Стек ELK - современный инструментарий мониторинга сети

Многолетнее использование rsyslog для хранения log-сообщений сетевого оборудования, и loganalyzer для их отображения, обозначило необходимость поиска, соответствующей современным требованиям, альтернативы. В результате анализа различных решений был сделан выбор в пользу open-source компонентов стека ELK.

Основными его элементами являются: Elasticsearch (search engine – для хранения и быстрого поиска структурированных данных), Logstash (collector – для приема в различном формате данных, их фильтрации и преобразования, и последующей отправки в различные базы данных) и Kibana (инструмент для визуализации), которые и составляют акроним ELK.

Подробное описание работы этих компонентов выходит за рамки данной статьи. Интересующая информация по стеку очень хорошо представлена на официальном сайте, в т.ч. в виде подробной документации, а также в огромном количестве статей и публикаций.

Расмотрим вариант установки ELK, его конфигурации, приема log-сообщений от сетевого оборудования, их обработки, создания графиков и таблиц, единого дашборда для отображения.

В качестве хоста для разворачивания стека был выбран Ubuntu Server 18.04 LTS, недавно обновленная до 7.7.0 версия ELK, сетевое оборудование Juniper. В отношении последнего, сразу оговорюсь, взято для примера). На самом деле обрабатывать сообщения, имеющие текстовый формат, в Logstash можно любые (любых вендоров).

Формат изложения в данной статье имеет практическую направленность, пошагового разбит по пунктам, содержит команды, примеры конфигурации, наглядные скриншоты по созданию графиков и дашборда.

Итак...

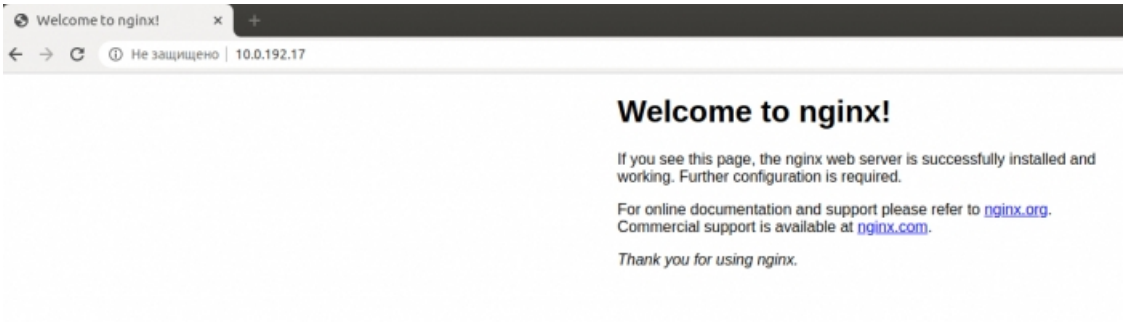
1) Обновляем пакеты операционной системы.

```
sudo apt-get update
sudo apt-get upgrade
```

2) Устанавливаем WEB-сервер nginx.

```
sudo apt install nginx -y
```

Проверяем nginx, введя в адресной строке браузера адрес хоста, на котором он установлен.



3) Устанавливаем Java.

```
sudo apt install default-jre -y
sudo apt install default-jdk -y
```

4) Установку Elasticsearch выполняем из репозитория (при этом следует отметить, что существует несколько вариантов установки).

Загрузка и установка ключей.

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

Установка пакета apt-transport-https.

```
sudo apt-get install apt-transport-https
```

Добавление репозитория в sources.list.

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

Установка Elasticsearch.

sudo apt-get update && sudo apt-get install elasticsearch

Редактирование файла конфигурации Elasticsearch.

sudo nano /etc/elasticsearch/elasticsearch.yml

В нем необходимо заменить #network.host: 192.168.0.1 на network.host: localhost

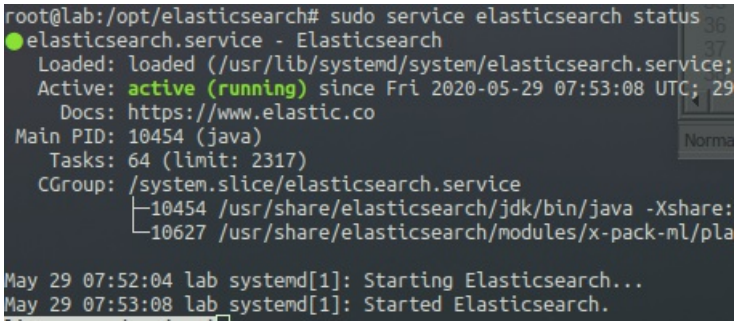
Было:
#network.host: 192.168.0.1
Стало:
network.host: localhost

Запуск Elasticsearch.

sudo service elasticsearch start

Проверка статуса Elasticserach.

sudo service elasticsearch status

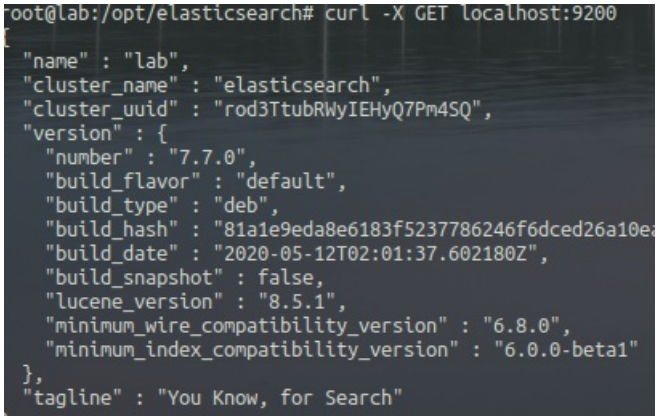


Включение юнита Elasticserach в автозагрузку.

