// +----------------------------------------------------------------------------------------------------------------

// | ELK实践【ElasticSearch, LogStash, kibana】

// +----------------------------------------------------------------------------------------------------------------

// | Version: 1.0

// +----------------------------------------------------------------------------------------------------------------

// | Author: JettJia <JettJia@qq.com>

// +----------------------------------------------------------------------------------------------------------------

// | Date: 2017.10.1

// +----------------------------------------------------------------------------------------------------------------

灰色底**黑色加粗**：输入的系统命令

灰色底： 输入的系统命令返回的信息或者配置文件文本信息

黄色底： 技巧或需要注意的注释信息

橙色底： 需要特别注意的地方

蓝色字体： 内容注释

# 1 ELK介绍

## 1.1 elasticsearch

### 1.1.1 elasticsearch介绍

ElasticSearch是一个基于Lucene的搜索服务器。它提供了一个分布式多用户能力的全文搜索引擎，基于RESTful web接口。Elasticsearch是用Java开发的，并作为Apache许可条款下的开放源码发布，是第二流行的企业搜索引擎。设计用于云计算中，能够达到实时搜索，稳定，可靠，快速，安装使用方便。

### 1.1.2 elasticsearch几个重要术语

* NRT   
  elasticsearch是一个近似实时的搜索平台，从索引文档到可搜索有些延迟，通常为1秒。
* 集群   
  集群就是一个或多个节点存储数据，其中一个节点为主节点，这个主节点是可以通过选举产生的，并提供跨节点的联合索引和搜索的功能。集群有一个唯一性标示的名字，默认是elasticsearch，集群名字很重要，每个节点是基于集群名字加入到其集群中的。因此，确保在不同环境中使用不同的集群名字。一个集群可以只有一个节点。强烈建议在配置elasticsearch时，配置成集群模式。
* 节点   
  节点就是一台单一的服务器，是集群的一部分，存储数据并参与集群的索引和搜索功能。像集群一样，节点也是通过名字来标识，默认是在节点启动时随机分配的字符名。当然啦，你可以自己定义。该名字也蛮重要的，在集群中用于识别服务器对应的节点。   
  节点可以通过指定集群名字来加入到集群中。默认情况下，每个节点被设置成加入到elasticsearch集群。如果启动了多个节点，假设能自动发现对方，他们将会自动组建一个名为elasticsearch的集群。
* 索引   
  索引是有几分相似属性的一系列文档的集合。如nginx日志索引、syslog索引等等。索引是由名字标识，名字必须全部小写。这个名字用来进行索引、搜索、更新和删除文档的操作。   
  索引相对于关系型数据库的库。
* 类型   
  在一个索引中，可以定义一个或多个类型。类型是一个逻辑类别还是分区完全取决于你。通常情况下，一个类型被定于成具有一组共同字段的文档。如ttlsa运维生成时间所有的数据存入在一个单一的名为logstash-ttlsa的索引中，同时，定义了用户数据类型，帖子数据类型和评论类型。   
  类型相对于关系型数据库的表。
* 文档   
  文档是信息的基本单元，可以被索引的。文档是以JSON格式表现的。   
  在类型中，可以根据需求存储多个文档。   
  虽然一个文档在物理上位于一个索引，实际上一个文档必须在一个索引内被索引和分配一个类型。   
  文档相对于关系型数据库的列。
* 分片和副本   
  在实际情况下，索引存储的数据可能超过单个节点的硬件限制。如一个十亿文档需1TB空间可能不适合存储在单个节点的磁盘上，或者从单个节点搜索请求太慢了。为了解决这个问题，elasticsearch提供将索引分成多个分片的功能。当在创建索引时，可以定义想要分片的数量。每一个分片就是一个全功能的独立的索引，可以位于集群中任何节点上。   
  分片的两个最主要原因：   
  a、水平分割扩展，增大存储量   
  b、分布式并行跨分片操作，提高性能和吞吐量   
  分布式分片的机制和搜索请求的文档如何汇总完全是有elasticsearch控制的，这些对用户而言是透明的。   
  网络问题等等其它问题可以在任何时候不期而至，为了健壮性，强烈建议要有一个故障切换机制，无论何种故障以防止分片或者节点不可用。   
  为此，elasticsearch让我们将索引分片复制一份或多份，称之为分片副本或副本。   
  副本也有两个最主要原因：   
  高可用性，以应对分片或者节点故障。出于这个原因，分片副本要在不同的节点上。   
  提供性能，增大吞吐量，搜索可以并行在所有副本上执行。   
  总之，每一个索引可以被分成多个分片。索引也可以有0个或多个副本。复制后，每个索引都有主分片(母分片)和复制分片(复制于母分片)。分片和副本数量可以在每个索引被创建时定义。索引创建后，可以在任何时候动态的更改副本数量，但是，不能改变分片数。   
  默认情况下，elasticsearch为每个索引分片5个主分片和1个副本，这就意味着集群至少需要2个节点。索引将会有5个主分片和5个副本(1个完整副本)，每个索引总共有10个分片。   
  每个elasticsearch分片是一个Lucene索引。一个单个Lucene索引有最大的文档数LUCENE-5843, 文档数限制为2147483519(MAX\_VALUE – 128)。 可通过\_cat/shards来监控分片大小。

## 1.2 logstash

### 1.2.1 logstash 介绍

LogStash由JRuby语言编写，基于消息（message-based）的简单架构，并运行在Java虚拟机（JVM）上。不同于分离的代理端（agent）或主机端（server），LogStash可配置单一的代理端（agent）与其它开源软件结合，以实现不同的功能。

### 1.2.2 logStash的四大组件

* Shipper：发送事件（events）至LogStash；通常，远程代理端（agent）只需要运行这个组件即可；
* Broker and Indexer：接收并索引化事件；
* Search and Storage：允许对事件进行搜索和存储；
* Web Interface：基于Web的展示界面   
  正是由于以上组件在LogStash架构中可独立部署，才提供了更好的集群扩展性。

