Большие информационные системы генерируют огромное количество служебной информации, которую нужно где-то хранить. Я расскажу о том, как настроить хранилище для логов на базе Elasticsearch, Logstash и Kibana, которое часто называют ELK Stack. В это хранилище можно настроить отправку практически любых логов в разных форматах и большого объема.

Содержание:

Введение

Сразу хочу предупредить, что у меня нет большого опыта использования данного стека. Я некоторое время назад познакомился с ним, настроил одну систему. Посмотрел на нее, потестировал, мне понравилось. Настроил еще одну. В данный момент я занимаюсь ее изучением. В данной статье будут самые азы — как сделать быструю установку и базовую настройку, чтобы можно было собирать логи в Elasticsearch и смотреть их в Kibana.

Инструкция по установке будет полностью основана на официальной документации. Если вы хорошо читаете английские тексты, то можете использовать ее. Неудобство будет в том, что вся информация разрозненно располагается в разных статьях. Если вы первый раз настраиваете ELK Stack, то будет не очень просто сразу во всем разобраться. В своей статье я собрал в одном месте необходимый минимум для запуска Elasticsearch, Logstash, Kibana и агентов Filebeat и Winlogbeat для отправки логов с серверов.

Что такое ELK Stack

Расскажу своими словами о том, что мы будем настраивать. Ранее на своем сайте я уже рассказывал о централизованном сборе логов с помощью syslog-ng. Это вполне рабочее решение, хотя очевидно, что когда у тебя становится много логов, хранить их в текстовых файлах неудобно. Надо как-то обрабатывать и выводить в удобном виде для просмотра. Именно эти проблемы и решает представленный стек программ:

- **Elasticsearch** используется для хранения, анализа, поиска по логам.
- **Kibana** представляет удобную и красивую web панель для работы с логами.
- **Logstash** сервис для сбора логов и отправки их в Elasticsearch. В самой простой конфигурации можно обойтись без него и отправлять логи напрямую в еластик. Но с logstash это делать удобнее.
- **Beats** — агенты для отправки логов в Logstash. Они бывают разные. Я буду использовать **Filebeat** для отправки данных из текстовых логов linux и **Winlogbeat** для отправки логов из журналов Windows систем.

К этой связке еще добавляется Nginx, который проксирует соединения на Kibana. В самом простом случае он не нужен, но с ним удобнее. Можно, к примеру, добавить авторизацию или ssl сертификат, в nginx удобно управлять именем домена. В случае большой нагрузки, разные службы разносятся по разным серверам или кластерам. В своем примере я все установлю на один сервер. Схематично работу данной системы можно изобразить вот так:

Начнем по этапам устанавливать и настраивать все компоненты нашей будущей системы хранения и обработки логов различных информационных систем.

Установка Java 8

> Приведенная ниже инструкция по установке Java 8 немного устарела в связи с изменением лицензии Oracle. Про установку Oracle Java на Centos или Ubuntu читайте отдельную, актуальную на сегодняшний день статью.

Все компоненты системы, за исключением агентов на серверах, написаны на java, поэтому нашу настройку начнем с установки Oracle Java 8. Эта версия выбрана, потому что поддерживается всеми версиями Elasticsearch. Подробнее о совместимости программных продуктов смотрите в отдельной таблице из документации.

Если у вас еще нет своего сервера с CentOS 8, то рекомендую мои материалы на эту тему:

- Установка CentOS 8.
- Настройка CentOS.

Если у вас еще не настроен сервер с Debian, рекомендую мои материалы на эту тему:

- Установка Debian на сервер
- Базовая настройка Debian после установки

CentOS

Идем на страницу https://www.oracle.com/technetwork/java/javase/downloads/jre8-downloads-2133155.html, жмем плашку «Accept License Agreement» и скачиваем файл ***-linux-x64.rpm**. Чтобы его загрузить, придется зарегистрироваться и авторизоваться на сайте oracle.com.

