

Оглавление

Об авторах	13
О научном редакторе	15
Предисловие	16
Структура книги.....	16
Что вам понадобится для работы	18
Для кого предназначена книга	18
Условные обозначения.....	18
Загрузка примеров кода.....	19
Скачивание цветных изображений, использованных в книге	19
Глава 1. Введение в Elastic Stack.....	20
Что такое система Elasticsearch и чем она хороша	21
Неструктурированность и документоориентированность.....	21
Поиск.....	22
Анализ данных.....	23
Поддержка пользовательских библиотек и REST API	23
Легкое управление и масштабирование.....	23
Работа в псевдореальном времени.....	24
Высокая скорость работы.....	24
Устойчивость к ошибкам и сбоям.....	24
Обзор компонентов Elastic Stack.....	25
Elasticsearch.....	25
Logstash.....	25
Beats.....	26

Kibana.....	26
X-Pack	27
Elastic Cloud.....	28
Способы применения Elastic Stack.....	28
Анализ и безопасность логов	29
Поиск по продуктам.....	29
Анализ показателей.....	30
Веб-поиск и поиск по сайту.....	30
Скачивание и установка	31
Установка Elasticsearch.....	31
Установка Kibana.....	32
Резюме.....	32
Глава 2. Начало работы с Elasticsearch	34
Пользовательский интерфейс Kibana Console	34
Основные понятия Elasticsearch.....	36
Индекс.....	37
Тип	38
Документ.....	38
Узел.....	39
Кластер	39
Шарды и копии.....	40
Разметка и типы данных.....	42
Обратный индекс	47
Операции CRUD.....	48
Index API.....	48
Get API	50
Update API	50
Delete API.....	52
Создание индексов и контролирование разметки.....	52
Создание индекса.....	53
Создание разметки типов в существующем индексе	54
Обновление разметки.....	55
Обзор REST API.....	56
Общие правила API	57
Управление несколькими индексами	58
Резюме.....	60

8 Оглавление

Глава 3. Поиск — вот что важно	61
Основы анализа текста.....	61
Анализаторы Elasticsearch	62
Использование встроенных анализаторов.....	66
Добавление автозавершения в пользовательском анализаторе	70
Поиск по структурированным данным	74
Запрос диапазона	75
Запрос существования	78
Терм-запрос.....	79
Полнотекстовый поиск	80
Запрос соответствия	82
Запрос соответствия фразы.....	85
Запрос нескольких соответствий	87
Написание составных запросов	88
Запрос постоянной оценки.....	88
Булев запрос	91
Резюме.....	94
Глава 4. Анализ данных с помощью Elasticsearch	96
Основы агрегации	96
Сегментарные агрегации	97
Метрические агрегации	98
Матричные агрегации	98
Агрегации контейнеров	98
Подготовка данных к анализу	99
Загрузка данных с помощью Logstash	102
Метрические агрегации	103
Агрегации суммы, среднего, максимального и минимального значений.....	103
Агрегации статистики и расширенной статистики.....	106
Агрегация мощности.....	108
Сегментарные агрегации	109
Сегментирование строковых данных.....	109
Сегментирование числовых данных	114
Агрегирование отфильтрованных данных	117
Вложенные агрегации	118
Сегментирование с нестандартными условиями.....	121
Сегментирование данных даты/времени	123
Сегментирование геопространственных данных.....	129
Агрегации контейнеров.....	132
Резюме.....	134