sudo systemctl enable elasticsearch

Проверка статуса через XGET:

curl -X GET localhost:9200



5) Установка Kibana из репозитория, добавленного в sources.list ранее.

sudo apt-get update && sudo apt-get install kibana

Изменение конфигурации для kibana:

sudo nano /etc/kibana/kibana.yml

Было:

#server.port: 5601

Стало:

server.port: 5601

Было:

#server.host: "localhost"

Стало:

server.host: "localhost"

Было:

#elasticsearch.hosts: ["http://localhost:9200"]

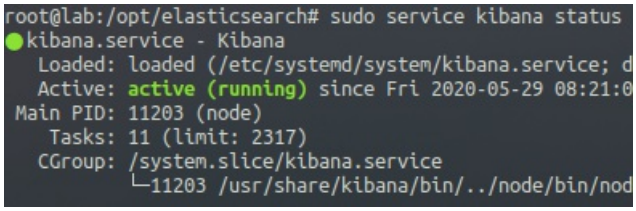
Стало:

elasticsearch.hosts: ["http://localhost:9200"]

Запуск Kibana.

sudo service kibana start

sudo service kibana status



Включение юнита Kibana в автозагрузку.

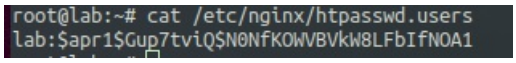
sudo systemctl enable kibana

6) Создание учётной записи и проверка пользователя nginx (Важно! Opensource-ные пакеты ELK не предоставляют инструментов безопасности).

echo "lab:`openssl passwd -apr1`" | sudo tee -a /etc/nginx/htpasswd.users

Проверяем наличие пользователя в nginx:

cat /etc/nginx/htpasswd.users



7) Создаем файл с виртуальным сайтом для веб-сервера Nginx, выполнив команду.

sudo nano /etc/nginx/sites-available/elk

8) В этот файл добавляем следующую информацию:

```
server {
listen 80;
server_name <внешний IP-адрес веб-сервера>;
auth_basic "Restricted Access";
auth_basic_user_file /etc/nginx/htpasswd.users;
location / {
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;
}
}
```

Сохраняем изменения, используя сочетание клавиш CTRL+O, и завершаем редактирование, используя сочетание клавиш CTRL+X.

9) Активируем новую конфигурацию Nginx, выполнив команду.

```
sudo ln -s /etc/nginx/sites-available/elk /etc/nginx/sites-enabled/
```

10) Перезагружаем Kibana, выполнив команду.

```
sudo systemctl restart kibana
```

11) Перезагружаем веб-сервер Nginx, выполнив команду.

```
sudo systemctl restart nginx
```

12) Проверяем синтаксис конфигурационного файла nginx не отсутствие ошибок, выполнив команду.

```
sudo nginx -t
```

13) Установливаем Logstash.

```
sudo apt install logstash
```

14) Создаем конфигурационный файл для Logstash.

```
sudo nano /etc/logstash/conf.d/networklogs.conf
```

Вносим в него данные.

```
input {
udp {
port => 1514
type => "syslog"
tags => junos
add_field => {"_index_name" => "syslog"}
}
}

filter {
grok{
patterns_dir => ["/usr/share/logstash/vendor/bundle/jruby/2.5.0/gems/logstash-patterns-core-4.1.2/patterns"]
match => {
"message" => [
"%{DATESTAMP:time_syslog} %{HOSTNAME:hostname} %{MSGSOURCE:msg_source}: %%{FACILITY:facility}-%{SEVERITY:severity}:
%{MSGTEXT:msgtext}",
"%{DATESTAMP:time_syslog} %{HOSTNAME:hostname} %{MSGSOURCE:msg_source}: %%{FACILITY:facility}-%{SEVERITY:severity}-
%{TAG:tag}: %{MSGTEXT:msgtext}",
"%{DATESTAMP:time_syslog} %{HOSTNAME:hostname} : %%{FACILITY:facility}-%{SEVERITY:severity}: %{MSGSOURCE:msgsource}
%{TAG:tag}: %{MSGTEXT:msgtext}",
"%{DATESTAMP:time_syslog} %{HOSTNAME:hostname} : %%{FACILITY:facility}-%{SEVERITY:severity}: %{MSGTEXT:msgtext}",
"%{DATESTAMP:time_syslog} %{HOSTNAME:hostname} %{MSGTEXT:msgtext}"
]
}
}

mutate {
gsub => [
"severity", "0", "emergency",
"severity", "1", "alert",
"severity", "2", "critical",
"severity", "3", "error",
"severity", "4", "warning",
"severity", "5", "notice",
"severity", "6", "info",
"severity", "7", "any"
]
}
}

###Enable stdout for debug
output {
if [type] == "syslog" {
stdout {
codec => rubydebug }
}
}
```

```
#output {
# if [type] == "syslog" {
# elasticsearch {
# hosts => ["127.0.0.1:9200"]
# index => "juniperlogs"
# }
# }
#}
```

В первоначальной конфигурации не отправляем распарсенные данные в Elasticsearch, а выводим их на стандартный поток в консоль в формате ruby.