После этого копируем любым удобным для вас способом пакет с java на сервер. Я обычно это делаю через scp. Устанавливаем Java 8.

```
# dnf localinstall jre-*
```

Проверим установленную версию:

```
java version "1.8.0_231"
Java(TM) SE Runtime Environment (build 1.8.0_231-b11)
Java HotSpot(TM) 64-Bit Server VM (build 25.231-b11, mixed mode)
```

Ubuntu / Debian

Идем на страницу https://www.oracle.com/technetwork/java/javase/downloads/jre8-downloads-2133155.html, жмем плашку «Accept License Agreement» и скачиваем файл *-linux-x64.tar.gz. Чтобы его загрузить, придется зарегистрироваться и авторизоваться на сайте oracle.com.

После загрузки любым удобным для вас способом скопируйте файл на целевой сервер, куда будем устанавливать Java 8. Я для этого использую scp. Копирую файл в домашнюю директорию /root. Создаем директорию для java и распаковываем туда бинарники.

```
# mkdir /usr/lib/jvm
# tar -zxf /root/jre-8u231-linux-x64.tar.gz -C /usr/lib/jvm
```

Java машина должна расположиться в директории */usr/lib/jvm/jre1.8.0_231*. Теперь нам необходимо создать символьные ссылки на установленную версию java. Делаем это с помощью **update-alternatives**.

```
# update-alternatives --install "/usr/bin/java" "java" "/usr/lib/jvm/jre1.8.0_231/bin/java" 1500
# update-alternatives --install "/usr/bin/javaws" "javaws" "/usr/lib/jvm/jre1.8.0_231/bin/javaws" 1500
```

Зададим переменную JAVA_HOME, которую используют некоторые приложения. Для этого добавляем в файл */etc/environment* следующую строку.

```
JAVA_HOME="/usr/lib/jvm/jre1.8.0_231"
```

Сохраняем его и применяем изменение.

```
# source /etc/environment
```

Теперь проверим, что у нас получилось.

```
# java -version
# echo $JAVA_HOME
```

Установка Elasticsearch

Устанавливаем ядро системы по сбору логов — Elasticsearch. Его установка очень проста за счет готовых пакетов под все популярные платформы.

Centos 7 / 8

Копируем публичный ключ репозитория:

```
# rpm --import https://artifacts.elastic.co/GPG-KEY-elasticsearch
```

Подключаем репозиторий Elasticsearch:

```
# mcedit /etc/yum.repos.d/elasticsearch.repo
```

```
[elasticsearch-7.x]
name=Elasticsearch repository for 7.x packages
baseurl=https://artifacts.elastic.co/packages/7.x/yum
gpgcheck=1
gpgkey=https://artifacts.elastic.co/GPG-KEY-elasticsearch
enabled=1
autorefresh=1
```

Приступаем к установке еластика:

```
# yum install elasticsearch
```

В в завершении установки добавим elasticsearch в автозагрузку и запустим его с дефолтными настройками:

```
# systemctl daemon-reload
# systemctl enable elasticsearch.service
# systemctl start elasticsearch.service
```

Проверяем, запустилась ли служба:

```
# systemctl status elasticsearch.service
```

Ubuntu/Debian

Копируем себе публичный ключ репозитория:

```
# wget -qO - https://artifacts.elastic.co/GPG-KEY-elasticsearch | apt-key add -
```

Если у вас нет пакета apt-transport-https, то надо установить:

```
# apt install apt-transport-https
```

Добавляем репозиторий Elasticsearch в систему:

```
echo "deb https://artifacts.elastic.co/packages/7.x/apt stable main" | sudo tee -a /etc/apt/sources.list.d/elastic-7.x.list
```

Устанавливаем Elasticsearch на Debian или Ubuntu:

```
# apt update && apt-get install elasticsearch
```

После установки добавляем elasticsearch в автозагрузку и запускаем.

```
# systemctl daemon-reload
# systemctl enable elasticsearch.service
# systemctl start elasticsearch.service
```

Проверяем, запустился ли он:

```
# systemctl status elasticsearch.service
```

Если все в порядке, то переходим к настройке Elasticsearch.

