

# Система сбора логов ELK



Сергей  
Андрюнин



**Сергей Андрюнин**

DevOps инженер  
RTLabs



---

# План занятия

1. [Введение](#)
2. [Logstash](#)
3. [Kibana](#)
4. [Filebeat](#)
5. [Index Lifecycle Management \(ILM\)](#)
6. [Пример политики hot-warm-cold](#)
7. [Настройка Elasticsearch](#)
8. [Итоги](#)
9. [Домашнее задание](#)



# Введение

# Введение



**Централизованный сбор логов** - один из важнейших разделов мониторинга в современном мире.

Существует множество решений, но основным на текущий момент является система сбора логов, построенная на основе стэка технологий **Elasticsearch-Logstash-Kibana (ELK)**.

Данный стэк позволяет “из коробки” получить средство агрегации и трансформации, хранения и визуализации данных логирования.

Наиболее распространенное решения для построения системы сбора логов на основе ELK является **архитектура “Hot-Warm”**.

“Hot-Warm” архитектура представляет из себя кластер Elasticsearch узлов, каждый из которых имеет свою метку (hot/warm).

---

# Введение



Кластерные узлы, помеченные, как **“Hot”** осуществляют **хранение и сбор “быстрых” данных логирования**, например за последние сутки.

Такие узлы требовательны к скорости операций чтения/записи диска и, соответственно, хранят данные на **SSD-носителях**.

В то же время, обычно им не требуется большой размер носителя, так как время хранения данных на таких узлах небольшое.

# Введение



Кластерные узлы, помеченные, как “**Warm**” осуществляют **хранение и сбор “медленных” данных логирования**, например всё что следует за текущими сутками.

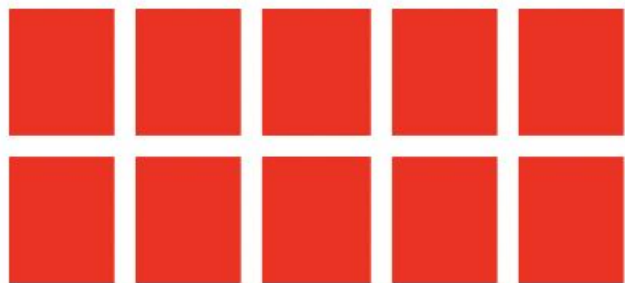
Такие узлы не требовательны к операциям чтения/записи диска, но требовательны к объёму. Соответственно, хранение данных можно осуществлять на **HDD-носителях**.

Также возможно использовать “**Cold**” узлы, которые, например, **хранят данные старше 30 дней**.

# Введение



"data": "hot"



"data": "warm"



"data": "cold"



Взято с сайта: [elastic.co](https://elastic.co)



# Введение



**Средством агрегации логов, единой точкой входа является Logstash.** Данное средство позволяет *парсить и преобразовывать логи и складывать в `elasticsearch` в удобном виде.*

**Средством визуализации логов является Kibana.** Данное средство представляет из себя веб-интерфейс и *позволяет просматривать логи, строить графики на основе данных из логов и многое другое.*

**Сбор логов осуществляется двумя путями:**

- Приложение пишет логи напрямую в logstash (например, посредством tcp)
- Сбор производит Filebeat, который считывает данные логов из файлов.



# Logstash

---

# Logstash

**Logstash** в ELK-стэке сбора логов является инструментом для приёма данных, их преобразования и отправки в кластер БД elasticsearch.

**Конфигурирование Logstash** осуществляется через специальный конфигурационный файл.

Есть **универсальный обработчик входных данных *grok***. Данный обработчик позволяет парсить входные данные и раскладывать приведённые данные в виде ключ-значение для хранения в elasticsearch.



# Logstash

**Графа input** конфигурационного файла *задаёт входные точки logstash.*

Это может быть интерактивный обмен, сетевое соединение, файл и т.д.

Также указывается тип входных событий для их оптимальной обработки, например syslog.

```
input {  
  tcp {  
    port => 5000  
    type => syslog  
  }  
  udp {  
    port => 5000  
    type => syslog  
  }  
}
```

Взято с сайта: [elastic.co](https://www.elastic.co)

# Logstash

**Графа filter** отвечает за *парсинг входных данных и преобразованию их в необходимую структуру*.

Logstash имеет множество стандартных парсеров входных данных, но самым универсальным является grok (но наименее производительным).