Из доступных фильтров используем grok, позволяющий парсить по regex.

Logstash содержит большое количество патернов для парсинга входящих данных. К сожалению, для распарсивания логов Juniper готовых нет.

Создаем файл сами:

```
sudo nano /usr/share/logstash/vendor/bundle/jruby/2.5.0/gems/logstash-patterns-core-4.1.2/patterns/juniper-log
```

И вносим туда данные индексов и соответсвующих им регулярных выражений:

```
ID \d+
DATESTAMP (\w{3}\s+\d+ \d{2}:\d{2}:\d{2})
HOSTNAME (\w+\d+)
MSGSOURCE \S+
FACILITY \w+
SEVERITY \d
TAG \w+
MSGTEXT (\S+|\s+)+
```

Сохраняем файл.

15) Выполняем конфигурацию маршрутизаторов Juniper для отправки log-сообщений в Logstash.

```
lab@PE3# show system syslog
user * {
any emergency;
}
host 10.0.192.17 {
any any;
port 1514;
source-address 10.0.192.21;
explicit-priority;
}
```

Explicit-priority позволяет добавлять уровни facility и severity в формат обычных лог сообщений.

16) Проверяем корректность созданного конфигурационного файла для Logstash (ключ -t в конце строки).

```
sudo /usr/share/logstash/bin/logstash -f /etc/logstash/conf.d/networklogs.conf -t
```

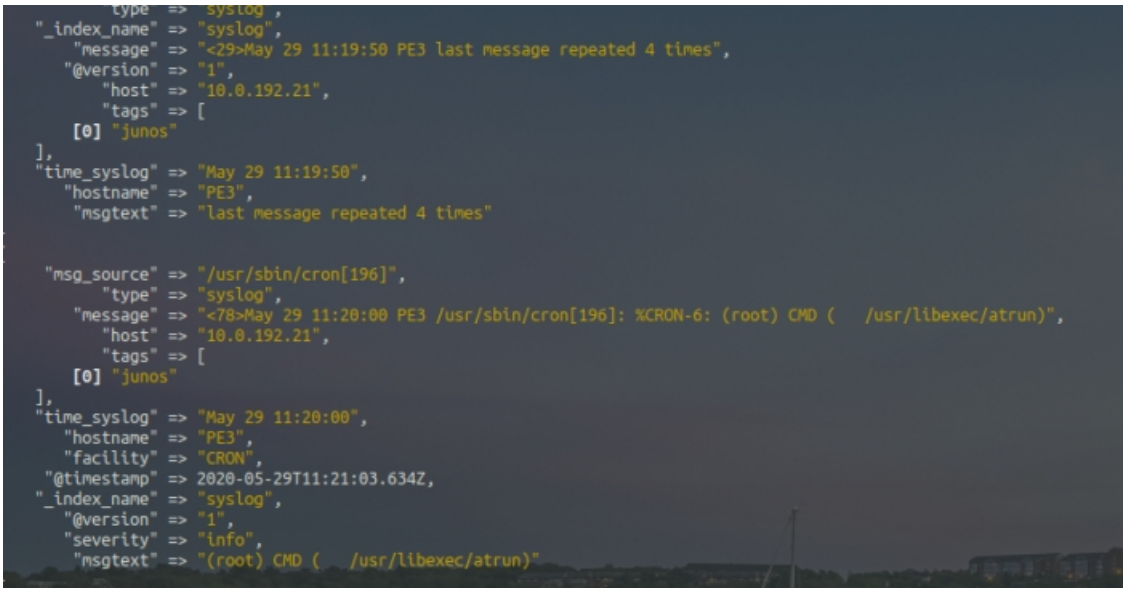
Вывод:

```
lab@lab:~$ sudo systemctl status logstash
● logstash.service - logstash
   Loaded: loaded (/etc/systemd/system/logstash.service; disabled; vendor preset: enabled)
   Active: active (running) since Fri 2020-05-29 10:45:18 UTC; 10s ago
 Main PID: 11848 (java)
    Tasks: 14 (limit: 2317)
   CGroup: /system.slice/logstash.service
           └─11848 /usr/bin/java -Xms1g -Xmx1g -XX:+UseConcMarkSweepGC -XX:CMSInitiatingOccu
```

17) Запускаем logstash с нашим конфигурационным файлом.

```
sudo /usr/share/logstash/bin/logstash -f /etc/logstash/conf.d/networklogs.conf
```

В консоли на stdout отображаются поступающие лог-сообщения с маршрутизатора:



18) Прерываем тест с помощью Ctrl+C.

19) Редактируем конфигурационный файл Logstash для отправки структурированных сообщений в Elasticsearch.

sudo nano /etc/logstash/conf.d/networklogs.conf

В нем необходимо закомментировать блок с stdout, и расскомментировать блок elasticsearch.