Настройка Elasticsearch

Настройки Elasticsearch находятся в файле */etc/elasticsearch/elasticsearch.yml*. На начальном этапе нас будут интересовать следующие параметры:

```
path.data: /var/lib/elasticsearch # директория для хранения данных
network.host: 127.0.0.1 # слушаем только локальный интерфей
```

По-умолчанию Elasticsearch слушает все имеющиеся сетевые интерфейсы. Нам это не нужно, так как данные в него будет передавать logstash, который будет установлен локально. Обращаю отдельное внимание на параметр для директории с данными. Чаще всего они будут занимать значительное место, иначе зачем нам Elasticsearch :) Подумайте заранее, где вы будете хранить логи. Все остальные настройки я оставляю дефолтными.

После изменения настроек, надо перезапустить службу:

```
# systemctl restart elasticsearch.service
```

Смотрим, что получилось:

```
# netstat -tulnp | grep 9200
tcp6       0      0 127.0.0.1:9200          :::*                    LISTEN      14130/java
```

Elasticsearch повис на локальном интерфейсе. Причем я вижу, что он слушает ipv6, а про ipv4 ни слова. Но его он тоже слушает, так что все в порядке. Переходим к установке kibana.

Установка Kibana

Дальше устанавливаем web панель Kibana для визуализации данных, полученных из Elasticsearch. Тут тоже ничего сложного, репозиторий и готовые пакеты есть под все популярные платформы. Репозитории и публичный ключ для установки Kibana будут такими же, как в установке Elasticsearch. Но я еще раз все повторю для тех, кто будет устанавливать только Kibana, без всего остального. Это продукт законченный и используется не только в связке с Elasticsearch.

Centos

Импортируем ключ репозитория:

```
# rpm --import https://artifacts.elastic.co/GPG-KEY-elasticsearch
```

Добавляем конфиг репозитория:

```
# mcedit /etc/yum.repos.d/kibana.repo
```

```
[kibana-7.x]
name=Kibana repository for 7.x packages
baseurl=https://artifacts.elastic.co/packages/7.x/yum
gpgcheck=1
gpgkey=https://artifacts.elastic.co/GPG-KEY-elasticsearch
enabled=1
```

```
autorefresh=1
```

Запускаем установку Kibana:

```
# yum install kibana
```

Добавляем Кибана в автозагрузку и запускаем:

```
# systemctl daemon-reload
# systemctl enable kibana.service
# systemctl start kibana.service
```

Проверяем состояние запущенного сервиса:

```
# systemctl status kibana.service
```

По-умолчанию, Kibana слушает порт 5601. Только не спешите его проверять после запуска. Кибана стартует долго. Подождите примерно минуту и проверяйте.

```
# netstat -tulnp | grep 5601
tcp        0      0 127.0.0.1:5601          0.0.0.0:*               LISTEN      27401/node
```

Ubuntu/Debian

Установка Kibana на Debian или Ubuntu такая же, как и на центос — подключаем репозиторий и ставим из deb пакета. Добавляем публичный ключ:

```
# wget -qO - https://artifacts.elastic.co/GPG-KEY-elasticsearch | apt-key add -
```

Добавляем рпозиторий Kibana:

```
# echo "deb https://artifacts.elastic.co/packages/7.x/apt stable main" | tee -a /etc/apt/sources.list.d/elastic-7.x.list
```

Запускаем установку Kibana:

```
# apt update && apt install kibana
```

Добавляем Кибана в автозагрузку и запускаем:

```
# systemctl daemon-reload
# systemctl enable kibana.service
# systemctl start kibana.service
```

Проверяем состояние запущенного сервиса:

```
# systemctl status kibana.service
```

По-умолчанию, Kibana слушает порт 5601. Только не спешите его проверять после запуска. Кибана стартует долго. Подождите примерно минуту и проверяйте.

```
# netstat -tulnp | grep 5601
tcp        0      0 127.0.0.1:5601          0.0.0.0:*               LISTEN      27401/node
```

Настройка Kibana

Файл с настройками Кибана располагается по пути — */etc/kibana/kibana.yml*. На начальном этапе можно вообще ничего не трогать и оставить все как есть. По-умолчанию kibana слушает только localhost и не позволяет подключаться удаленно. Это нормальная ситуация, если у вас будет на этом же сервере установлен nginx в качестве reverse proxy, который будет принимать подключения и проксировать их в кибана. Так и нужно делать в продакшене, когда системой будут пользоваться разные люди из разных мест. С помощью nginx можно будет разграничивать доступ, использовать сертификат, настраивать нормальное доменное имя и т.д.