В данной графе можно указать в какие ключи складывать данные, вставлять унарные операторы в случае особых входных событий и т.д.

```
filter {  
  if [path] =~ "access" {  
    mutate { replace => { type => "apache_access" } }  
    grok {  
      match => { "message" => "%{COMBINEDAPACHELOG}" }  
    }  
    date {  
      match => [ "timestamp" , "dd/MMM/yyyy:HH:mm:ss Z" ]  
    }  
  } else if [path] =~ "error" {  
    mutate { replace => { type => "apache_error" } }  
  } else {  
    mutate { replace => { type => "random_logs" } }  
  }  
}
```

Взято с сайта: [elastic.co](https://www.elastic.co)

# Logstash

**Графа output** отвечает за *конечную точку для отправки данных из Logstash*.

Можно указать несколько конечных точек для параллельной отправки данных. Например, файлы и Elasticsearch.

Также для каждой конечной точки существуют дополнительные гибкие настройки, например добавление даты в название файла или применение шаблона индексов в Elasticsearch к данным.

```
output {  
  elasticsearch { hosts => ["localhost:9200"] }  
  stdout { codec => rubydebug }  
}
```

Взято с сайта: [elastic.co](https://www.elastic.co)



# Kibana

---

# Kibana

**Kibana** является эффективным *средством визуализации данных логов*.

В Kibana можно производить вывод как текстовых данных, так и производить сортировку логов по определенным значениям (например, уровням логирования).

Возможно проведение агрегации данных и их визуализация в виде диаграм, графиков или интерактивных карт.





# Kibana

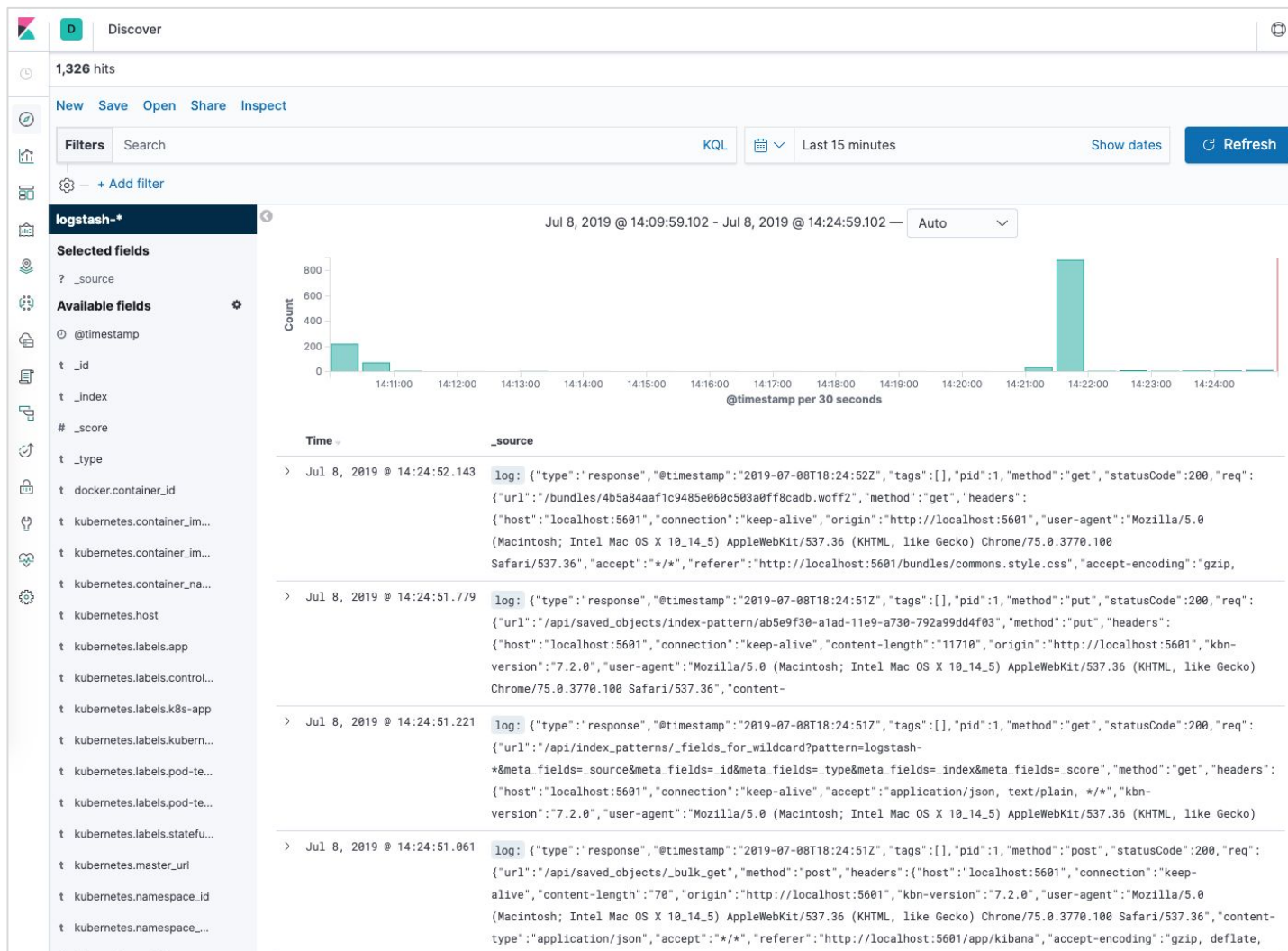
**Базовая настройка Kibana** довольно проста и заключается лишь в указании интерфейса, на котором она доступна и конечных точек источников данных в виде массива сетевых параметров узлов elasticsearch.