Было:

```
###Enable stdout for debug
output {
if [type] == "syslog" {
stdout {
codec => rubydebug }
}
}

#output {
# if [type] == "syslog" {
# elasticsearch {
# hosts => ["127.0.0.1:9200"]
# index => "juniperlogs"
# }
# }
#}
```

Стало:

```
###Enable stdout for debug
#output {
# if [type] == "syslog" {
# stdout {
# codec => rubydebug }
# }
#}

output {
if [type] == "syslog" {
elasticsearch {
hosts => ["127.0.0.1:9200"]
index => "juniperlogs"
}
}
}
```

Сохраняем изменения и закрываем файл.

20) Запускаем Logstash.

sudo systemctl start logstash

21) Проверяем статус.

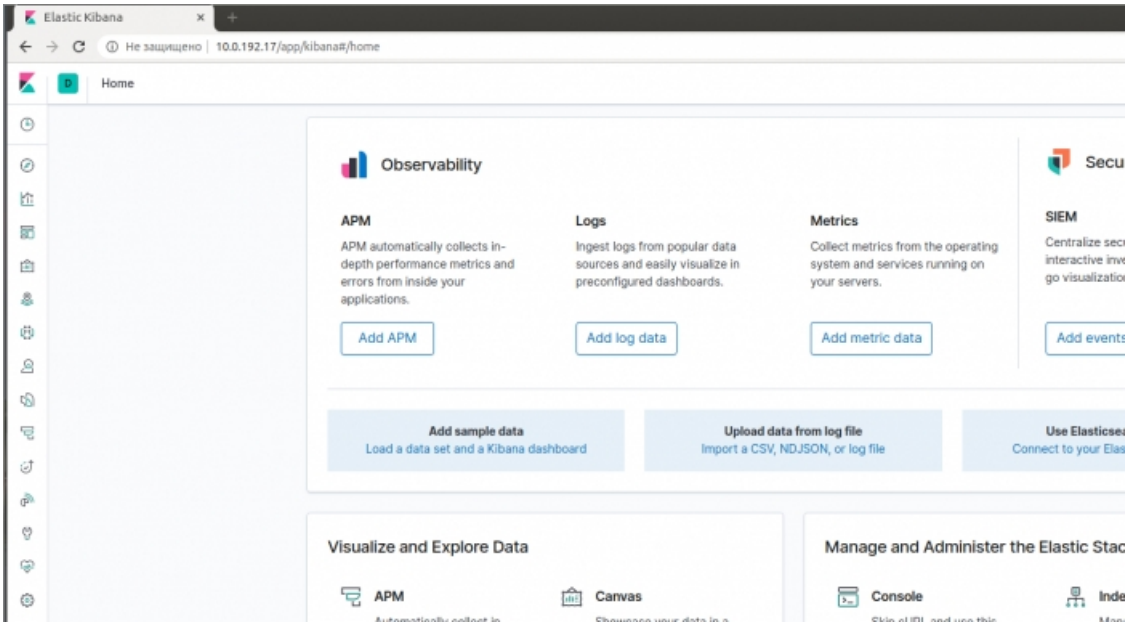
sudo systemctl status logstash

```
lab@lab:~$ sudo /usr/share/logstash/bin/logstash -f /etc/logstash/conf.d/networklogs.conf -t
OpenJDK 64-Bit Server VM warning: Option UseConcMarkSweepGC was deprecated in version 9.0 and will likely be removed in a future release.
WARNING: An illegal reflective access operation has occurred
WARNING: Illegal reflective access by com.headius.backport9.modules.Modules (file:/usr/share/logstash/logstash-core/lib/jars/jruby-complete-9
WARNING: Please consider reporting this to the maintainers of com.headius.backport9.modules.Modules
WARNING: Use --illegal-access=warn to enable warnings of further illegal reflective access operations
WARNING: All illegal access operations will be denied in a future release
WARNING: Could not find logstash.yml which is typically located in $LS_HOME/config or /etc/logstash. You can specify the path using --path.se
Could not find log4j2 configuration at path /usr/share/logstash/config/log4j2.properties. Using default config which logs errors to the conso
[WARN ] 2020-05-29 10:40:07.556 [LogStash::Runner] multilocal - Ignoring the 'pipelines.yml' file because modules or command line options are
[INFO ] 2020-05-29 10:40:13.284 [LogStash::Runner] Reflections - Reflections took 154 ms to scan 1 urls, producing 21 keys and 41 values
Configuration OK
[INFO ] 2020-05-29 10:40:15.135 [LogStash::Runner] runner - Using config.test_and_exit mode. Config Validation Result: OK. Exiting Logstash
```

22) Включаем юнит Logstash в автозагрузку:

sudo systemctl enable logstash

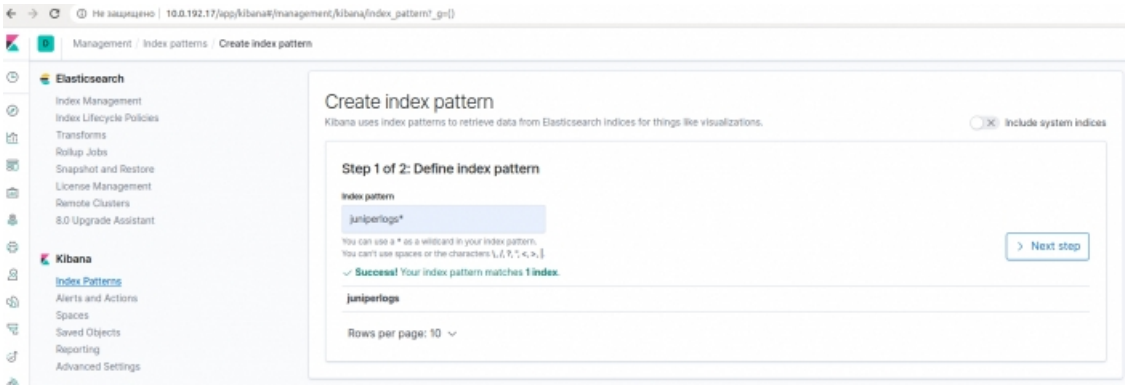
22) Запускаем Kibana в браузере используя ранее созданную в nginx учётную запись.