Если же у вас это тестовая установка, то можно обойтись без nginx. Для этого надо разрешить Кибана слушать внешний интерфейс и принимать подключения. Измените параметр **server.host**, указав ip адрес сервера, например вот так:

```
server.host: "10.1.4.114"
```

Если хотите, чтобы она слушала все интерфейсы, укажите в качестве адреса 0.0.0.0. После этого Kibana надо перезапустить:

```
# systemctl restart kibana.service
```

Теперь можно зайти в веб интерфейс по адресу http://10.1.4.114:5601.

Можно продолжать настройку и тестирование, а когда все будет закончено, запустить nginx и настроить проксирование. Я настройку nginx оставлю на самый конец. В процессе настройки буду подключаться напрямую к Kibana.

Установка и настройка Logstash

Logstash устанавливается так же просто, как Elasticsearch и Kibana, из того же репозитория. Не буду еще раз показывать, как его добавить. Просто установим его и добавим в автозагрузку.

Установка Logstash в Centos 7:

```
# yum install logstash
```

Установка Logstash в Debian/Ubuntu:

```
# apt install logstash
```

Добавляем logstash в автозагрузку:

```
# systemctl enable logstash.service
```

Запускать пока не будем, надо его сначала настроить. Основной конфиг logstash лежит по адресу */etc/logstash/logstash.yml*. Я его трогать не буду, а все настройки буду по смыслу разделять по разным конфигурационным файлам в директории */etc/logstash/conf.d*. Создаем первый конфиг *input.conf*, который будет описывать прием информации с beats агентов.

```
input {
  beats {
    port => 5044
  }
}
```

Тут все просто. Указываю, что принимаем информацию на 5044 порт. Этого достаточно. Если вы хотите использовать ssl сертификаты для передачи логов по защищенным соединениям, здесь добавляются параметры ssl. Я буду собирать данные из закрытого периметра локальной сети, у меня нет необходимости использовать ssl.

Теперь укажем, куда будем передавать данные. Тут тоже все относительно просто. Рисуем конфиг *output.conf*, который описывает передачу данных в Elasticsearch.

```
output {
        elasticsearch {
            hosts    => "localhost:9200"
            index    => "nginx-%{+YYYY.MM.dd}"
        }
        #stdout { codec => rubydebug }
}
```

Здесь все просто — передавать все данные в elasticsearch под указанным индексом с маской в виде даты. Насколько я понял, разбивка индексов по дням и по типам данных удобна с точки зрения управления данными. Потом легко будет выполнять очистку данных по этим индексам. Я закомментировал последнюю строку. Она отвечает за логирование. Если ее включить, то все поступающие данные logstash будет отправлять дополнительно в системный лог. В centos это */var/log/messages*. Используйте только во время отладки, иначе лог быстро разрастется дублями поступающих данных.

Остается последний конфиг с описание обработки данных. Тут начинается небольшая уличная магия, в которой я разбирался некоторое время. Расскажу ниже. Рисуем конфиг *filter.conf*.

```
filter {
 if [type] == "nginx_access" {
    grok {
        match => { "message" => "%{IPORHOST:remote_ip} - %{DATA:user} \[%{HTTPDATE:access_time}\] \"%{WORD:http_method}
%{DATA:url} HTTP/%{NUMBER:http_version}\" %{NUMBER:response_code} %{NUMBER:body_sent_bytes} \"%{DATA:referrer}\"
\"%{DATA:agent}\"" }
    }
  }
  date {
        match => [ "timestamp" , "dd/MMM/YYYY:HH:mm:ss Z" ]
  }
  geoip {
         source => "remote_ip"
         target => "geoip"
         add_tag => [ "nginx-geoip" ]
  }
}
```

Первое, что делает этот фильтр, парсит логи nginx с помощью grok, если указан соответствующий тип логов, и выделяет из лога ключевые данные, которые записывает в определенные поля, чтобы потом с ними было удобно работать. Сначала я не понял, зачем это нужно. В документации к filebeat хорошо описаны модули, идущие в комплекте, которые все это и так уже умеют делать из коробки, нужно только подключить соответствующий модуль.

