскресенье
2	Воскресенье — последний день недели
3	Выбор за вами

Обратный индекс

Терм (определение)	Частота	Документы (постинги)
вами	1	3
воскресенье	2	1, 2
выбор	1	3
день	1	2
завтра	1	1
неделя	1	2
последний	1	2

Обратите внимание на следующее.

- Документы разбиваются на определения без учета пунктуации и прописных букв.
- Термы сортируются по алфавиту.
- Столбец «Частота» показывает, как часто данное определение встречается во всем наборе документов.
- Третий столбец показывает, в каких документах было найдено определение. Кроме того, там могут содержаться точные места нахождения (офсеты внутри документа).

Обратный индекс

Терм (определение)	Частота	Документы (постинги)
вами	1	3
воскресенье	2	1, 2
выбор	1	3
день	1	2
завтра	1	1
неделя	1	2
последний	1	2

Когда пользователь ищет два слова, например «последнее воскресенье». Обратный индекс может быть настроен на поиск отдельно слов «последнее» и «воскресенье». Документ 2 содержит оба определения, следовательно, это более подходящее соответствие, чем документ 1, содержащий только одно определение.

Elasticsratch API

Операции CRUD

CRUD - create, read, update, delete

- Index API;
- Get API;
- Update API;
- Delete API.

Index API

- индексирование с предоставлением идентификатора;
- индексирование без предоставления идентификатора.

Индексирование документа с предоставлением идентификатора

```
PUT /<index>/<type>/<id>
```

```
PUT /catalog/product/1
{
    "sku": "SP000001",
    "title": "Elasticsearch",
    "description": "Elasticsearch for kubernetes",
    "author": "Otus",
    "ISBN": "1785288997",
    "price": 26.99
}
```

Индексирование документа без предоставления идентификатора

```
POST /catalog/product
{
    "sku": "SP000003",
    "title": "Logstash",
    "description": "Logstash for kubernetes",
    "author": "Otus",
    "price": 54.99
}
```

Индексирование документа без предоставления идентификатора

```
{
    "_index": "catalog",
    "_type": "product",
    "_id": "AVrASKqgaBGmnAMj1SBe",
    "_version": 1,
    "result": "created",
    "_shards": {
        "total": 2,
        "successful": 1,
        "failed": 0
    },
    "created": true
}
```

Хозяйке на заметку

По соглашениям REST POST используется для создания нового ресурса, а PUT — для обновления существующего ресурса. В нашем случае использование PUT означает следующее: «Я знаю, какой идентификатор хочу присвоить, следовательно, я указываю его в процессе индексирования документа».

Get API

```
GET /catalog/product/AVrASKqgaBGmnAMj1SBe
```

```
GET /<index>/<type>/<id>
```

```
{
    "_index": "catalog",
    "_type": "product",
    "_id": "AVrASKqgaBGmnAMj1SBe",
    "_version": 1,
    "found": true,
    "_source": {
        "sku": "SP000003",
        "title": "Logstash",
        "description": "Logstash for kubernetes",
        "author": "Otus",
        "price": 54.99
    }
}
```

Update API

```
POST /catalog/product/1/_update
{
    "doc": {
        "price": "28.99"
    }
}
```

Update API используется для обновления существующего идентификатора документа.

doc_as_upsert

```
POST /catalog/product/3/_update
{
    "doc": {
        "author": "Otus",
        "title": "Beats",
        "description": "Beats for kubernetes",
        "price": "54.99"
    },
    "doc_as_upsert": true
}
```

Параметр doc_as_upsert проверяет, существует ли документ с предоставленным идентификатором и объединяет предоставленный элемент doc с существующим документом. Если документ с таким идентификатором не существует, будет вставлен новый документ с необходимым содержимым.

Delete API

```
DELETE /catalog/product/AVrASKqgaBGmnAMj1SBe
```

```
{

    "found": true,
    "_index": "catalog",
    "_type": "product",
    "_id": "AVrASKqgaBGmnAMj1SBe",
    "_version": 4,
    "result": "deleted",
    "_shards": {
        "total": 2,
        "successful": 1,
        "failed": 0,
    }
}
```

Создание индексов и контролирование разметки

- создание индекса;
- создание разметки;
- обновление разметки.

Создание индекса

```
PUT /catalog
{
    "settings": {
        "index": {
            "number_of_shards": 5,
            "number_of_replicas": 2
        }
    }
}
```

Указание разметки для типа во время создания индекса

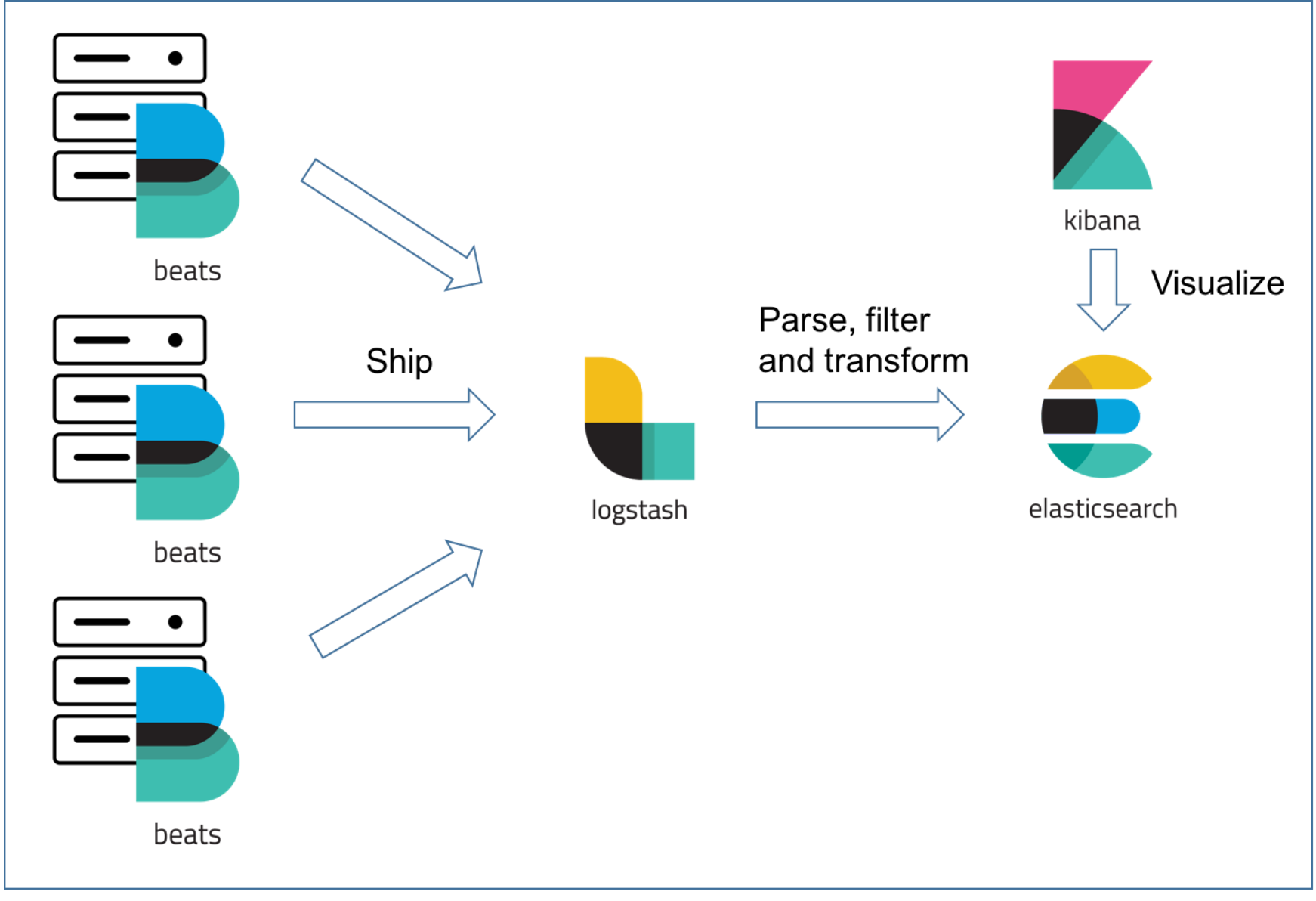
```
PUT /catalog
{
    "settings": {
        "index": {
            "number_of_shards": 5,
            "number_of_replicas": 2
        }
    },
    "mappings": {
        "my_type": {
            "properties": {
                "f1": {
                  "type": "text"
                },
                "f2": {
                  "type": "keyword"
                }
            }
        }
    }
}
```