### 1.2.2 LogStash主机分类

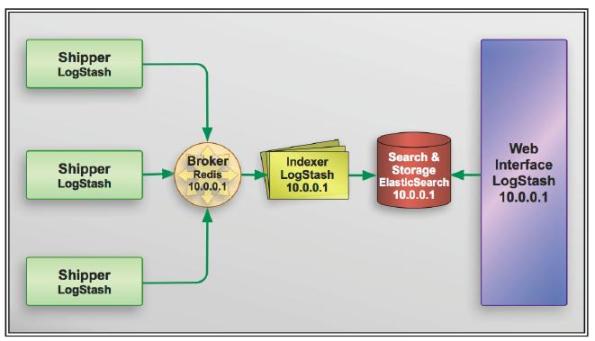
* 代理主机（agent host）：作为事件的传递者（shipper），将各种日志数据发送至中心主机；只需运行Logstash 代理（agent）程序；
* 中心主机（central host）：可运行包括中间转发器（Broker）、索引器（Indexer）、搜索和存储器（Search and Storage）、Web界面端（Web Interface）在内的各个组件，以实现对日志数据的接收、处理和存储。

## 1.3 kibana

Logstash是一个完全开源的工具，他可以对你的日志进行收集、分析，并将其存储供以后使用（如，搜索），您可以使用它。说到搜索，logstash带有一个web界面，搜索和展示所有日志。

# 2 使用ELK必要性（解决运维痛点）

* 开发人员不能登录线上服务器查看详细日志
* 各个系统都有日志，日至数据分散难以查找
* 日志数据量大，查询速度慢，或者数据不够实时

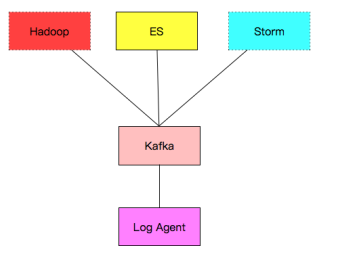


**架构升级**

logstash写入kafka

然后es从kafka读取

hadoop从kafka读取



参考文档: es+kafka+logstash+kibana

# 3 开发环境准备

## 3.1 服务器介绍

|  |  |
| --- | --- |
| IP | 名称 |
| 10.0.0.11 | linux-node1 |
| 10.0.0.22 | linux-node2 |

## 3.2 系统环境两台机器一致

### 3.2.1 系统环境

[root@linux-node1 ~]# cat /etc/redhat-release

CentOS release 6.7 (Final)

[root@linux-node1 ~]# uname -a

Linux linux-node1 2.6.32-573.el6.x86\_64 #1 SMP Thu Jul 23 15:44:03 UTC 2015 x86\_64 x86\_64 x86\_64 GNU/Linux

[root@linux-node1 ~]# cat /etc/hosts

127.0.0.1 localhost localhost.localdomain localhost4 localhost4.localdomain4

::1 localhost localhost.localdomain localhost6 localhost6.localdomain6

10.0.0.11 linux-node1

10.0.0.22 linux-node2

### 3.2.2 JDK安装

两个节点都安装

[root@linux-node1 ~]# yum install -y java

[root@linux-node1 ~]# java -version

openjdk version "1.8.0\_171"

OpenJDK Runtime Environment (build 1.8.0\_171-b10)

OpenJDK 64-Bit Server VM (build 25.171-b10, mixed mode)

## 3.3 ELK安装(两个节点完全一致)

### 3.3.1 elasticsearch安装

1）下载并安装GPG key

[root@linux-node1 ~]# rpm --import <https://packages.elastic.co/GPG-KEY-elasticsearch>

[root@linux-node1 ~]# vim /etc/yum.repos.d/elasticsearch.repo

[elasticsearch-2.x]

name=Elasticsearch repository for 2.x packages

baseurl=http://packages.elastic.co/elasticsearch/2.x/centos

gpgcheck=1

gpgkey=http://packages.elastic.co/GPG-KEY-elasticsearch

enabled=1

1. 安装elasticsearch

[root@linux-node1 ~]# yum install -y elasticsearch

### 3.3.2 logstash安装

1）下载并安装GPG key

[root@linux-node1 ~]# rpm --import <https://packages.elastic.co/GPG-KEY-elasticsearch>

[root@linux-node1 ~]# vim /etc/yum.repos.d/logstash.repo

[logstash-2.1]

name=Logstash repository for 2.1.x packages

baseurl=http://packages.elastic.co/logstash/2.1/centos

gpgcheck=1

gpgkey=http://packages.elastic.co/GPG-KEY-elasticsearch

enabled=1

2）安装logstash

[root@linux-node1 ~]# yum install -y logstash

### 3.3.3 kibana安装

[root@linux-node1 ~]# cd /usr/local/src

[root@linux-node1 src]#wget <https://download.elastic.co/kibana/kibana/kibana-4.3.1-linux-x64.tar.gz>

[root@linux-node1 src]# tar zxf kibana-4.3.1-linux-x64.tar.gz

[root@linux-node1 src]# mv kibana-4.3.1-linux-x64 /usr/local/

[root@linux-node1 src]# ln -s /usr/local/kibana-4.3.1-linux-x64/ /usr/local/kibana

# 4 管理配置elasticsearch

## 4.1 管理linux-node1节点的elasticsearch

1）修改elasticsearch配置

[root@linux-node1 ~]# vim /etc/elasticsearch/elasticsearch.yml

[root@linux-node1 ~]# grep -n '^[a-Z]' /etc/elasticsearch/elasticsearch.yml

17:cluster.name: chuck-cluster #判别节点是否是统一集群

23:node.name: linux-node1 #节点的hostname

33:path.data: /data/es-data #数据存放路径

37:path.logs: /var/log/elasticsearch/ #日志路径

43:bootstrap.memory\_lock: true #锁住内存，使内存不会再swap中使用

54:network.host: 0.0.0.0 #允许访问的ip

58:http.port: 9200 #端口

2）创建数据存放目录，并授权

[root@linux-node1 ~]# mkdir -p /data/es-data

[root@linux-node1 ~]# chown elasticsearch.elasticsearch /data/es-data/

1. 启动elasticsearch

[root@linux-node1 ~]# /etc/init.d/elasticsearch start

正在启动 elasticsearch： [确定]

[root@linux-node1 ~]# service elasticsearch status

elasticsearch (pid 27028) 正在运行...