Оказалось, что модули filebeat работают только в том случае, если вы отправляете данные напрямую в elasticsearch. На него вы тоже ставите соответсвующий плагин и получаете отформатированные данные. Но у нас работает промежуточное звено logstash, который принимает данные. С ним, как я понял, плагины filebeat не работают, поэтому приходится отдельно в logstash парсить данные. Это не очень сложно, но тем не менее. Как я понял, это плата за удобства, которые дает logstash. Если у вас много разрозненных данных, то отправлять их напрямую в elasticsearch не так удобно, как с использованием предобработки в logstash. Если я не прав, прошу меня поправить. Я так понял этот момент.

Для фильтра grok, который использует logstash, есть удобный дебаггер, где можно посмотреть, как будут парситься ваши данные. Покажу на примере

одной строки из конфига nginx. Например, возьмем такую строку из лога nginx:

```
180.163.220.100 - travvels.ru [05/Sep/2018:14:45:52 +0300] "GET /assets/galleries/26/1.png HTTP/1.1" 304 0
"https://travvels.ru/ru/glavnaya/" "Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/50.0.2661.102 Safari/537.36"
```

И посмотрим, как ее распарсит правило grok, которое я использовал в конфиге выше.

```
%{IPORHOST:remote_ip} - %{DATA:user} \[%{HTTPDATE:access_time}\] \"%{WORD:http_method} %{DATA:url}
HTTP/%{NUMBER:http_version}\" %{NUMBER:response_code} %{NUMBER:body_sent_bytes} \"%{DATA:referrer}\" \"%{DATA:agent}\"
```

Собственно, результат вы можете сами увидеть в дебаггере. Фильтр распарсит лог и на выходе сформирует json, где каждому значению будет присвоено свое поле, по которому потом удобно будет в еластике строить отчеты и делать выборки. Только не забывайте про формат логов. Приведенное мной правило соответствует дефолтному формату main логов в nginx. Если вы каким-то образом модифицировали формат логов, внесите изменения в grok фильтр.

Надеюсь понятно объяснил работу этого фильтра. Вы можете таким образом парсить любые логи и передавать их в еластикс. Потом на основе этих данных строить отчеты, графики, дашборды. Я планирую распарсить как мне нужно почтовые логи postfix и dovecot.

Дальше используется модуль **date** для того, чтобы выделять дату из поступающих логов и использовать ее в качестве даты документа в elasticsearch. Делается это для того, чтобы не возникало путаницы, если будут задержки с доставкой логов. В системе сообщения будут с одной датой, а внутри лога будет другая дата. Неудобно разбирать инциденты.

В конце я использую **geoip** фильтр, который на основе ip адреса, который мы получили ранее с помощью фильтра grok и записали в поле remote_ip, определяет географическое расположение. Он добавляет новые метки и записывает туда географические данные. Для его работы мне было достаточно описать его в конфиге и скачать базу адресов в папку */etc/logstash*. Раньше она была доступна по прямой ссылке, но с 30-го декабря 2019 года правила изменились. База по прежнему доступна бесплатно, но для загрузки нужна регистрация на сайте сервиса. Регистрируемся и качаем отсюда — https://dev.maxmind.com/geoip/geoip2/geolite2/#Download_Access. Передаем на сервер, распаковываем и копируем в /etc/logstash.

```
# gunzip GeoLite2-City.mmdb.gz
```

Закончили настройку logstash. Запускаем его:

```
# systemctl start logstash.service
```

Можете проверить на всякий случай лог */var/log/logstash/logstash-plain.log*, чтобы убедиться в том, что все в порядке. Теперь настроим агенты для отправки данных.

Установка Filebeat для отправки логов в Logstash

Установим первого агента Filebeat на сервер с nginx для отправки логов веб сервера на сервер с ELK. Ставить можно как из общего репозитория, который мы подключали ранее, так и по отдельности пакеты. Как ставить — решать вам. В первом случае придется на все хосты добавлять репозиторий, но зато потом удобно обновлять пакеты. Если подключать репозиторий не хочется, можно просто скачать пакет и установить его.