Создание разметки типов в существующем индексе

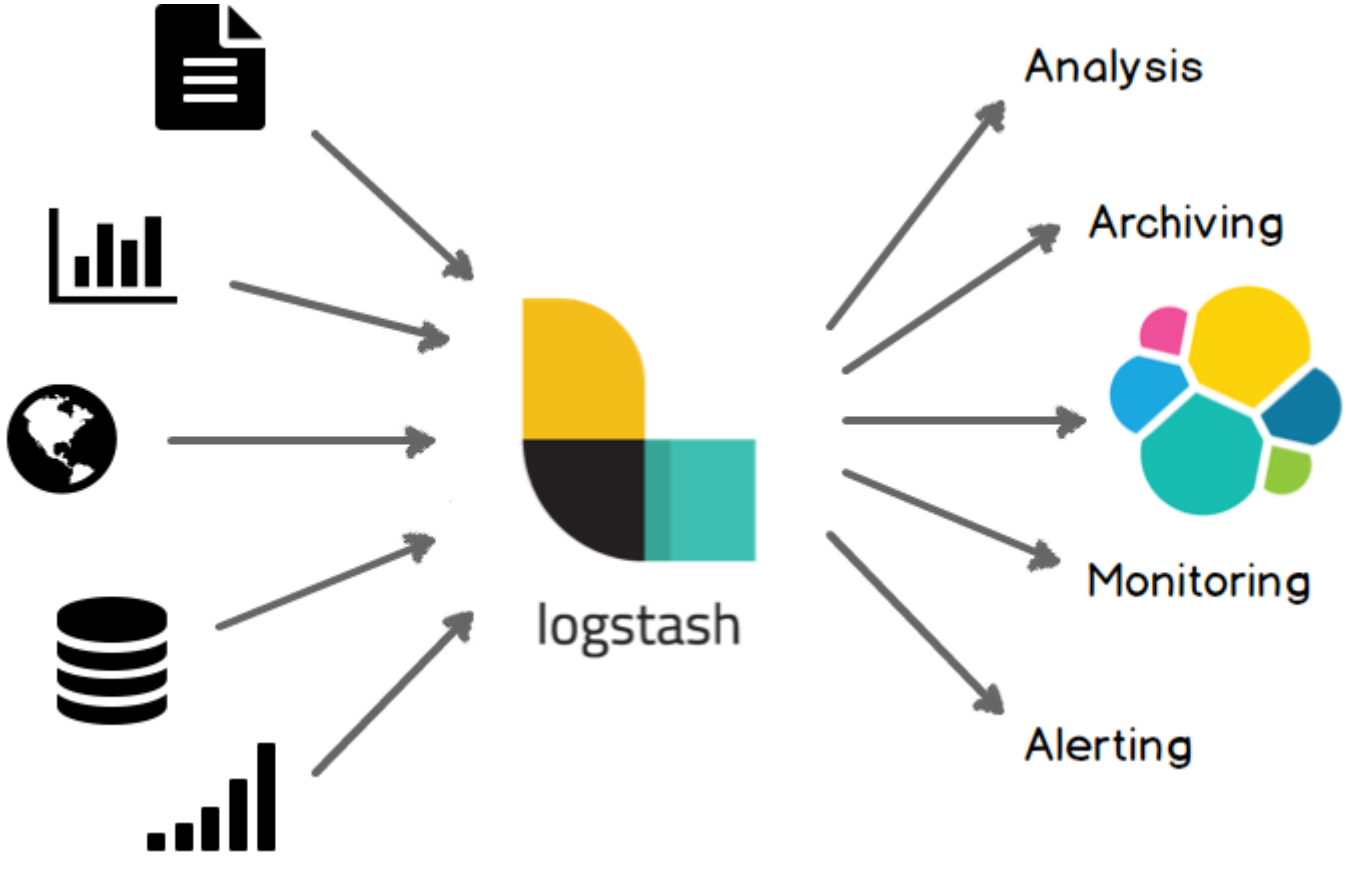
```
PUT /catalog/_mapping/category
{
    "properties": {
        "name": {
          "type": "text"
        }
    }
}

POST /catalog/category
{
  "name": "books"
}

POST /catalog/category
{
  "name": "phones"
}
```

Logstash

Анализ журнальных данных


beats
beats
beats
Ship
logstash
Parse, filter
and transform
kibana
Visualize
elasticsearch