[root@linux-node1 ~]# netstat -lntup|grep 9200

tcp 0 0 :::9200 :::\* LISTEN 27028/java

1. 访问测试

[root@linux-node1 ~]# curl -i 10.0.0.11:9200

HTTP/1.1 200 OK

Content-Type: application/json; charset=UTF-8

Content-Length: 365

{

"name" : "linux-node1",

"cluster\_name" : "chuck-cluster",

"cluster\_uuid" : "NWgUz94YTdeGNv7AiZHfmg",

"version" : {

"number" : "2.4.6",

"build\_hash" : "5376dca9f70f3abef96a77f4bb22720ace8240fd",

"build\_timestamp" : "2017-07-18T12:17:44Z",

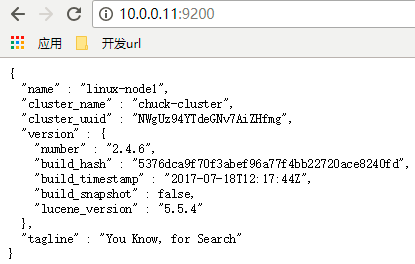
"build\_snapshot" : false,

"lucene\_version" : "5.5.4"

},

"tagline" : "You Know, for Search"

}



## 4.2 elasticsearch交互

### 4.2.1 交互的两种方法

* Java API

node client

Transport client

* RESTful API

Javascript

.NET

php

Perl

Python

Ruby

### 4.2.2 RESTful API方法

[root@linux-node1 ~]# curl -i -XGET '10.0.0.11:9200/\_count?pretty'

HTTP/1.1 200 OK

Content-Type: application/json; charset=UTF-8

Content-Length: 95

{

"count" : 0,

"\_shards" : {

"total" : 0,

"successful" : 0,

"failed" : 0

}

}

## 4.3 插件

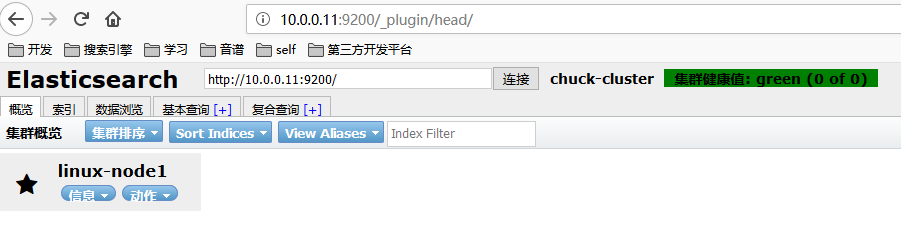
### 4.3.1 head插件显示索引和分片情况

1安装插件

[root@linux-node1 ~]# /usr/share/elasticsearch/bin/plugin install mobz/elasticsearch-head

2访问插件

<http://10.0.0.11:9200/_plugin/head/>

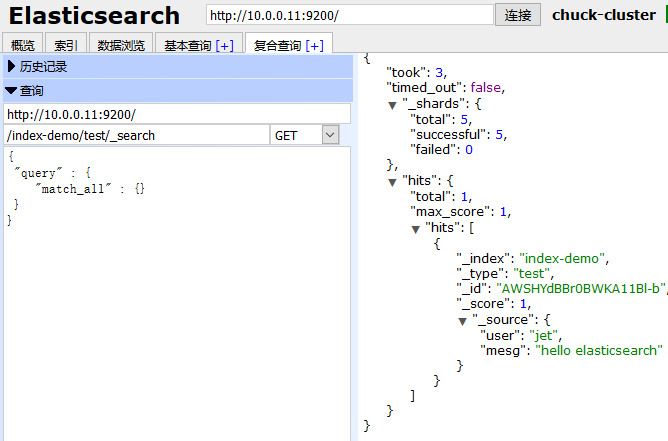


3测试

1. 添加数据



1. 查找数据



### 4.3.2 kopf插件监控elasticsearch

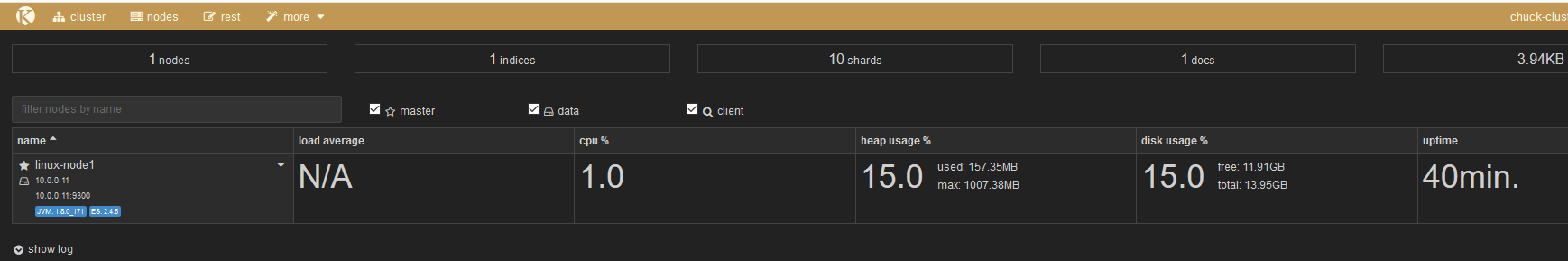
1安装

[root@linux-node1 ~]# /usr/share/elasticsearch/bin/plugin install lmenezes/elasticsearch-kopf