Ставим на Centos.

```
# curl -L -O https://artifacts.elastic.co/downloads/beats/filebeat/filebeat-7.5.1-x86_64.rpm
# rpm -vi filebeat-7.5.1-x86_64.rpm
```

В Debian/Ubuntu ставим так:

```
# curl -L -O https://artifacts.elastic.co/downloads/beats/filebeat/filebeat-7.5.1-amd64.deb
# dpkg -i filebeat-7.5.1-amd64.deb
```

Или просто:

```
# yum install filebeat
# apt install filebeat
```

После установки рисуем примерно такой конфиг */etc/filebeat/filebeat.yml* для отправки логов в logstash.

```
filebeat.inputs:
- type: log
```

```
  enabled: true
  paths:
      - /var/log/nginx/*-access.log
  fields:
    type: nginx_access
  fields_under_root: true
  scan_frequency: 5s

- type: log
  enabled: true
  paths:
      - /var/log/nginx/*-error.log
  fields:
    type: nginx_error
  fields_under_root: true
  scan_frequency: 5s

output.logstash:
  hosts: ["10.1.4.114:5044"]

xpack.monitoring:
  enabled: true
  elasticsearch:
    hosts: ["http://10.1.4.114:9200"]
```

Некоторые пояснения к конфигу, так как он не совсем дефолтный и минималистичный. Я его немного модифицировал для удобства. Во-первых, я разделил логи access и error с помощью отдельного поля type, куда записываю соответствующий тип лога: nginx_access или nginx_error. В зависимости от типа меняются правила обработки в logstash. Плюс, я включил мониторинг и для этого указал адрес elastichsearch, куда filebeat передает данные мониторинга напрямую. Показываю это для вас просто с целью продемонстрировать возможность. У меня везде отдельно работает мониторинг на zabbix, так что большого смысла в отдельном мониторинге нет. Но вы посмотрите на него, возможно вам он пригодится. Чтобы мониторинг работал, его надо активировать в соответствующем разделе в Kibana — Monitoring. И не забудьте запустить elasticsearch на внешнем интерфейсе. В первоначальной настройке я указал слушать только локальный интерфейс.

Запускаем filebeat и добавляем в автозагрузку.

```
# systemctl start filebeat
# systemctl enable filebeat
```

Проверяйте лог по адресу */var/log/filebeat/filebeat*. Он весьма информативен. Если все в порядке, увидите список всех логов в директории /var/log/nginx, которые нашел filebeat и начал готовить к отправке. Если все сделали правильно, то данные уже потекли в elasticsearch. Мы их можем посмотреть в Kibana. Для этого открываем web интерфейс и переходим в раздел Discover. Так как там еще нет индекса, нас перенаправит в раздел Managemet, где мы сможем его добавить.

Вы должны увидеть индекс, который начал заливать logstash в elasticsearch. В поле **Index pattern** введите *nginx-** и нажмите **Next Step**. На следующем этапе выберите имя поля для временного фильтра. У вас будет только один вариант — @timestamp, выбирайте его и жмите **Create Index Pattern**.

Новый индекс добавлен. Теперь при переходе в раздел Discover, он будет открываться по-умолчанию со всеми данными, которые в него поступают.

Получением логов с linux серверов настроили. Теперь сделаем то же самое для журналов windows.

Установка и настройка Winlogbeat

Для настройки централизованного сервера сбора логов с Windows серверов, установим сборщика системных логов winlogbeat. Для этого скачаем его и установим в качестве службы. Идем на страницу загрузок и скачиваем версию под вашу архитектуру — 32 или 64 бита.

Распаковываем скачанный архив и копируем его содержимое в директорию *C:\Program Files\Winlogbeat*. Сразу же создаем там папку logs для хранения логов самого winlogbeat.