## Пример настройки Kibana:

```
server.host: "123.456.789"  
  
elasticsearch.url: "[https://123.456.789:9200](http://123.456.789:9200/)"
```

Взято с сайта: [kb.objectrocket.com](http://kb.objectrocket.com)

# Kibana

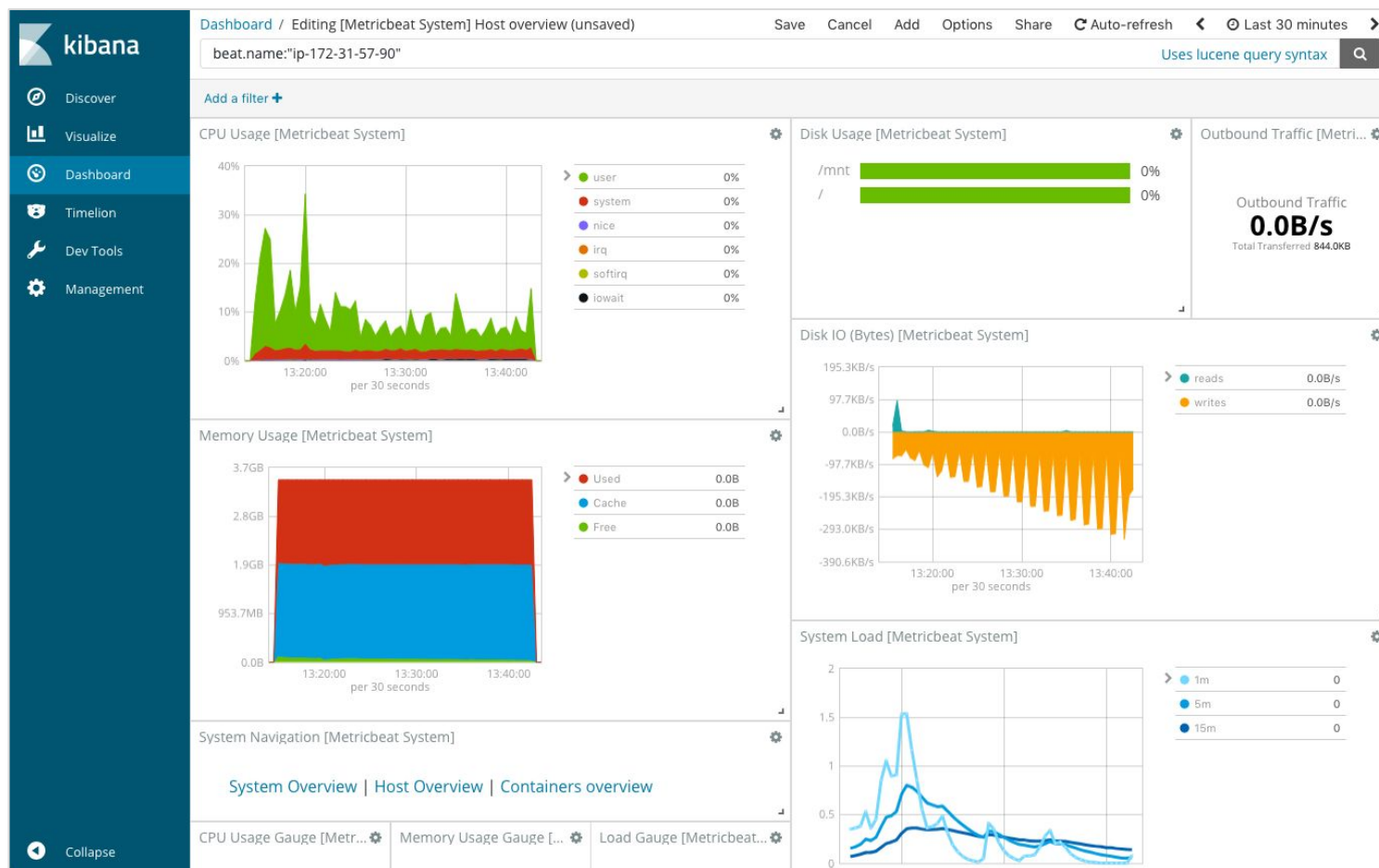


# Kibana



Взято с сайта: balagetechn.com

# Kibana



Взято с сайта: logz.io



# **Filebeat**

---

# Filebeat

**Filebeat** является эффективным средством сбора данных логирования из файлов и отправки в *logstash/elasticsearch*.

В процессе сбора Filebeat может “схлопывать” данные логов при их последовательном повторе, хранить данные в буфере до восстановления конечной точки отправки, игнорировать определенные события логирования.



## Filebeat

# Filebeat

**Базовая конфигурация Filebeat** довольно проста и заключается в настройке источника данных и конечной точки в конфигурационном файле.

**Пример настройки чтения логов из /var/logs и отправки их в logstash:**

```
filebeat.inputs:  
- type: log  
  enabled: true  
  paths:  
    - /var/log/*.log
```

```
output.logstash:  
  hosts: ["127.0.0.1:5044"]
```

Взято с сайта: [elastic.co](https://www.elastic.co)



# Index Lifecycle Management (ILM)



# Index Lifecycle Management

**Index Lifecycle Management (ILM)** - процесс перемещения индексов (данных) между нодами в Hot-Warm архитектуре.

Данный процесс настраивается с помощью *API Elasticsearch*.

**Настройка процесса** перемещения индекса между узлами - называется *политикой*.

**В политике** можно указать триггер перемещения индекса по времени или по размеру primary shard индекса на ноде определённого типа.

```