Глава 5. Анализ журнальных данных	136
Вызовы при анализе логов	136
Logstash.....	138
Архитектура Logstash	142
Обзор плагинов Logstash.....	144
Установка или обновление плагинов	145
Обзор плагинов.....	148
Узел поглощения данных.....	158
Определение контейнера	158
Ingest API.....	159
Резюме.....	162
Глава 6. Разработка контейнеров данных с помощью Logstash	163
Обработка и дополнение логов с помощью Logstash	163
Плагины фильтрации	164
Введение в Beats	171
Beats от Elastic.co	173
Компоненты Beats от сообщества	174
Logstash против Beats	175
Filebeat	176
Скачивание и установка Filebeat.....	176
Архитектура.....	178
Настройка Filebeat.....	179
Резюме.....	190
Глава 7. Визуализация данных в Kibana	191
Скачивание и установка Kibana.....	191
Установка в Windows	192
Установка в Linux	192
Настройка Kibana	194
Подготовка данных.....	195
Пользовательский интерфейс Kibana.....	196
Взаимодействие с пользователем.....	197
Настройка шаблона индекса	198
Исследование	200
Визуализация	213
Создание визуализации	216
Типы визуализаций	218
Визуализации в деле	219

Панели управления	226
Timelion	230
Использование плагинов.....	234
Установка плагинов	234
Удаление плагинов.....	235
Резюме.....	235
Глава 8. Elastic X-Pack	236
Установка X-Pack	236
Установка X-Pack в Elasticsearch	236
Установка X-Pack в Kibana	239
Удаление X-Pack.....	240
Настройка X-Pack	240
Безопасность.....	241
Пользовательская аутентификация	241
Авторизация пользователя.....	243
Безопасность на деле	245
Мониторинг Elasticsearch	260
Раздел Monitoring (Мониторинг) пользовательского интерфейса	261
Метрики Elasticsearch	262
Уведомления.....	267
Структура наблюдателя.....	268
Уведомления в деле	272
Резюме.....	279
Глава 9. Запуск Elastic Stack в работу	280
Размещение Elastic Stack в управляемом облаке	280
Настройка и запуск Elastic Cloud.....	281
Использование Kibana	283
Изменение настроек.....	283
Восстановление	284
Размещение Elastic Stack на своем хостинге.....	285
Выбор аппаратной части	285
Выбор операционной системы	286
Настройка узлов Elasticsearch.....	286
Управление Elasticsearch и ее мониторинг	287

Запуск в контейнерах Docker	287
Особенности запуска в облаке	288
Резервное копирование и восстановление.....	291
Настройка репозитория для снепшотов	291
Облако распределенных файловых систем	292
Создание снепшотов	293
Восстановление из определенного снепшота.....	293
Присвоение псевдонимов индексам	294
Что такое псевдонимы индексов	294
Чем могут помочь псевдонимы индексов.....	295
Настройка шаблонов индексов	295
Определение шаблона индекса	296
Создание индекса на ходу	296
Моделирование данных временного ряда.....	297
Масштабирование индекса с непредсказуемым объемом через некоторое время	297
Изменение разметки со временем.....	299
Автоматическое удаление более старых документов.....	299
Как временный индекс решает эти проблемы	300
Резюме.....	301
Глава 10. Создание приложения для анализа данных с датчиков.....	302
Введение в приложение	302
Создаваемые датчиками данные.....	304
Метаданные с датчиков	304
Окончательно сохраняемые данные.....	306
Моделирование данных в Elasticsearch.....	307
Настройка базы метаданных	309
Создание контейнера данных Logstash	310
Прием запросов JSON по сети.....	311
Дополнение JSON метаданными из нашей базы данных MySQL.....	312
Сохранение в Elasticsearch полученных в результате документов	315
Отправка данных в Logstash через HTTP	316
Визуализация данных в Kibana.....	317
Настройка шаблона индекса в Kibana	317
Создание визуализаций.....	318
Создание панели управления	324
Резюме.....	326

Глава 11. Мониторинг серверной инфраструктуры.....	328
Metricbeat.....	328
Скачивание и установка Metricbeat	328
Архитектура.....	330
Настройка Metricbeat	334
Конфигурация модуля	334
Общие настройки.....	336
Конфигурация вывода	337
Логирование	339
Сбор системных метрик.....	339
Запуск Metricbeat с модулем system	340
Указание псевдонимов.....	343
Визуализация системных метрик в Kibana	344
Архитектура внедрения	347
Резюме.....	349

Плагины фильтрации

Поскольку в следующей главе мы будем детально рассматривать различные способы изменения и дополнения данных логов с помощью разных плагинов фильтрации, сейчас мы пропустим связанную с этим информацию.