23) Создаем индексы в Elasticsearch для принимаемых с Logstash структурированных сообщений. Открываем:

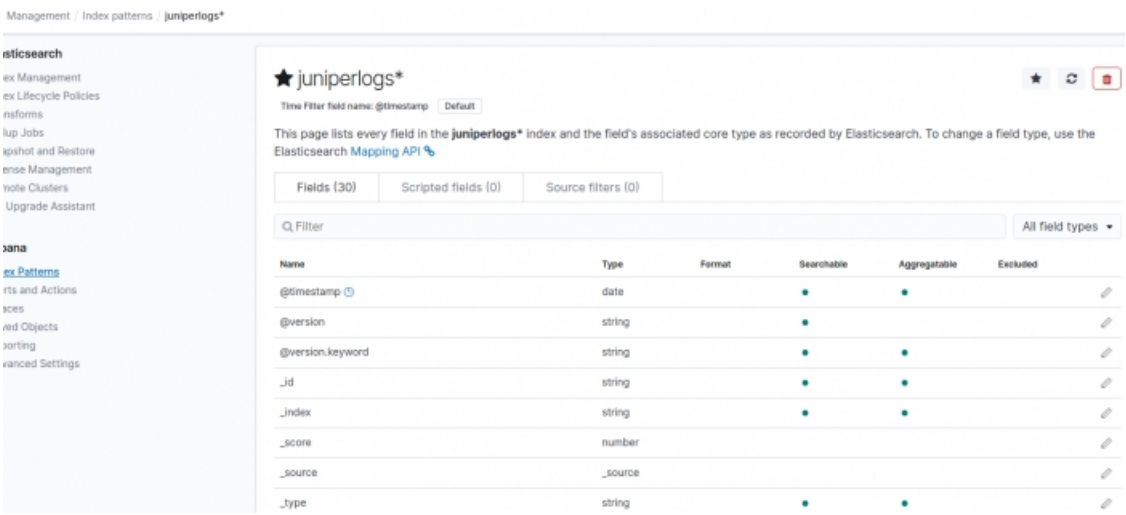
Management/Index patterns/Create index pattern

В поле Index pattern указываем juniperlogs* и нажимаем Next step

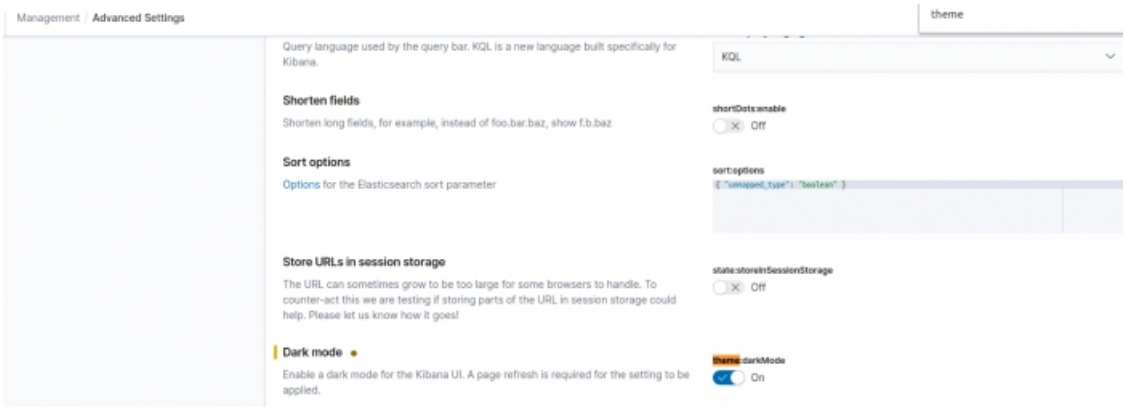


Выбираем на втором шаге в поле Time Filter field name параметр @timestamp и нажимаем кнопку Create index pattern. В результате получаем индексы значений в Elasticsearch.