PUT /_ilm/policy/my_policy
{
  "policy":{
    "phases":{
      "hot":{
        "actions":{
          "rollover":{
            "max_size":"50gb",
            "max_age":"30d"
          }
        }
      }
    }
  }
}
```

Взято с сайта: [elastic.co](https://www.elastic.co)

# Index Lifecycle Management

Чтобы **привязать индекс к политике**, необходимо *указать шаблон, в котором указано, к каким индексам применяется данная политика.*

В шаблоне индексов вы можете использовать **wildcard** и распространить данный шаблон на несколько или на все индексы.

Также в шаблоне необходимо указать псевдоним (**alias**) для индекса.

```
PUT _template/my_template
{
  "index_patterns": ["test-*"],
  "settings": {
    "index.lifecycle.name": "my_policy",
    "index.lifecycle.rollover_alias": "test-alias"
  }
}
```

Взято с сайта: [elastic.co](https://elastic.co)

# Index Lifecycle Management

**Привязка псевдонимов** также осуществляется посредством *API Elasticsearch*.

Псевдонимы упрощают поиск индексов между нодами кластера и ограничивает операции записи только на индексы *hot* узлов.

```
PUT test-000001
{
  "aliases": {
    "test-alias": {
      "is_write_index": true
    }
  }
}
```

Взято с сайта: [elastic.co](https://www.elastic.co)

---

# Пример политики hot-warm-cold

# Пример политики hot-warm-cold

На данном этапе устанавливаем **высокий приоритет индекса**, чтобы горячие индексы восстанавливались раньше других индексов (при перезагрузке узла).