2 访问

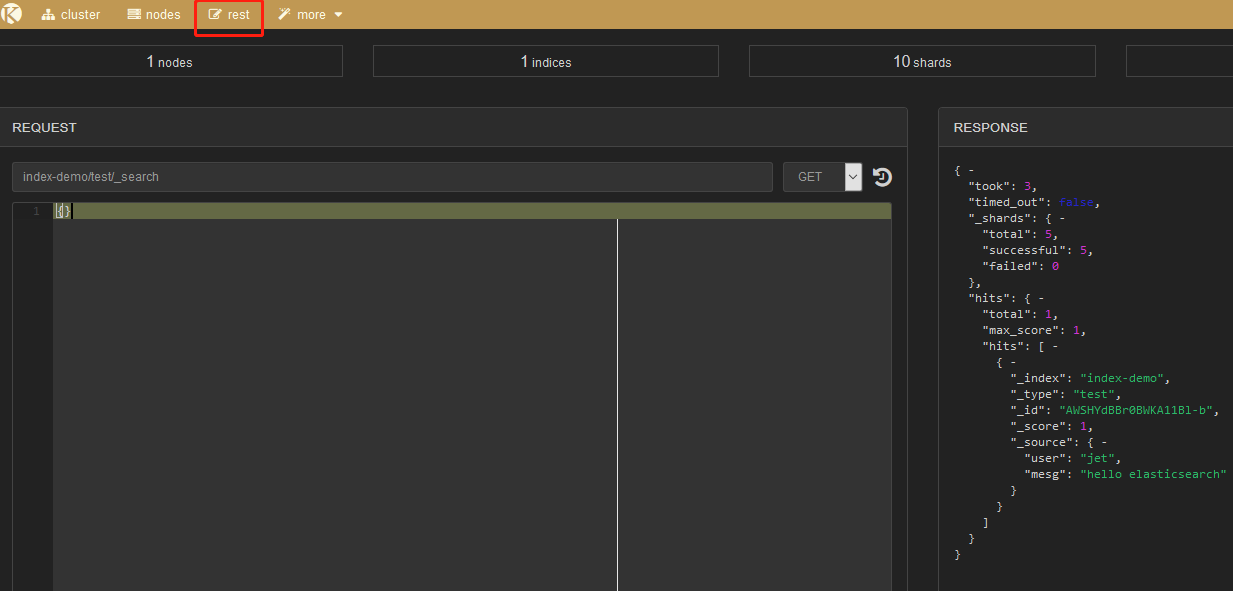
<http://10.0.0.11:9200/_plugin/kopf>

3 简单介绍



从图中可以看出节点的负载，cpu适应情况，java对内存的使用（heap usage），磁盘使用，启动时间

下面 reset模块提供了REST API



## 4.4 管理linux-node2节点的elasticsearch

1）修改elasticsearch配置

[root@linux-node2 ~]# vim /etc/elasticsearch/elasticsearch.yml

[root@linux-node2 ~]# grep -n '^[a-Z]' /etc/elasticsearch/elasticsearch.yml

17:cluster.name: chuck-cluster #判别节点是否是统一集群

23:node.name: linux-node1 #节点的hostname

33:path.data: /data/es-data #数据存放路径

37:path.logs: /var/log/elasticsearch/ #日志路径

43:bootstrap.memory\_lock: true #锁住内存，使内存不会再swap中使用

54:network.host: 0.0.0.0 #允许访问的ip

58:http.port: 9200 #端口

2）创建数据存放目录，并授权

[root@linux-node2 ~]# mkdir -p /data/es-data

[root@linux-node2 ~]# chown elasticsearch.elasticsearch /data/es-data/

1. 启动elasticsearch

[root@linux-node2 ~]# /etc/init.d/elasticsearch start

正在启动 elasticsearch： [确定]

[root@linux-node2 ~]# netstat -lntup|grep 9200

tcp 0 0 :::9200 :::\* LISTEN 27028/java

1. 访问测试

[root@linux-node1 ~]# curl -i 10.0.0.22:9200

7）查看 head，看node2节点是否加入集群

<http://10.0.0.11:9200/_plugin/head/>

没有加入，修改node2配置 修改单播模式

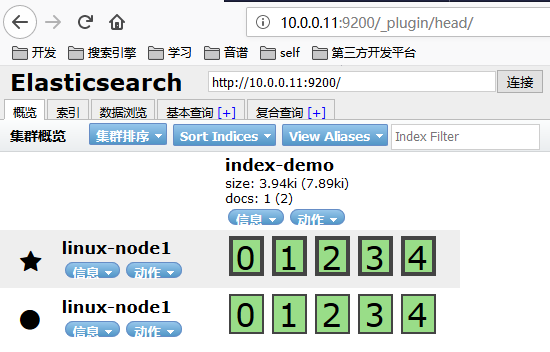
discovery.zen.ping.unicast.hosts: ["10.0.0.11", "10.0.0.22"]

说明：虚拟机环境下，网络设置的nat模式，可能会有影响。所以把node2节点设置成了单播的模式

重启：

[root@linux-node2 src]# /etc/init.d/elasticsearch restart

查看是否集群设置好



# 5 管理配置logstash

## 5.1 终端演示

演示1：启动一个logstash,-e：在命令行执行；input输入，stdin标准输入，是一个插件；output输出，stdout：标准输出

[root@linux-node1 ~]# /opt/logstash/bin/logstash -e 'input { stdin{} } output { stdout{} }'

Settings: Default filter workers: 1

Logstash startup completed

hello

2018-07-11T05:50:48.267Z linux-node1 hello

world

2018-07-11T05:50:52.940Z linux-node1 world

演示2：使用rubydebug显示详细输出，codec为一种编解码器

[root@linux-node1 ~]# /opt/logstash/bin/logstash -e 'input { stdin{} } output { stdout{ codec => rubydebug} }'