24)

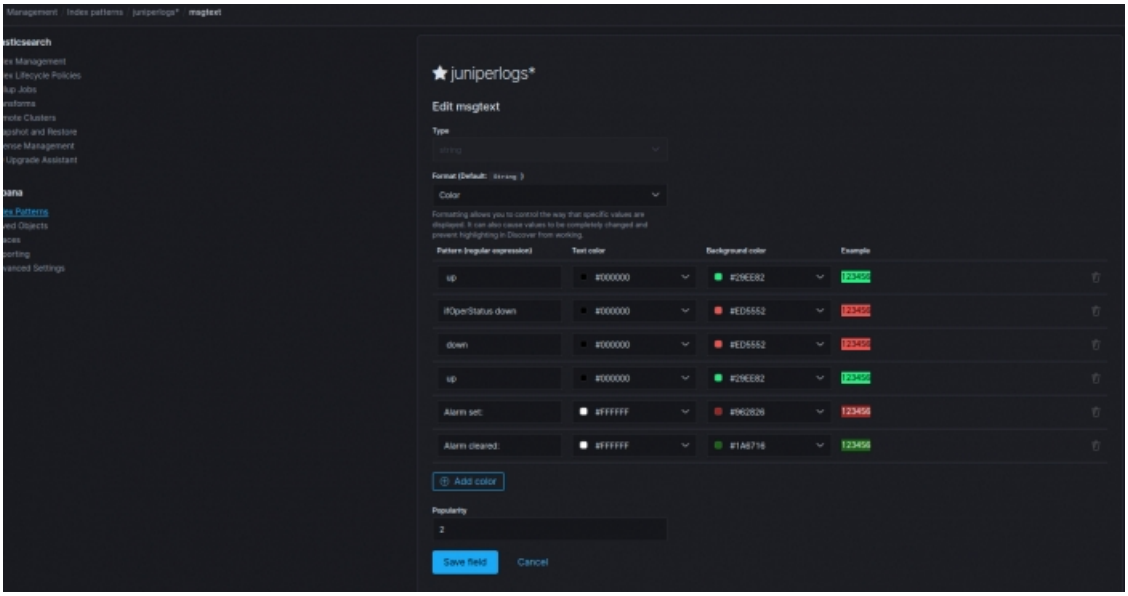


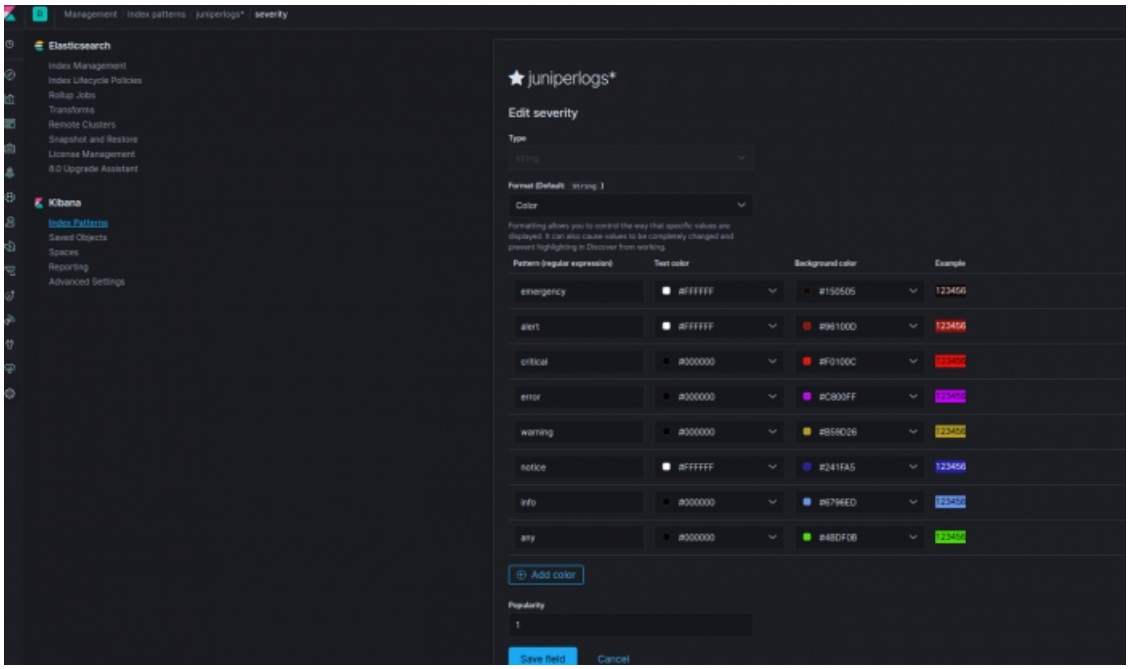
Прежде чем приступить к редактированию созданных индексов, созданию графиков, дашбордов, изменим тему Kibana на темную (Dark mode).



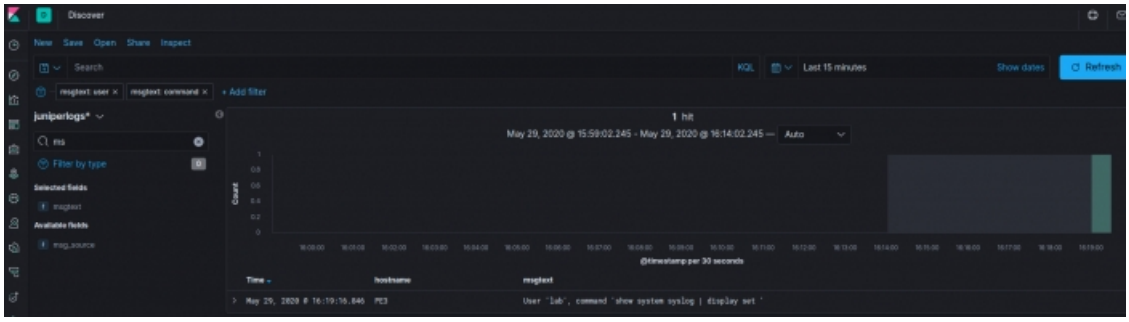
Сохраняем изменения, нажав на кнопку Save changes внизу.