Узел поглощения данных

До версии Elasticsearch 5.0, если нам требовалась предварительная обработка документов до индексирования, единственным возможным способом сделать это было использовать Logstash или преобразовывать их вручную, а потом индексировать в Elasticsearch. Не хватало функционала для возможности предварительной обработки/трансформации документов, они индексировались в исходном виде. Однако в версии Elasticsearch 5.x и далее была представлена функция узла поглощения данных, которая является легковесным решением для предварительной обработки и дополнения документов в Elasticsearch до индексирования.

Если Elasticsearch запущена в конфигурации по умолчанию, она будет работать как главный узел, узел данных и узел поглощения. Для отключения функции поглощения измените следующую настройку в файле `elasticsearch.yml`:

```
node.ingest: false
```

Узел поглощения может быть использован для предварительной обработки документов до момента фактического индексирования. Эта обработка происходит путем перехвата составных и индексирующих запросов, применения трансформаций данных и отправки документов обратно к API. С появлением новой функции поглощения Elasticsearch переняла часть функционала Logstash, ответственную за фильтрацию, и теперь мы можем выполнять обработку сырых логов и дополнять их внутри Elasticsearch.

Для предварительной обработки логов перед индексированием необходимо указать контейнер (который выполняет последовательность действий по трансформации входящего документа). Для этого мы попросту указываем параметр `pipeline` в индексе или в составном запросе, чтобы узел поглощения знал, какой контейнер использовать:

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

Определение контейнера

Контейнер определяет серию процессоров. Каждый процессор трансформирует документ в том или ином виде. Каждый процессор запускается в том порядке, в котором он определен в контейнере. Контейнер хранит два основных поля:

описание и список процессоров. Параметр `description` – необязательное поле, которое предназначено для хранения информации, связанной с использованием контейнера; с помощью параметра `processors` можно указать список процессоров для трансформации документа.

Типичная структура контейнера выглядит следующим образом:

```
{
  "description" : "...",
  "processors" : [ ... ]
}
```

Узел поглощения имеет около 20 встроенных процессоров, включая `gsub`, `grok`, `convert`, `remove`, `rename` и пр. для использования в создании контейнеров. Вместе со встроенными процессорами вы также можете применять доступные плагины поглощения, такие как `ingest attachment`, `ingest geo-ip`, `ingest user-agent`. По умолчанию они недоступны, но можно установить их как любой другой плагин Elasticsearch.

Ingest API

Узел поглощения предоставляет набор API, известный как Ingest API. Вы можете использовать эти API для определения, симуляции, удаления или поиска информации о контейнерах. Конечная точка данного API – `_ingest`.

API определения контейнера

Этот API используется для определения и добавления нового контейнера, а также для обновления имеющихся контейнеров.

Рассмотрим пример. Как видно в следующем коде, мы определили новый контейнер с названием `firstpipeline`, который переводит значения поля `message` в верхний регистр:

```
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"
      }
    }
  ]
}'
```

При создании контейнера вы можете указать несколько процессоров, а порядок их выполнения зависит от порядка, в котором они указаны в конфигурации. Рассмотрим это на примере. Как видно в следующем коде, мы создали новый контейнер с именем `secondpipeline`, который преобразует значения поля `"message"`