Settings: Default filter workers: 1

Logstash startup completed

hello

{

"message" => "hello",

"@version" => "1",

"@timestamp" => "2018-07-11T05:52:43.898Z",

"host" => "linux-node1"

}

演示3：使用logstash将信息写入到elasticsearch

[root@linux-node1 ~]# /opt/logstash/bin/logstash -e 'input { stdin{} } output { elasticsearch { hosts => ["10.0.0.11:9200"] } }'

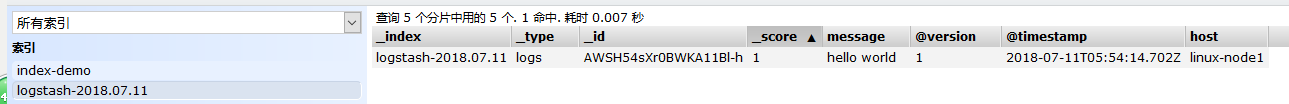
Settings: Default filter workers: 1

Logstash startup completed

hello world

在head插件中查看





## 5.2 学习input和output插件编写

官方地址：<https://www.elastic.co/guide/en/logstash/2.4/index.html>

示例如下：

input {

file {

path => "/var/log/messages"

type => "syslog"

}

file {

path => "/var/log/apache/access.log"

type => "apache"

}

}

filter {

}

output {

elasticsearch {

hosts => ["192.168.56.11:9200"]

index => "system-%{+YYYY.MM.dd}"

}

}

## 5.3 编写conf文件实践

### 5.3.1 收集系统日志messages

1）编辑收集日志的conf文件

[root@linux-node1 ~]# vim /etc/logstash/conf.d/system.conf

input {

file {

path => "/var/log/messages"

type => "system"

start\_position => "beginning"

}

}

filter {

}

output {

elasticsearch {

hosts => ["10.0.0.11:9200"]

index => "system-%{+YYYY.MM.dd}"

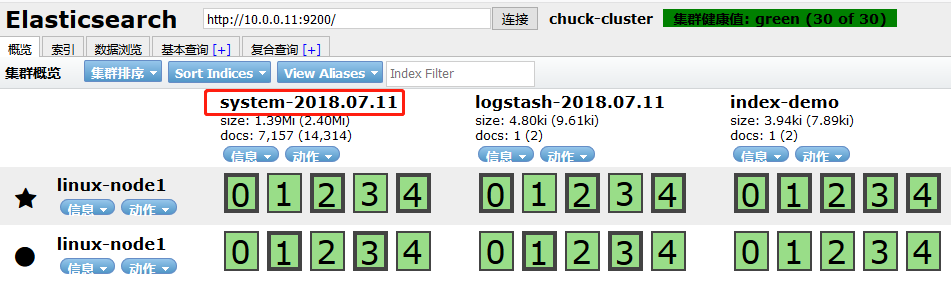
}

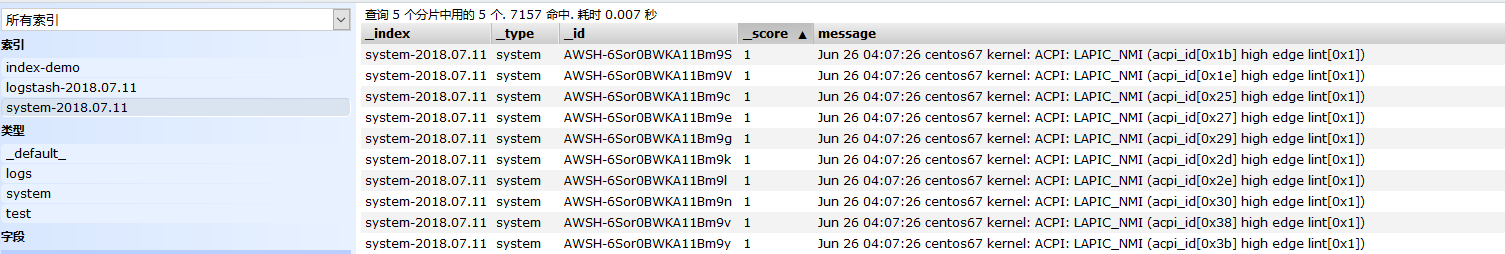
}

2）指定conf文件收集

[root@linux-node1 ~]# /opt/logstash/bin/logstash -f /etc/logstash/conf.d/system.conf

1. 浏览器查看验证





### 5.3.2 收集elasticsearch的error日志

此处把上个system日志和这个error（java程序日志）日志，放在一起。使用if判断，两种日志分别写到不同索引中.此处的type（固定的就是type，不可更改）不可以和日志格式的任何一个域（可以理解为字段）的名称重复，也就是说日志的域不可以有type这个名称。

1）修改配置文件

[root@linux-node1 ~]# mv /etc/logstash/conf.d/system.conf /etc/logstash/conf.d/all.conf

[root@linux-node1 ~]# cat /etc/logstash/conf.d/all.conf

input {

file {

path => "/var/log/messages"

type => "system"

start\_position => "beginning"

}

file {

path => "/var/log/elasticsearch/chuck-cluster.log"

type => "elasticsearch-error-log"

start\_position => "beginning"

}

}

filter {

}

output {

if [type] == "system" {

elasticsearch {

hosts => ["10.0.0.11:9200"]

index => "system-%{+YYYY.MM.dd}"

}

}

if [type] == "elasticsearch-error-log" {

elasticsearch {

hosts => ["10.0.0.11:9200"]

index => "elasticsearch-error-log-%{+YYYY.MM.dd}"