Запускаем консоль Powershell от администратора и выполняем команды:

```
PS C:\Users\Administrator> cd 'C:\Program Files\Winlogbeat'
PS C:\Program Files\Winlogbeat> .\install-service-winlogbeat.ps1
```

У меня был такой вывод:

```
Status   Name               DisplayName
------   ----               -----------
Stopped  winlogbeat         winlogbeat
```

Служба установилась и в данный момент она остановлена. Правим конфигурационный файл *winlogbeat.yml*. Я его привел к такому виду:

```
winlogbeat.event_logs:
  - name: Application
    ignore_older: 72h
  - name: Security
  - name: System

tags: ["winsrv"]

output.logstash:
  hosts: ["10.1.4.114:5044"]

logging.level: info
logging.to_files: true
logging.files:
  path: C:/Program Files/Winlogbeat/logs
  name: winlogbeat
```

```
  keepfiles: 7

xpack.monitoring:
  enabled: true
  elasticsearch:
    hosts: ["http://10.1.4.114:9200"]
```

В принципе, по тексту все понятно. Я беру 3 системных лога **Application**, **Security**, **System** (для русской версии используются такие же названия) и отправляю их на сервер с logstash. Настраиваю хранение логов и включаю мониторинг по аналогии с filebeat. Отдельно обращаю внимание на tags: [«winsrv»]. Этим тэгом я помечаю все отправляемые сообщения, чтобы потом их обработать в logstash и отправить в elasticsearch с отдельным индексом. Я не смог использовать поле type, по аналогии с filebeat, потому что в winlogbeat в поле type жестко прописано **wineventlog** и изменить его нельзя. Если у вас данные со всех серверов будут идти в один индекс, то можно tag не добавлять, а использовать указанное поле type для сортировки. Если же вы данные с разных среверов хотите помещать в разные индексы, то разделяйте их тэгами.

После настройки можно запустить службу winlogbeat, которая появится в списке служб windows. Но для того, чтобы логи виндовых журналов пошли в elasticsearch не в одну кучу вместе с nginx логами, нам надо настроить для них отдельный индекс в logstash в разделе output. Идем на сервер с logstash и правим конфиг *output.conf*.

```
output {
    if [type] == "nginx_access" {
        elasticsearch {
            hosts     => "localhost:9200"
            index    => "nginx-%{+YYYY.MM.dd}"
        }
    }
    else if [type] == "nginx_error" {
        elasticsearch {
            hosts     => "localhost:9200"
            index    => "nginx-%{+YYYY.MM.dd}"
        }
    }
    else if "winsrv" in [tags] {
```

```
        elasticsearch {
            hosts     => "localhost:9200"
            index    => "winsrv-%{+YYYY.MM.dd}"
        }
    }
    else {
        elasticsearch {
            hosts     => "localhost:9200"
            index    => "unknown_messages"
        }
    }
        #stdout { codec => rubydebug }
}
```

Думаю, по тексту понятен смысл. Я разделил по разным индексам логи nginx, системные логи виндовых серверов и добавил отдельный индекс unknown_messages, в который будет попадать все то, что не попало в предыдущие. Это просто задел на будущее, когда конфигурация будет более сложная, можно будет ловить сообщения, которые по какой-то причине не попали ни в одно из приведенных выше правил.

Я не смог в одно правило поместить оба типа nginx_error и nginx_access, потому что не понял сходу, как это правильно записать, а по документации уже устал лазить, выискивать информацию. Так получился рабочий вариант. После этого перезапустил logstash, подождал немного, пока появятся новые логи на виндовом сервере, зашел в kibana и добавил новые индексы. Напомню, что делается это в разделе **Management -> Kibana -> Index Patterns**.

Можно идти в раздел Discover и просматривать логи с Windows серверов.

У меня без проблем все заработало на серверах с русской версией Windows. Все логи, тексты в сообщениях на русском языке нормально обрабатываются и отображаются. Проблем не возникло нигде.

На этом я закончил настройку стека Elasticsearch, Logstash, Kibana, Filebeat и Winlogbeat. Описал основной функционал по сбору логов. Дальше с ними уже

можно работать по ситуации — строить графики, отчеты, собирать дашборды и т.д. Возможно, опишу это отдельно.

Проксирование подключений к Kibana через Nginx

Я не буду подробно рассказывать о том, что такое проксирование в nginx. У меня есть отдельная статья на эту тему — настройка proxy_pass в nginx. Приведу просто пример конфига для передачи запросов с nginx в kibana. Я рекомендую использовать ssl сертификаты для доступа к Kibana. Даже если в этом нет объективной необходимости, надоедают уведомления браузеров о том, что у вас небезопасное подключение. Подробная инструкция по установке, настройке и использованию ssl в nginx так же есть у меня на сайте — настройка web сервера nginx. Все подробности настройки nginx смотрите там.