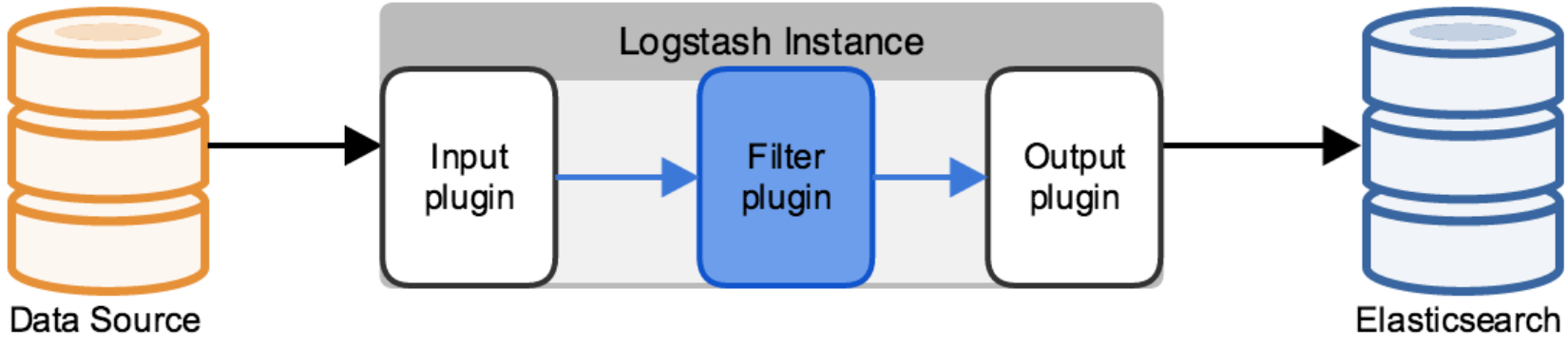
Logstash



Logstash

- Гибкая архитектура, основанная на плагинах
- Расширяемость
- Централизованная обработка данных
- Универсальность
- Совместная работа

Архитектура Logstash



Пример конфигурации Logstash

```
#A simple logstash configuration
input {
    stdin { }
}
filter {
    mutate {
        uppercase => [ "message" ]
    }
}
output {
    stdout {
        codec => rubydebug
    }
}
```

Обзор плагинов Logstash

```
bin/logstash-plugin list --verbose
bin/logstash-plugin list --group filter
bin/logstash-plugin list 'pager'
bin/logstash-plugin install logstash-output-email
bin/logstash-plugin update logstash-output-s3
```

Плагины ввода

logstash-input-beats	logstash-input-couch-db_changes	logstash-input-elastic-search	logstash-input-ganglia
logstash-input-xmpp	logstash-input-unix	logstash-input-syslog	logstash-input-stdin
logstash-input-udp	logstash-input-twitter	logstash-input-tcp	logstash-input-sqs
logstash-input-snmp-trap	logstash-input-redis	logstash-input-pipe	logstash-input-log4j
logstash-input-s3	logstash-input-rabbitmq	logstash-input-lum-berjack	logstash-input-http_ poller
logstash-input-exec	logstash-input-file	logstash-input-http	logstash-input-imap
logstash-input-gelf	logstash-input-jdbc	logstash-input-irc	logstash-input-gener-ator
logstash-input-heart-beat	logstash-input-graphite		

https://www.elastic.co/guide/en/logstash/7.3/input-plugins.html

Обзор плагинов ввода | File

- File (<path.data>/plugins/inputs/file)

```
input
{ file{
    path => ["/usr/local/logfiles/*","/var/log/*.txt"
    start_position => "beginning"
    exclude => ["*.csv]
    discover_interval => "10s"
    type => "applogs"
    }
}
output
{
    stdout {
        codec => rubydebug
    }
}
```

Обзор плагинов ввода | Beats

```
input {
    beats {
        host => "192.168.10.229"
        port => 1234
    }
    beats {
        host => "192.168.10.230"
        port => 5065
    }
}
output {
    elasticsearch {
    }
}
```

Плагины вывода

logstash-output-cloud-watch	logstash-output-nagios	logstash-output-irc	logstash-output-pagerduty
logstash-output-xmpp	logstash-output-tcp	logstash-output-stdout	logstash-output-redis
logstash-output-webhdfs	logstash-output-statsd	logstash-output-sns	logstash-output-rabbitmq
logstash-output-udp	logstash-output-sqs	logstash-output-s3	logstash-output-pipe
logstash-output-csv	logstash-output-graphite	logstash-output-file	logstash-output-elasticsearch
logstash-output-http			

https://www.elastic.co/guide/en/logstash/7.3/output-plugins.html

Плагины вывода | Elasticsearch

```
input {
    stdin{
    }
}
output {
    elasticsearch {
        index => "company"
        document_type => "employee"
        hosts => "198.162.43.30:9200"
        user => "elastic"
        password => "elasticpassword"
    }
}
```

Плагины вывода | Kafka

```
input {
    stdin{
    }
}
output {
    kafka {
        bootstrap_servers => "localhost:9092"
        topic_id => 'logstash'
    }
}
```

Плагины вывода | PagerDuty

```
input {
    elasticsearch {
        hosts => "localhost:9200"
        index => "ngnixlogs"
        query => '{ "query": { "match": { "statuscode": 404} }}'
    }
}
output {
    pagerduty {
        service_key => "service_api_key"
        details => {
            "timestamp" => "%{[@timestamp]}"
            "message" => "Problem found: %{[message]}"
        }
        event_type => "trigger"
    }
}
```

Плагины кодеков | JSON

```
input{
    stdin{
        codec => "json"
    }
}
```

Плагины кодеков | Rubydebug

```
output{
    stdout{
        codec => "rubydebug"
    }
}
```

Плагины кодеков | Multiline

```
input {
    file {
        path => "/var/log/access.log"
        codec => multiline {
            pattern => "^\s "
            negate => false
            what => "previous"
        }
    }
}
```