Через 30 дней или 50 ГБ (в зависимости от того, что наступит раньше) индекс будет обновлен (перенесён), и будет создан новый индекс.

Новый индекс запустит политику заново, а текущий индекс будет перенесён в warm фазу.

```
"hot": {  
  "actions": {  
    "rollover": {  
      "max_size": "50gb",  
      "max_age": "30d"  
    },  
    "set_priority": {  
      "priority": 50  
    }  
  }  
},
```

Взято с сайта: [elastic.co](https://www.elastic.co)

# Пример политики hot-warm-cold

Когда индекс перейдёт в теплую фазу:

- сократится количество его шардов до 1
- установится приоритет индекса на значение ниже hot (но больше, чем cold)
- произойдет миграция индекса на warm узлы
- срок жизни в warm фазе составляет 23 дня

**P.S.:** *min\_age* указывает сколько должно минимально пройти времени с создания индекса до перехода в какую-либо фазу

```
"warm": {
  "min_age": "7d",
  "actions": {
    "forcemerge": {
      "max_num_segments": 1
    },
    "shrink": {
      "number_of_shards": 1
    },
    "allocate": {
      "require": {
        "data": "warm"
      }
    },
    "set_priority": {
      "priority": 25
    }
  }
},
```

Взято с сайта: [elastic.co](https://www.elastic.co)

# Пример политики hot-warm-cold

## При переходе в холодную фазу:

- индекс замораживается, то есть становится недоступным, но хранится для его возможного будущего использования
- установится минимальный приоритет восстановления индекса
- произойдет миграция индекса на cold узлы
- срок жизни индекса в cold фазе составляет 30 дней

```
"cold": {  
  "min_age": "30d",  
  "actions": {  
    "set_priority": {  
      "priority": 0  
    },  
    "freeze": {},  
    "allocate": {  
      "require": {  
        "data": "cold"  
      }  
    }  
  }  
},
```

Взято с сайта: [elastic.co](https://www.elastic.co)

# Пример политики hot-warm-cold

Данная фаза управляет **удалением индексов**.

В текущей политике указано, что индекс должен быть удален по истечении срока его жизни 60 дней, в независимости от его фазы.

```
"delete": {  
  "min_age": "60d",  
  "actions": {  
    "delete": {}  
  }  
}
```

Взято с сайта: [elastic.co](https://www.elastic.co)





# Настройка Elasticsearch

# Настройка elasticsearch

Указание типа узла происходит путём добавление конфигурации в elasticsearch.yml (прочие настройки стандартны для кластера elasticsearch):

- *Hot-ноды:*

```
~]# cat /etc/elasticsearch/elasticsearch.yml | grep attr
# Add custom attributes to the node:
node.attr.box_type: hot
```

- *Warm-ноды:*

```
~]# cat /etc/elasticsearch/elasticsearch.yml | grep attr
# Add custom attributes to the node:
node.attr.box_type: warm
```

- *Cold-ноды:*

```
~]# cat /etc/elasticsearch/elasticsearch.yml | grep attr
# Add custom attributes to the node:
node.attr.box_type: cold
```

# Настройка elasticsearch

Рекомендуемые конфигурации для построения hot-warm кластера:

- **Hot узлы**
  - 16 CPU
  - 32 GB RAM
  - 256 SSD
- **Warm узлы**
  - 8 CPU
  - 16 GB RAM
  - 1 Tb Hdd

Данная конфигурация позволяет обслуживать обработку следующих потоков данных логирования:

- размер индексов: до 30 Gb
- количество индексов: до 35



# Итоги

---

# Итоги

В данной лекции мы узнали:

- Что такое hot-warm архитектура
- Домены ответственности инструментов Logstash/Kibana/Filebeat и их базовую настройку
- Как реализовать hot-warm архитектуру на основе ELK-стэка
- Рассмотрели пример политики реализации hot-warm архитектуры
- Рассмотрели рекомендуемые требования hot-warm elasticsearch кластера в составе ELK

---

# Домашнее задание

Давайте посмотрим ваше [домашнее задание](#).

- Вопросы по домашней работе задавайте **в чате** мессенджера Slack.
- Задачи можно сдавать **по частям**.
- Зачёт по домашней работе проставляется после того, как **приняты все задачи**.

**Задавайте вопросы и  
пишите отзыв о лекции!**

**Сергей Андрюнин**