25) Изменяем параметры отображения на дашбордах для некоторых из индексов.





26) Создаем таблицу контроля за действиями сетевых инженеров на сетевом оборудовании. Для этого слева на панели откроем вкладку Discover, добавим необходимые для отображения поля и включим фильтры для индекса msgtext по значениям user и command:

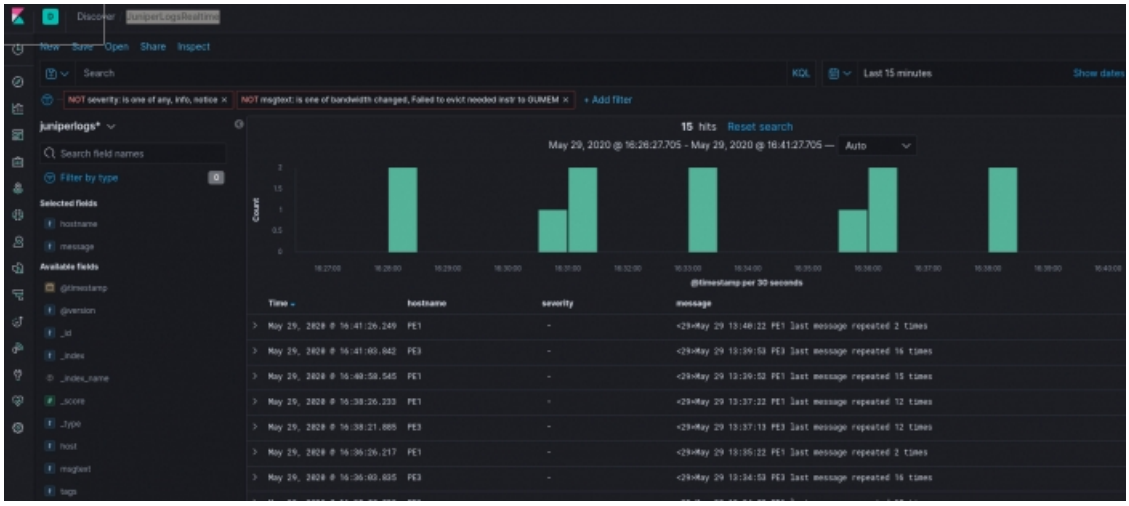


Сохраняем таблицу под именем JuniperUserCommands.

27) Создаем таблицу отображения основных логов на сетевом оборудовании. Откроем снова вкладку Discover и добавим для отображения поля hostname, severity, message.

Создадим фильтры для исключения низкоприоритетных сообщений any, info, notice и сообщений о перерасчете label-switch-path.

Сохраним таблицу под названием JuniperLogsRealtime.



28) Создадим несколько графиков для нашего дашборда. Открываем вкладку Visualize и нажимаем кнопку Create new visualization. Из предложенных вариантов графиков выберем Pie (круговая диаграмма).



Сохраним график с названием JuniperLogsSeverityPie.

И еще несколько графиков гистограмм с различными фильтрами для отображения различных событий.

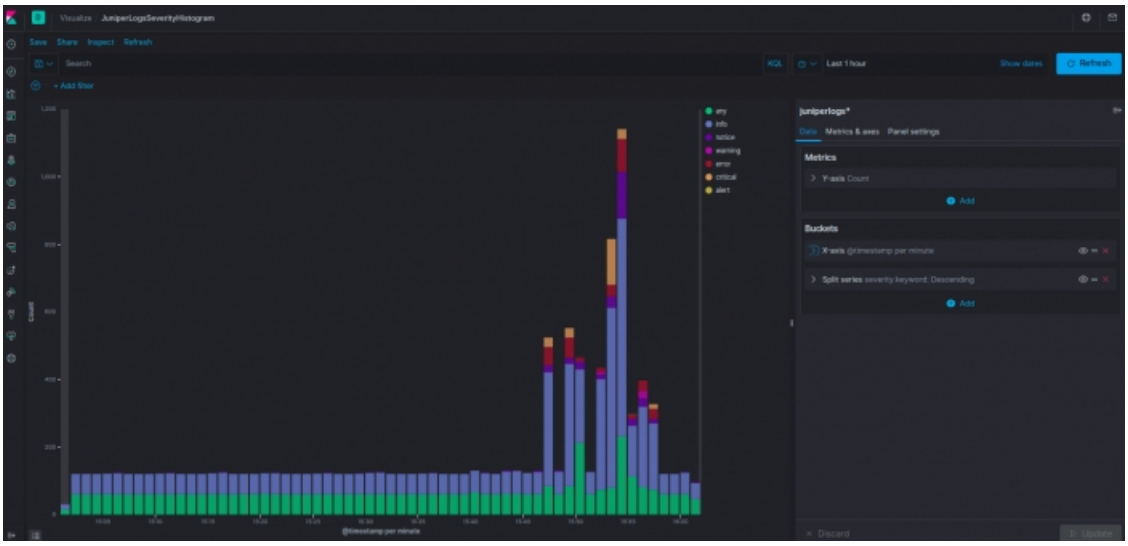


График JuniperLogsSeverityHistogram. Его предназначение следует из названия.