Плагины фильтров

logstash-filter-cidr	logstash-filter-clone	logstash-filter-grok	logstash-filter-geoip
logstash-filter-date	logstash-filter-csv	logstash-filter-throttle	logstash-filter-xml
logstash-filter-finger-print	logstash-filter-dns	logstash-filter-drop	logstash-filter-dissect
logstash-filter-sys-log_pri	logstash-filter-useragent	logstash-filter-split	logstash-filter-trans-late
logstash-filter-uuid	logstash-filter-urlde-code	logstash-filter-sleep	logstash-filter-ruby
logstash-filter-mutate	logstash-filter-metrics	logstash-filter-kv	logstash-filter-json

https://www.elastic.co/guide/en/logstash/7.3/filter-plugins.html

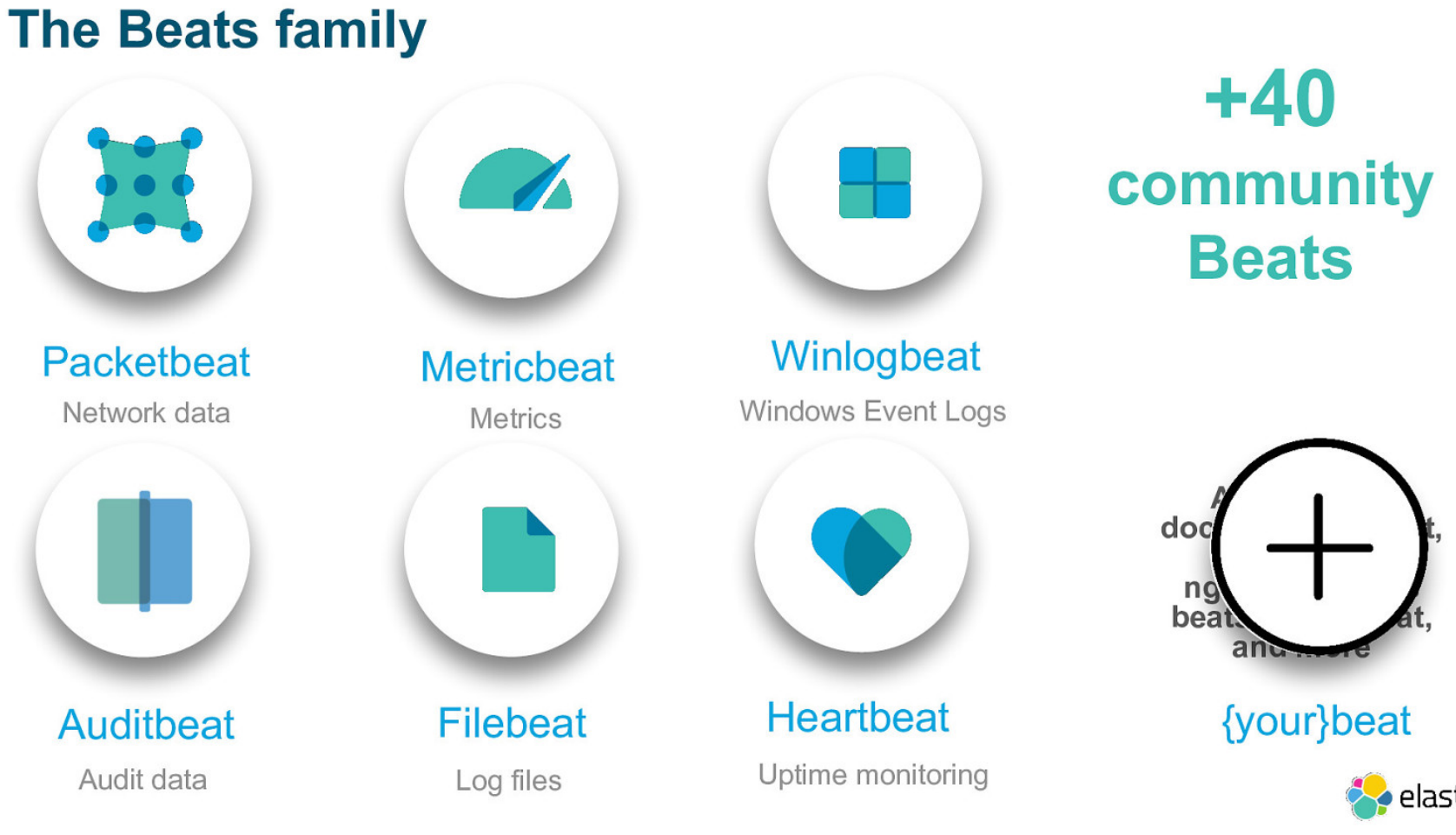
Фильтр Mutate

```
input {
    file{
        path => "/var/logs/users.csv"
        start_position => "beginning"
        sincedb_path => "NULL"
    }
}
filter {
    csv{
        autodetect_column_names => true
    }
    mutate {
        convert => {
            "Age" => "integer"
            "Salary" => "float"
    }
    rename => { "FName" => "Firstname"
                "LName" => "Lastname" }
    gsub => [
        "EmailId", "\.", "_"
    ]
    strip => ["Firstname", "Lastname"]
    uppercase => [ "Gender" ]
    }
}
output {
    stdout {
```

```
        codec => rubydebug
    }
}
```