}

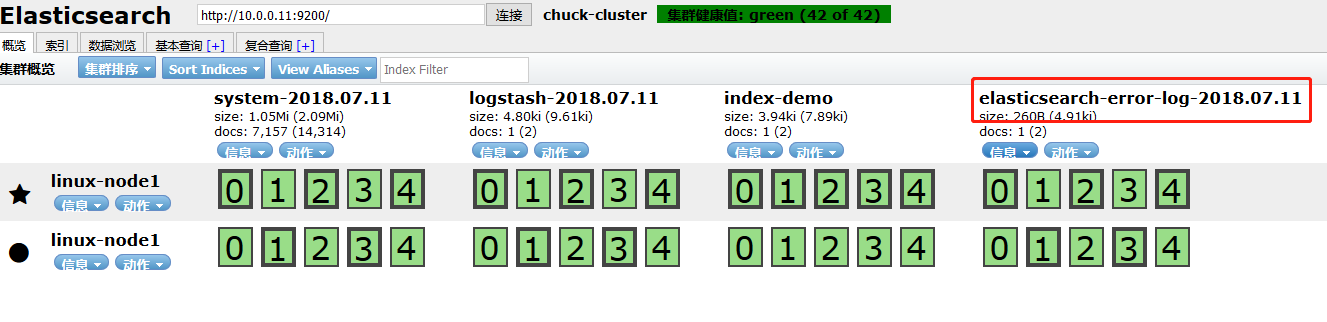
}

}

2）指定conf文件收集

[root@linux-node1 ~]# /opt/logstash/bin/logstash -f /etc/logstash/conf.d/all.conf

1. 浏览器查看验证



说明：上面不完善，因为日志文件拆成单行的了，实际上java日志是成段落的。优化如下

### 5.3.3 完善版es日志收集

1）了解手册

引入codec的multiline插件

官方文档：https://www.elastic.co/guide/en/logstash/2.4/plugins-codecs-multiline.html

官方文档模块：

input {

stdin {

codec => multiline {

pattern => "pattern, a regexp"

negate => "true" or "false"

what => "previous" or "next"`

}

}

}

regrxp：使用正则，什么情况下把多行合并起来

negate:正向匹配和反向匹配

what:合并到当前行还是下一行

在标准输入和标准输出中测试以证明多行收集到一个日志成功

1. 编写测试的conf，在终端验证

[root@linux-node1 ~]# cat muliline.conf

input {

stdin {

codec => multiline{

pattern => "^\["

negate => true

what => "previous"

}

}

}

output {

stdout {

codec => "rubydebug"

}

}

# 验证

[root@linux-node1 ~]# /opt/logstash/bin/logstash -f muliline.conf

Settings: Default filter workers: 1

Logstash startup completed

[22

[22

{

"@timestamp" => "2018-07-11T08:51:37.497Z",

"message" => "[22",

"@version" => "1",

"host" => "linux-node1"

}

wef

wew

[2223

{

"@timestamp" => "2018-07-11T08:51:50.349Z",

"message" => "[22\nwef\nwew",

"@version" => "1",

"tags" => [

[0] "multiline"

],

"host" => "linux-node1"

}

1. 完整的配置如下

[root@linux-node1 ~]# cat /etc/logstash/conf.d/all.conf

input {

file {

path => "/var/log/messages"

type => "system"

start\_position => "beginning"

}

file {

path => "/var/log/elasticsearch/chuck-cluster.log"

type => "elasticsearch-error-log"

start\_position => "beginning"

codec => multiline {

pattern => "^\["

negate => true

what => "previous"

}

}

}

filter {

}

output {

if [type] == "system" {

elasticsearch {

hosts => ["10.0.0.11:9200"]

index => "system-%{+YYYY.MM.dd}"

}

}

if [type] == "elasticsearch-error-log" {

elasticsearch {

hosts => ["10.0.0.11:9200"]

index => "elasticsearch-error-log-%{+YYYY.MM.dd}"

}

}

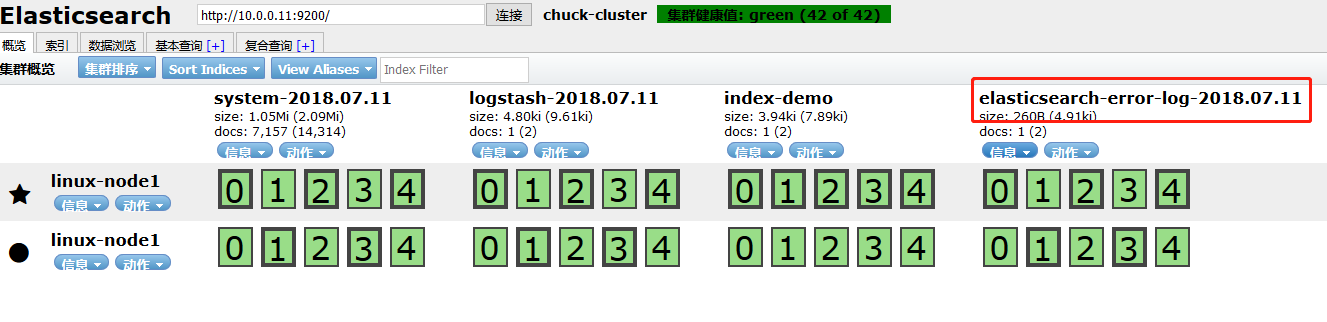
}

1. 删除之前测试的收集的信息，让其重新收集

[root@linux-node1 ~]# rm -rf .sincedb\_452905a167cf4509fd08acb964fdb20c

[root@linux-node1 ~]# /opt/logstash/bin/logstash -f /etc/logstash/conf.d/all.conf

1. 浏览器验证



# 6 管理配置kibana

## 6.1 启动kibana

1 编辑kibana配置文件

[root@linux-node1 ~]# grep '^[a-Z]' /usr/local/kibana/config/kibana.yml

server.port: 5601

server.host: "0.0.0.0"