в верхний регистр и переименовывает поле "message" в "data". Он создает новое поле с названием "label" и значением testlabel:

```
curl -X PUT http://localhost:9200/_ingest/pipeline/secondpipeline -H
'content-type: application/json'
-d '{
    "description" : "uppercase the incoming value in the message field",
    "processors" : [
        {
            "uppercase" : {
                "field": "message",
                "ignore_failure" : true
            }
        },
        {
            "rename": {
                "field": "message",
                "target_field": "data",
                "ignore_failure" : true
            }
        },
        {
            "set": {
                "field": "label",
                "value": "testlabel",
                "override": false
            }
        }
    ]
}'
```

Используем второй контейнер для индексирования документа-образца:

```
curl -X PUT 'http://localhost:9200/myindex/mytype/1?pipeline=secondpipeline'
-H 'content-type: application/json' -d '{
    "message": "elk is awesome"
}'
```

А теперь мы получим тот же документ и утвердим трансформацию:

```
curl -X GET http://localhost:9200/myindex/mytype/1 -H
'content-type: application/json'
```

Ответ:

```
{
    "_index": "myindex",
    "_type": "mytype",
    "_id": "1",
    "_version": 1,
    "found": true,
    "_source": {
        "label": "testlabel",
        "data": "ELK IS AWESOME"
    }
}
```



Если отсутствует поле, используемое процессором, тогда процессор генерирует исключение и документ не будет проиндексирован. Для предотвращения исключений процессора мы можем присвоить параметру "ignore_failure" значение true.

API получения контейнера

Этот API используется для получения определения существующего контейнера. С его помощью можно найти детали определения одного контейнера или всех.

Команда для поиска определений всех контейнеров выглядит следующим образом:

```
curl -X GET http://localhost:9200/_ingest/pipeline -H
'content-type: application/json'
```

Для поиска определения существующего контейнера введите его идентификатор для получения .api контейнеров. В примере ниже вы можете увидеть поиск определения контейнера с названием `secondpipeline`:

```
curl -X GET http://localhost:9200/_ingest/pipeline/secondpipeline -H
'content-type: application/json'
```

API удаления контейнера

Данный API удаляет контейнеры согласно идентификатору или совпадению с универсальным символом. Далее вы увидите пример удаления контейнера с названием `firstpipeline`:

```
curl -X DELETE http://localhost:9200/_ingest/pipeline/firstpipeline -H
'content-type: application/json'
```

API симуляции контейнера

Используется для симуляции выполнения контейнера относительно набора документов, выбранных в теле запроса. Можно указать как существующий контейнер, так и определение необходимого контейнера. Для симуляции контейнера поглощения добавьте конечную точку "`_simulate`" к API контейнера.

Ниже приведен пример симуляции существующего контейнера:

```
curl -X POST
http://localhost:9200/_ingest/pipeline/firstpipeline/_simulate -H
'contenttype: application/json' -d '{
  "docs" : [
    { "_source": {"message":"first document"} },
    { "_source": {"message":"second document"} }
  ]
}'
```

Далее вы увидите пример симулированного запроса с определением контейнера в теле запроса:

```
curl -X POST http://localhost:9200/_ingest/pipeline/_simulate -H
'contenttype: application/json' -d '{
```

```
"pipeline" : {
  "processors": [
    {
      "join": {
        "field": "message",
        "separator": "-"
      }
    },
    "docs" : [
      { "_source": {"message":["first","document"]} }
    ]
  }
}'
```

Резюме

В этой главе мы поговорили об основах Logstash. Мы установли и настроили Logstash, чтобы можно было начать работу с контейнерами данных, а также разобрались в архитектуре Logstash.

Вы узнали об узле поглощения (ingest node), который впервые появился в версии Elastic 5.x и который сегодня можно использовать вместо выделенной установки Logstash. Вы научились применять узел поглощения для предварительной обработки документов до момента фактического индексирования, а также познакомились с различными API.

В следующей главе мы поговорим о разнообразных фильтрах в арсенале Logstash, которые ставят его в один ряд с другими фреймворками потоковой обработки в реальном и псевдореальном времени с нулевым кодированием.