Beats

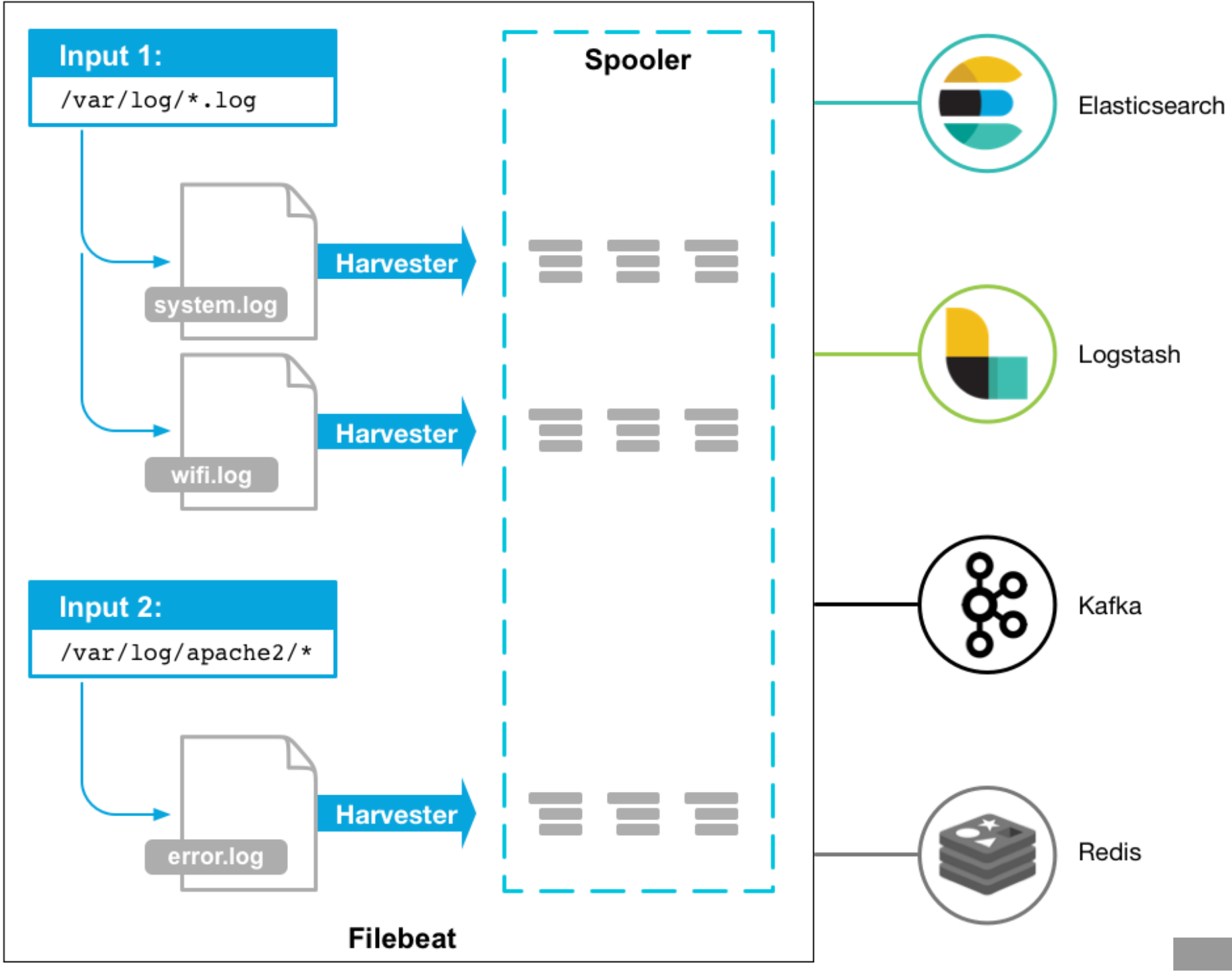
Beats



- Фреймворк Beats это легковесные компоненты доставки данных. Они устанавливаются как агенты на пограничные серверы для

передачи операционных данных в Elasticsearch.

Архитектура


Input 1:
/var/log/*.log
system.log
wifi.log
Harvester
Harvester
Input 2:
/var/log/apache2/*
error.log
Harvester
Spooler
Filebeat
Elasticsearch
Logstash
Kafka
Redis

Настройка Filebeat

- prospectors;
- глобальные настройки;
- общие настройки;
- конфигурация вывода;
- конфигурация обработки;
- конфигурация папок;
- конфигурация модулей;
- конфигурация панели управления;
- конфигурация сбора данных.

Проверка доставки логов в Elasticsearch

```
GET /metricbeat*/_search?pretty
```

Prospectors

- input_type
- paths
- exclude_lines
- include_lines
- tags
- fields
- scan_frequency
- document_type
- multiline

Модули Filebeat

- Конфигурации старателя Filebeat
- Определение контейнера поглощения Elasticsearch для обработки и дополнения логов
- Шаблоны Elasticsearch с определениями полей, используемых с целью корректной разметки полей для событий
- Примеры панелей управления Kibana, которые могут быть задействованы для визуализации логов

Плагины кодеков

logstash-codec-netflow	logstash-codec-cef	logstash-codec-es_bulk	logstash-codec-dots
logstash-codec-collectd	logstash-codec-mul-tiline	logstash-codec-msgpack	logstash-codec-line
logstash-codec-rubydebug	logstash-codec-json	logstash-codec-json_lines	logstash-codec-fluent
logstash-codec-plain	logstash-codec-graphite	logstash-codec-edn_lines	logstash-codec-edn

https://www.elastic.co/guide/en/logstash/7.3/codec-plugins.html

Ingest API

Узел поглощения данных

```
POST my_index/my_type?pipeline=my_pipeline_id
{
    "key": "value"
}
```

- Функция узла поглощения данных является легковесным решением для предварительной обработки и дополнения документов в Elasticsearch до индексирования

Определение контейнера

```
{
    "description" : "...",          # необязательное поле, предназначено для хранения
информации
    "processors" : [ ... ]          # список процессоров для трансформации документа
}
```

- Узел поглощения имеет около 20 встроенных процессоров, включая gsub, grok, convert, remove, rename

Ingest API | API определения контейнера

- _ingest

```
PUT _ingest/pipeline/<pipeline-name>
 {
   "pipeline": {
     "description": "pipeline-description",
     "processors": [] # принимают список processors, которые выполняются
последовательно.
   }
 }
```

```
curl -X PUT http://localhost:9200/_ingest/pipeline/firstpipeline -H 'content-type:
application/json'
-d '{
    "description" : "uppercase the incoming value in the message field",
    "processors" : [
        {
            "uppercase" : {
                "field": "message"
            }
        }
    ]
}'
```

API определения контейнера

- Пример записи лога с сервера NGINX

```
{
   "message": "89.169.145.79 - - 2019-09-11T09:02:34.234+03:00 service.hapicloud.io GET
/health HTTP/1.1 200 25 - Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML,
like Gecko) Chrome/76.0.3809.100 Safari/537.36 - 0.017 0.016 . 7353327 1"
}
```

Grok

- Grok - набор регулярных выражений для трансформации неструктурированных данных, таких как журналы, в структурированные данные

  %{SYNTAX:SEMANTIC}

  https://github.com/logstash-plugins/logstash-patterns-core/tree/master/patterns

Grok | Templates

```
89.169.145.79 - - 2019-09-11T09:02:34.234+03:00 service.hapicloud.io GET /health
HTTP/1.1 200 25 - Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/76.0.3809.100 Safari/537.36 - 0.017 0.016 . 7353327 1
```

```
%{IPORHOST:remote_ip} %{GREEDYDATA:dash} %{DATA:user_name} %
{TIMESTAMP_ISO8601:timestamp} %{DATA:referrer} %{WORD:http_method} %{DATA:url} HTTP/%
{NUMBER:http_version} %{NUMBER:response_code} %{NUMBER:body_sent_bytes} %{DATA:agent} -
%{BASE10NUM:request_time} %{BASE10NUM:upstream_response_time} %{GREEDYDATA:pipe} %
{NUMBER:connection} %{NUMBER:connection_requests}
```

https://grokdebug.herokuapp.com/patterns#

Тестирование с помощью _simulate

```
POST _ingest/pipeline/_simulate
 {
   "pipeline": {
     "description": "Item View Pipeline",
     "processors": [
       {
         "grok": {
           "field": "log",
           "patterns": ["%{IPORHOST:remote_ip} %{GREEDYDATA:dash} %{DATA:user_name} %
{TIMESTAMP_ISO8601:timestamp} %{DATA:referrer} %{WORD:http_method} %{DATA:url} HTTP/%
{NUMBER:http_version} %{NUMBER:response_code} %{NUMBER:body_sent_bytes} %{DATA:agent} -
%{BASE10NUM:request_time} %{BASE10NUM:upstream_response_time} %{GREEDYDATA:pipe} %
{NUMBER:connection} %{NUMBER:connection_requests}"]
         }
       }
     ]
   },
   "docs": [
     {
       "_source": {
         "log": "89.169.145.79 - - 2019-09-11T09:02:34.234+03:00 service.hapicloud.io
GET /health HTTP/1.1 200 25 - Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML,
like Gecko) Chrome/76.0.3809.100 Safari/537.36 - 0.017 0.016 . 7353327 1"
       }
     }
   ]
 }
```