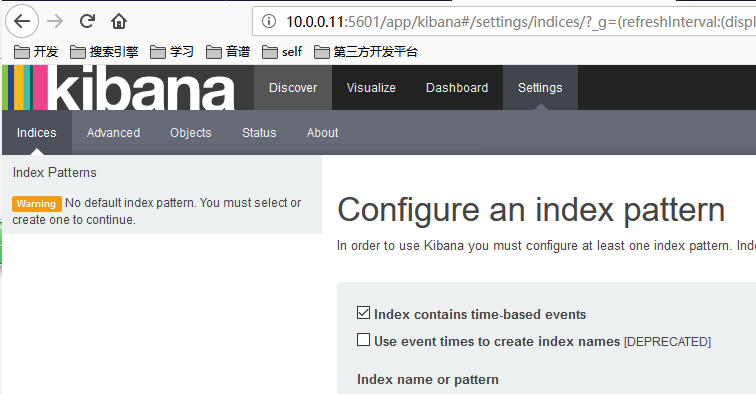
elasticsearch.url: "http://10.0.0.11:9200"

kibana.index: ".kibana"

2 启动kibana

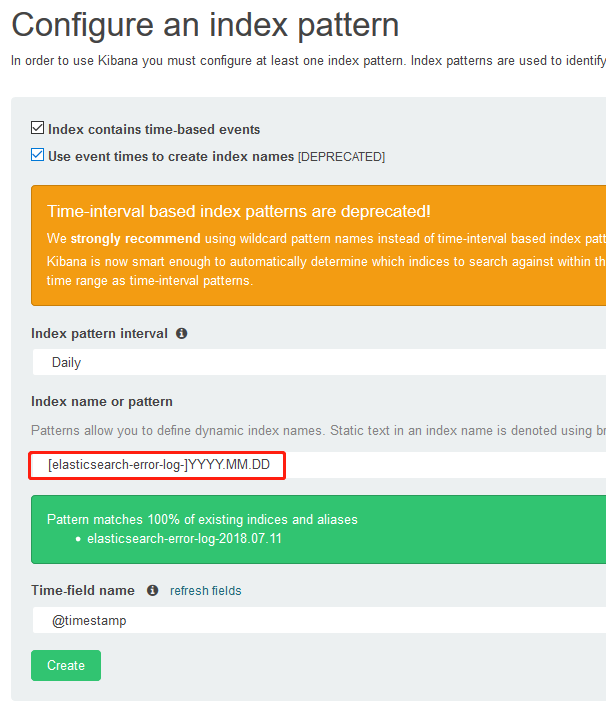
[root@linux-node1 ~]# /usr/local/kibana/bin/kibana