Вот примерный конфиг nginx для проксирования запросов к Kibana с ограничением доступа по паролю:

```
server {
        listen 443;

        server_name kibana.site.ru;
        ssl_certificate /etc/letsencrypt/live/kibana.site.ru/fullchain.pem;
        ssl_certificate_key /etc/letsencrypt/live/kibana.site.ru/privkey.pem;

        location / {
                auth_basic "Restricted Access";
                auth_basic_user_file /etc/nginx/htpasswd.kibana;
                proxy_pass http://localhost:5601;
                proxy_http_version 1.1;
                proxy_set_header Upgrade $http_upgrade;
                proxy_set_header Connection 'upgrade';
                proxy_set_header Host $host;
                proxy_cache_bypass $http_upgrade;
        }
}
```

Создаем файл для пользователей и паролей:

```
# htpasswd -c /etc/nginx/htpasswd.kibana kibanauser
```

Если утилиты htpasswd нет в системе, то установите ее:

```
# yum install -y httpd-tools
```

После этого выйдет запрос на ввод пароля. С помощью приведенной выше команды мы создали файл с информацией о пользователе и пароле kibanauser для ограничения доступа к web панели кибана.

Заключение

> Не понравилась статья и хочешь научить меня администрировать? Пожалуйста, я люблю учиться. Комментарии в твоем распоряжении. Расскажи, как сделать правильно!

Я постарался рассказать подробно и понятно о полной настройке ELK Stack. Информацию в основном почерпнул в официальной документации. Мне не попалось более ли менее подробных статей ни в рунете, ни в буржунете, хотя искал я основательно. Вроде бы такой популярный и эффективный инструмент, но статей больше чем просто дефолтная установка я почти не видел. Буквально одна на хабре попалась с какой-то более ли менее кастомизацией.

Какие-то проверенные моменты я не стал описывать в статье, так как посчитал их неудобными и не стал использовать сам. Например, если отказаться от logstash и отправлять данные с beats напрямую в elasticsearch, то на первый взгляд все становится проще. Штатные модули beats сами парсят вывод, устанавливают готовые визуализации и дашборды в Kibana. Вам остается только зайти и любоваться красотой :) Но на деле все выходит не так красиво, как хотелось бы. Кастомизация конфигурации усложняется. Изменение полей в логах приводит к более сложным настройкам по вводу этих изменений в систему. Все настройки поступающей информации переносятся на каждый beats, изменяются в конфигах отдельных агентов. Это неудобно.

В то же время, при использовании logstash, вы просто отправляете данные со всех beats на него и уже в одном месте всем управляете, распределяете индексы, меняете поля и т.д. Все настройки перемещаются в одно место. Это более удобный подход. Плюс, при большой нагрузке вы можете вынести

logstash на отдельную машину.

Я не рассмотрел в своей статье такие моменты как создание визуализаций и дашбордов в Кибана. Возможно, сделаю это отдельно, так как материал уже и так получился объемный, я устал писать эту статью :) Надо еще подумать над подачей, много картинок надо, чтобы было понятно. Там все не очень наглядно, так что надо внимательно разбираться.

Так же я не рассмотрел такой момент. Logstash может принимать данные напрямую через syslog. Вы можете, к примеру, в nginx настроить отправку логов в syslog, минуя файлы и beats. Это может быть более удобно, чем описанная мной схема. Особенно это актуально для сбора логов с различных сетевых устройств, на которые невозможно поставить агента. Syslog поток так же можно парсить на ходу с помощью grok. Отдельно надо рассмотреть автоочистку старых индексов в elasticsearch.

Подводя итог скажу, что с этой системой хранения логов нужно очень вдумчиво и внимательно разбираться. С наскока ее не осилить. Чтобы было удобно пользоваться, нужно много всего настроить. Я описал только немного кастомизированный сбор данных, их визуализация — отдельный разговор. Сам я в настоящее время изучаю систему, поэтому буду рад любым советам, замечаниям, интересным ссылкам и всему, что поможет освоить тему.

Все статьи раздела elk stack — https://serveradmin.ru/category/elk-stack/.