Удаляем запись с полем 'log'

```
POST _ingest/pipeline/_simulate
 {
   "pipeline": {
     "description": "Item View Pipeline",
     "processors": [
       {
         "grok": {
           "field": "log",
           "patterns": ["%{IPORHOST:remote_ip} %{GREEDYDATA:dash} %{DATA:user_name} %
{TIMESTAMP_ISO8601:timestamp} %{DATA:referrer} %{WORD:http_method} %{DATA:url} HTTP/%
{NUMBER:http_version} %{NUMBER:response_code} %{NUMBER:body_sent_bytes} %{DATA:agent} -
%{BASE10NUM:request_time} %{BASE10NUM:upstream_response_time} %{GREEDYDATA:pipe} %
{NUMBER:connection} %{NUMBER:connection_requests}"]
         }
       },
{
         "remove": {
           "field": "log"
         }
       }
     ]
   },
   "docs": [
     {
       "_source": {
         "log": "89.169.145.79 - - 2019-09-11T09:02:34.234+03:00 service.hapicloud.io
GET /health HTTP/1.1 200 25 - Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML,
```

```
like Gecko) Chrome/76.0.3809.100 Safari/537.36 - 0.017 0.016 . 7353327 1"
        }
      }
    ]
  }
```

API симуляции контейнера

```
POST _ingest/pipeline/_simulate
 {
   "pipeline": {
     "description": "Item View Pipeline",
     "processors": [
       { "grok": {
           "field": "log",
           "patterns": ["%{IPORHOST:remote_ip} %{GREEDYDATA:dash} %{DATA:user_name} %
{TIMESTAMP_ISO8601:timestamp} %{DATA:referrer} %{WORD:http_method} %{DATA:url} HTTP/%
{NUMBER:http_version} %{NUMBER:response_code} %{NUMBER:body_sent_bytes} %{DATA:agent} -
%{BASE10NUM:request_time} %{BASE10NUM:upstream_response_time} %{GREEDYDATA:pipe} %
{NUMBER:connection} %{NUMBER:connection_requests}"] }
       },
       { "remove": { "field": "log"} },
       {   "date_index_name": {
           "field": "timestamp",
           "index_name_prefix": "viewitem-",
           "date_rounding": "M",
           "index_name_format": "yyyy-MM" } } ]
   },
   "docs": [ { "_source": {"log": "89.169.145.79 - - 2019-09-11T09:02:34.234+03:00
service.hapicloud.io GET /health HTTP/1.1 200 25 - Mozilla/5.0 (X11; Linux x86_64)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/76.0.3809.100 Safari/537.36 - 0.017 0.016
. 7353327 1"} } ]
 }
```

Использование конвейера

```
PUT _ingest/pipeline/view_item_pipeline
 {
   "description": "Item View Pipeline",
   "processors": [
     {
       "grok": {
         "field": "log",
         "patterns": [ "%{IPORHOST:remote_ip} %{GREEDYDATA:dash} %{DATA:user_name} %
{TIMESTAMP_ISO8601:timestamp} %{DATA:referrer} %{WORD:http_method} %{DATA:url} HTTP/%
{NUMBER:http_version} %{NUMBER:response_code} %{NUMBER:body_sent_bytes} %{DATA:agent} -
%{BASE10NUM:request_time} %{BASE10NUM:upstream_response_time} %{GREEDYDATA:pipe} %
{NUMBER:connection} %{NUMBER:connection_requests}"]
       }
     },
     { "remove": { "field": "log" } },
     { "date_index_name": {
         "field": "timestamp",
         "index_name_prefix": "viewitem_",
         "date_rounding": "M",
         "index_name_format": "yyyy_MM" }
     }
   ]
 }
```

Создадим несколько записей в индексе chapter5

```
 POST chapter5/log/?pipeline=view_item_pipeline
 {   "log":"89.169.145.79 - - 2019-09-11T09:02:34.234+03:00 service1.hapicloud.io GET
/health HTTP/1.1 200 25 - Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML,
like Gecko) Chrome/76.0.3809.100 Safari/537.36 - 0.017 0.016 . 7353327 1" }

 POST chapter5/log/?pipeline=view_item_pipeline
 {   "log":"89.169.145.79 - - 2019-09-11T09:02:35.234+03:00 service2.hapicloud.io GET
/health HTTP/1.1 200 25 - Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML,
like Gecko) Chrome/76.0.3809.100 Safari/537.36 - 0.017 0.016 . 7353327 1" }


 POST chapter5/log/?pipeline=view_item_pipeline
 {   "log":"89.169.145.79 - - 2019-09-11T09:02:36.234+03:00 service3.hapicloud.io GET
/health HTTP/1.1 200 25 - Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML,
like Gecko) Chrome/76.0.3809.100 Safari/537.36 - 0.017 0.016 . 7353327 1" }


 POST chapter5/log/?pipeline=view_item_pipeline
 {   "log":"89.169.145.79 - - 2019-09-11T09:02:37.234+03:00 service4.hapicloud.io GET
/health HTTP/1.1 200 25 - Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML,
like Gecko) Chrome/76.0.3809.100 Safari/537.36 - 0.017 0.016 . 7353327 1 " }
```

API получения контейнера

```
PUT /_ingest/pipeline/firstpipeline
GET /_ingest/pipeline
```

API удаления контейнера

```
DELETE /_ingest/pipeline/firstpipeline
```

Развёртывание kubernetes в AWS с помощью kops

Setup IAM user

Необходимые права для создания кластера

- AmazonEC2FullAccess
- AmazonRoute53FullAccess
- AmazonS3FullAccess
- IAMFullAccess
- AmazonVPCFullAccess