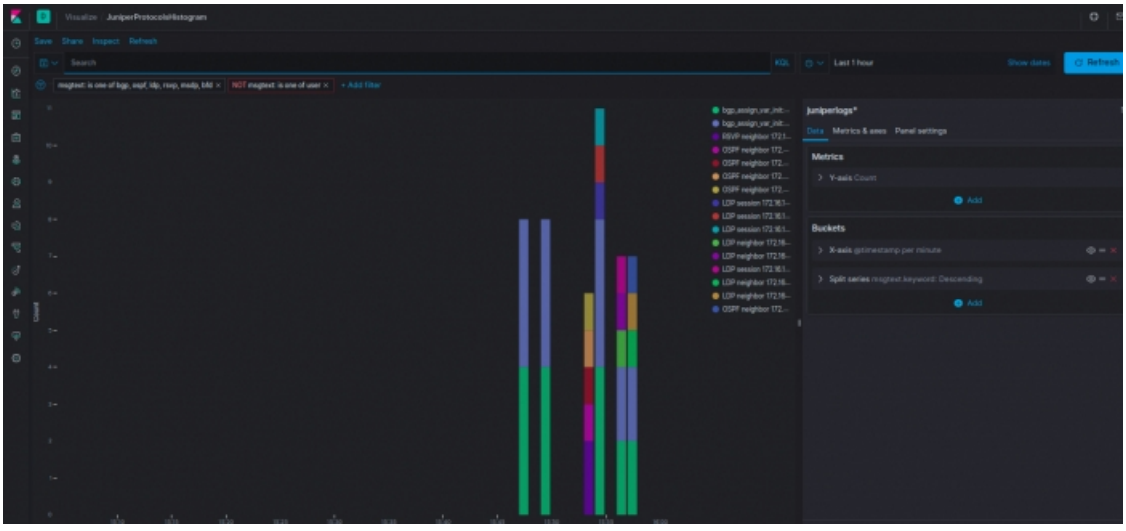


График JuniperProtocolsHistogram предназначен для отображения изменения состояния протоколов.

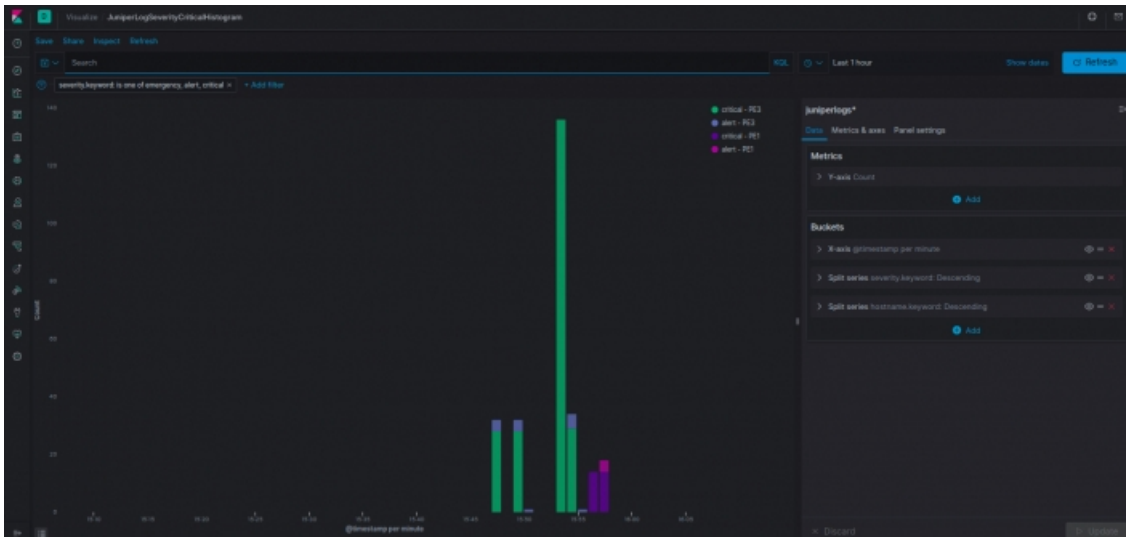


График JuniperLogSeverityCriticalHistogram отображает количество событий степеней Emergency, Critical и Alert.

[Стек ELK - современный инструментарий мониторинга сети]

JuniperLogAlarmHistogram для отображения аварийных сообщений типа Alarm.

[Стек ELK - современный инструментарий мониторинга сети]

JuniperIfdHistogram предназначен для отображения изменений состояния сетевых интерфейсов.

29) Создадим сам дашборд и разместим на нем созданные графики и таблицы. Ниже приведен пример одного из вариантов.

[Стек ELK - современный инструментарий мониторинга сети]

Период отображения можно установить как для дашборда в целом, так и для каждого из графиков в отдельности. Рекомендуемый refresh interval – 2 – 4 секунды. После сохранения скомпанованного дашборда доступно его отображение в режиме Full Screen. Редактирование любого из графиков и таблиц доступно из дашборда. Клик по событию на одном из графиков предоставляет возможность отображения по фильтру данного события на остальных графиках и в таблицах.

Таким образом, open-source вариант ELK позволяет создать соответствующую современным потребностям систему мониторинга сети.