Setup IAM user

```
aws iam create-group --group-name kops
aws iam attach-group-policy --policy-arn arn:aws:iam::aws:policy/AmazonEC2FullAccess --
group-name kops
aws iam attach-group-policy --policy-arn arn:aws:iam::aws:policy/AmazonRoute53FullAccess
--group-name kops
aws iam attach-group-policy --policy-arn arn:aws:iam::aws:policy/AmazonS3FullAccess --
group-name kops
aws iam attach-group-policy --policy-arn arn:aws:iam::aws:policy/IAMFullAccess --group-
name kops
aws iam attach-group-policy --policy-arn arn:aws:iam::aws:policy/AmazonVPCFullAccess --
group-name kops
aws iam create-user --user-name kops
aws iam add-user-to-group --user-name kops --group-name kops
aws iam create-access-key --user-name kops
```

AWS Credentials

```
# configure the aws client to use your new IAM user
aws configure           # Use your new access and secret key here
aws iam list-users      # you should see a list of all your IAM users here

# Because "aws configure" doesn't export these vars for kops to use, we export them now
export AWS_ACCESS_KEY_ID=$(aws configure get aws_access_key_id)
export AWS_SECRET_ACCESS_KEY=$(aws configure get aws_secret_access_key)
aws iam list-access-keys --user-name kops

# We can just create access keys with CLI
# You cannot list the secret access keys for IAM users. If the secret access keys are
lost, you must create new access keys using the create-access-keys command.

aws iam update-access-key
```

Для работы с AWS используем AWS CLI

```
aws configure
export AWS_ACCESS_KEY_ID=$(aws configure get aws_access_key_id)
export AWS_SECRET_ACCESS_KEY=$(aws configure get aws_secret_access_key)
```

Хранение стейта

```
aws s3api create-bucket \
          --bucket aakilin-state-store \
          --region eu-west-2 \
          --create-bucket-configuration LocationConstraint=eu-west-2
aws s3api put-bucket-versioning \
          --bucket aakilin-state-store \
          --versioning-configuration Status=Enabled
```

Создание кластера

```
export NAME=cluster2.k8s.local
export KOPS_STATE_STORE=s3://aakilin-state-store
kops create cluster --name cluster2.k8s.local --zones eu-west-2a --node-count 2 --state
s3://aakilin-state-store
kops get --name cluster2.k8s.local -o yaml
kops update cluster --name cluster2.k8s.local --yes
kops validate cluster -v10 --logtostderr
kubectl cluster-info dump
```

Редактирование кластера

```
kops edit ig --name=${NAME} nodes
kops edit ig --name=${NAME} master-eu-west-2a
```

Запуск кластера:

```
kops update cluster --name cluster2.k8s.local --yes
```

Просмотр кластера

```
kops validate cluster -v10 --logtostderr
kops get --name cluster2.k8s.local -o yaml
kubectl cluster-info dump
```

Завершение установки кластера - необходимые сервисы и RBAC

```
kubectl get nodes
```

Helm

tiller - это служба Helm, которая устанавливает для вас другие сервисы

```
# Создаём служебную учетную запись с именем «tiller», которой предоставляются привилегии администратора кластера (как требуется для RBAC)
kubectl create serviceaccount --namespace kube-system tiller
kubectl create clusterrolebinding tiller-cluster-rule --clusterrole=cluster-admin --serviceaccount=kube-system:tiller
# Затем мы подключаем Helm к кластеру и устанавливаем tiller, используя эту учетную запись службы
helm init --service-account tiller
```

Traefik

Traefik - это обратный прокси-сервер с панелью управления и автоматическим бесплатным SSL с использованием LetsEncrypt и балансировщик нагрузки, который автоматизирует многие настройки веб-сервера

```
dashboard:
  enabled: true
  domain: traefik.cluster2.k8s.local
  auth:
    basic:
      aakilin: Ku8erN3t3$
rbac:
  enabled: true
ssl:
  enabled: true
  enforced: true
```

```
helm install --name traefik \
     --values traefik.yaml stable/traefik
```

Проверяем сервисы

```
kubectl --namespace kube-system get svc
```

```
kubectl describe svc traefik --namespace default | grep Ingress | awk '{print $3}'
```

Развертывание проекта Django в Кубернетесе

Развертывание проекта Django в Кубернетесе (с celery worker)

```
git clone https://gitlab.com/ndevox/django-k8s-starter
```

Пререквезиты к базовому сервису Kubernetes Django

- deployment.
- service.
- ingress.
- configmap.
- job.

Настройка зависимостей

```
helm install --name rabbit --values rabbit-values.yaml stable/rabbitmq
helm install --name postgres --values postgres-values.yaml stable/postgresql

helm install --name postgres \
  --set postgresqlPassword=secretpassword,postgresqlDatabase=my-database \
    stable/postgresql

kubectl port-forward svc/django-k8s-starter-svc 8000:8000
```

Создание ConfigMap

```yaml
apiVersion: v1
kind: ConfigMap
metadata:
  name: django-k8s-starter-config
data:
  SECRET_KEY: 'REPLACE WITH YOUR SECRET KEY'
  BROKER_USER: demo  # Make sure this matches the one in rabbit-values.yaml
  BROKER_PASS: demo  # Make sure this matches the one in rabbit-values.yaml
  BROKER_HOST: rabbit-rabbitmq # This matches the name we gave the service with helm.
  DB_HOST: postgres-postgresql # This matches the name we gave the service with helm
  DB_NAME: demo  # Make sure this matches the one in postgres-values.yaml
  DB_USER: demo  # Make sure this matches the one in postgres-values.yaml
  DB_PASS: demo  # Make sure this matches the one in postgres-values.yaml
```

```
kubectl apply -f configmap.yaml
```

Создание deployment

```
    spec:
      containers:
      - name: django-k8s-starter-web
        image: ndevox/django-kubernetes-starter:latest
        imagePullPolicy: Always
        command: ["gunicorn", "--workers", "1", "--bind", ":8000",
                  "--log-level", "INFO", "django_k8s_starter.wsgi:application"]
        envFrom:
        - configMapRef: # configMapRef.name: имя ConfigMap для использования в среде.
            name: django-k8s-starter-config
        ports:
        - containerPort: 8000
```

```
kubectl apply -f deployment.yaml
```

Создание service

```
kubectl apply -f service.yaml
```

Создание ingress

```
kubectl apply -f ingress.yaml
```

EKSCTL

Запуск кластера с помощью EKSCTL

```
eksctl create cluster --name=eksworkshop-eksctl --nodes=3 --alb-ingress-access --
region=eu-west-2
# Запускаем дашборд
kubectl apply -f
https://raw.githubusercontent.com/kubernetes/dashboard/v1.10.1/src/deploy/recommended/ku
bernetes-dashboard.yaml
kubectl proxy --port=8080 --address='0.0.0.0' --disable-filter=true &
```

Устанавливаем Dashboard

```
kubectl apply -f
https://raw.githubusercontent.com/kubernetes/dashboard/v1.10.1/src/deploy/recommended/ku
bernetes-dashboard.yaml
kubectl apply -f
https://raw.githubusercontent.com/kubernetes/heapster/master/deploy/kube-
config/influxdb/heapster.yaml
kubectl apply -f
https://raw.githubusercontent.com/kubernetes/heapster/master/deploy/kube-
config/influxdb/influxdb.yaml
kubectl apply -f
https://raw.githubusercontent.com/kubernetes/heapster/master/deploy/kube-
config/rbac/heapster-rbac.yaml
kubectl apply -f /home/akilin/Paragon/kubernetes/ELK/EKS/eks-admin-service-account.yaml
kubectl -n kube-system describe secret $(kubectl -n kube-system get secret | grep eks-
admin | awk '{print $1}')
kubectl proxy
```

Запускаем метрики

```
DOWNLOAD_URL=$(curl --silent "https://api.github.com/repos/kubernetes-incubator/metrics-server/releases/latest" | jq -r .tarball_url)
DOWNLOAD_VERSION=$(grep -o '[^/v]*$' <<< $DOWNLOAD_URL)
curl -Ls $DOWNLOAD_URL -o metrics-server-$DOWNLOAD_VERSION.tar.gz
mkdir metrics-server-$DOWNLOAD_VERSION
tar -xzf metrics-server-$DOWNLOAD_VERSION.tar.gz --directory metrics-server-$DOWNLOAD_VERSION --strip-components 1
kubectl apply -f metrics-server-$DOWNLOAD_VERSION/deploy/1.8+/
kubectl get deployment metrics-server -n kube-system
# Посмотрим на метрики
kubectl get --raw /metrics
```

Deploy Prometheus

```
kubectl create namespace monitoring
helm install stable/prometheus \
--name prometheus \
--namespace monitoring \
--set
alertmanager.persistentVolume.storageClass="gp2",server.persistentVolume.storageClass="g
p2"
kubectl get pods -n monitoring
kubectl --namespace=monitoring port-forward deploy/prometheus-server 9090
```

Install Grafana

```
helm install --name grafana stable/grafana
kubectl apply -f monitoring/grafana/config.yml
kubectl get configmaps -n monitoring
helm install stable/grafana -f grafana/values.yml  --namespace monitoring  --name
grafana
helm upgrade --install grafana stable/grafana -f grafana/values.yml --namespace
monitoring

# Пойдём в графану через UI
kubectl get secret --namespace default grafana -o jsonpath="{.data.admin-password}" |
base64 --decode ; echo
export POD_NAME=$(kubectl get pods --namespace default -l "app=grafana,release=grafana"
-o jsonpath="{.items[0].metadata.name}")
kubectl --namespace default port-forward $POD_NAME 3000

kubectl port-forward -n monitoring svc/grafana 3000:80
```

Deploy Loki

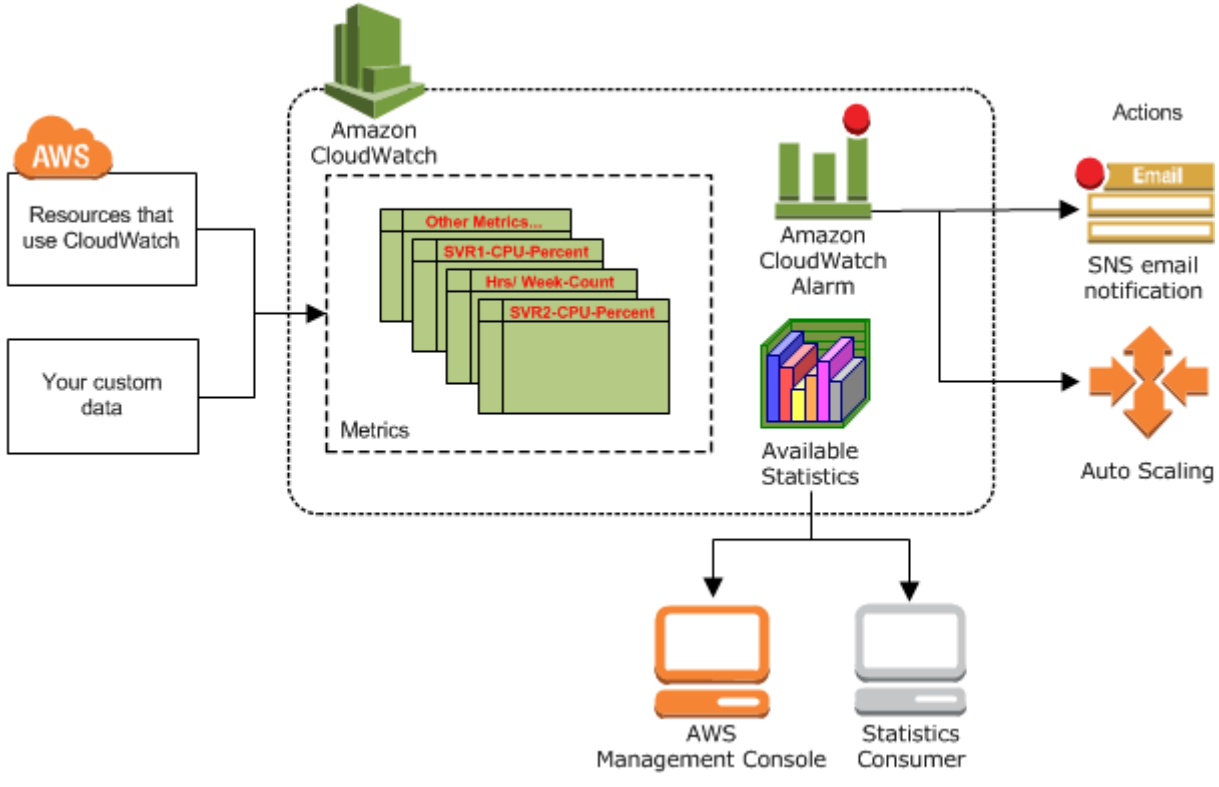
- Loki очень простой и малокушающий ресурсы сервис централизованного мониторинга, который легко интегрируется в графану. Тут нет парсинга логов и возможности их изменять на лету, тем не менее, как именно хранилка для логов очень подойдёт.

```
helm repo add loki https://grafana.github.io/loki/charts
helm repo update
helm upgrade --install loki --namespace=monitoring loki/loki-stack
```

https://grafana.github.io/loki/charts/ http://loki:3100/

AWS CloudWatch

Логирование с помощью CloudWatch



Amazon CloudWatch Concepts

- Namespaces
- Metrics
- Dimensions
- Statistics
- Percentiles
- Alarms