3 浏览器查看



## 6.2 管理kibana

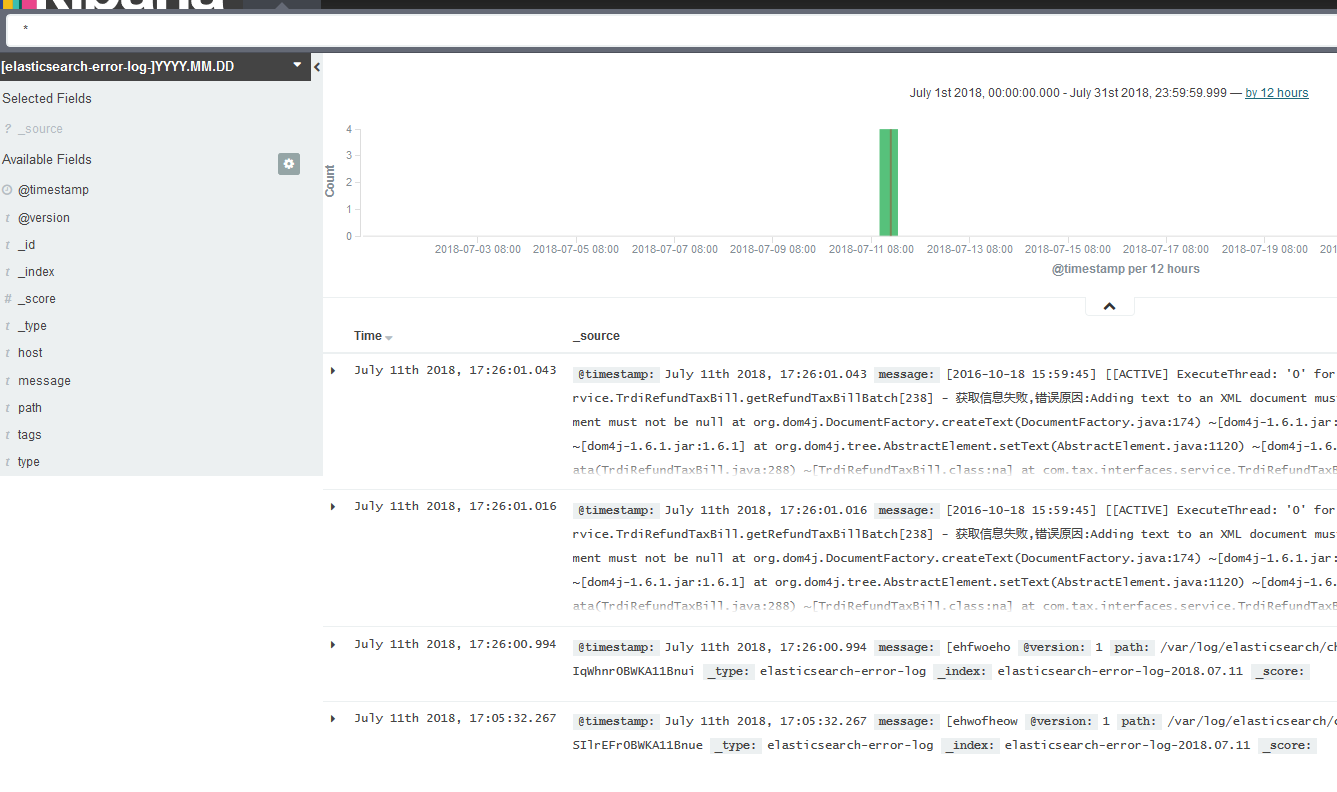
### 6.2.1添加一个已经存在的es的index



### 6.2.2 查看该index信息

设置查询的时间：在右上角

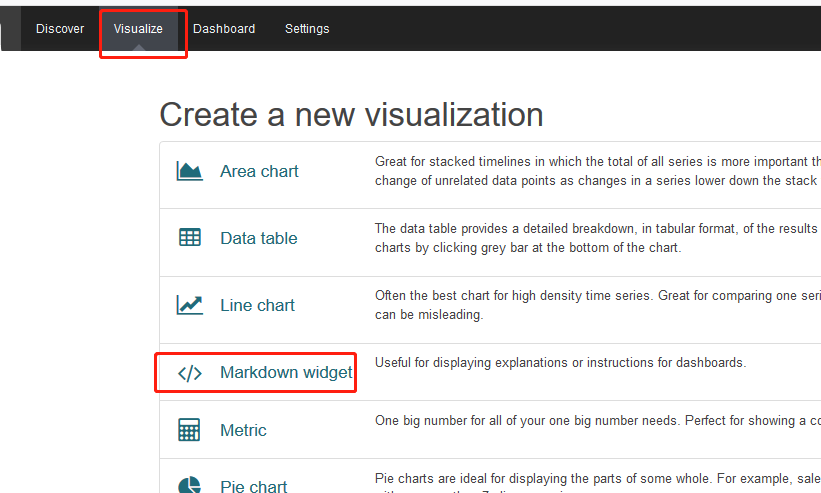
选择discover，已经成功验证了muliline插件

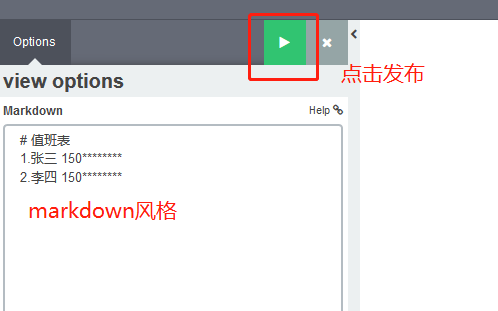


## 6.3 kibana使用

### 1 markdown编写值班表

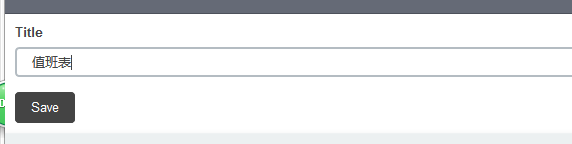
编写这个kibana的介绍信息,例如这里写值班的信息



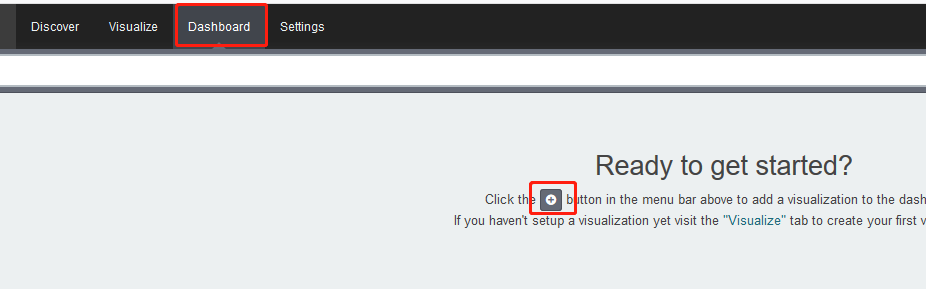


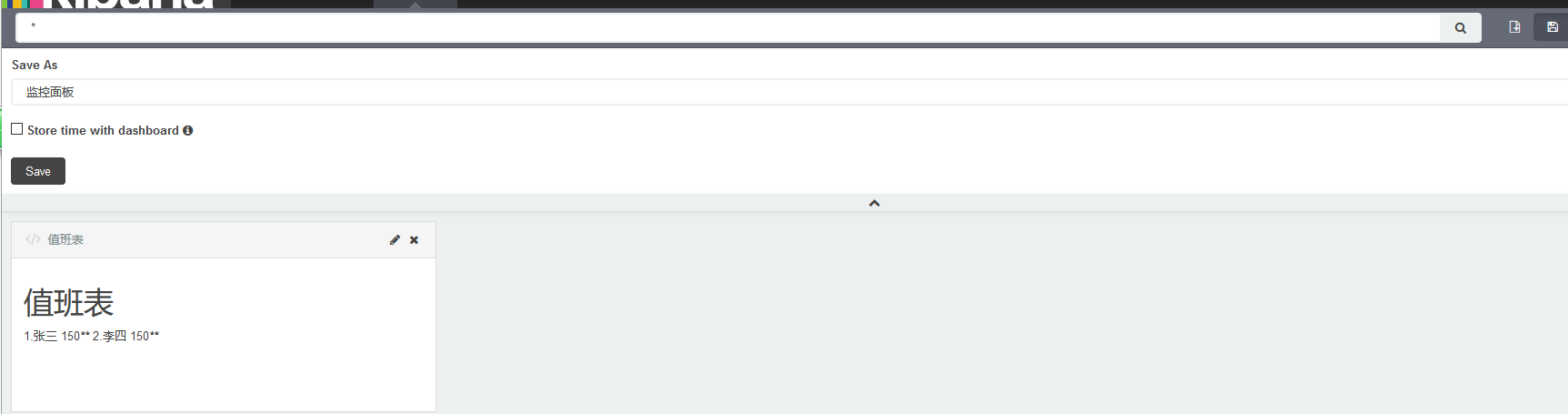
点击保存



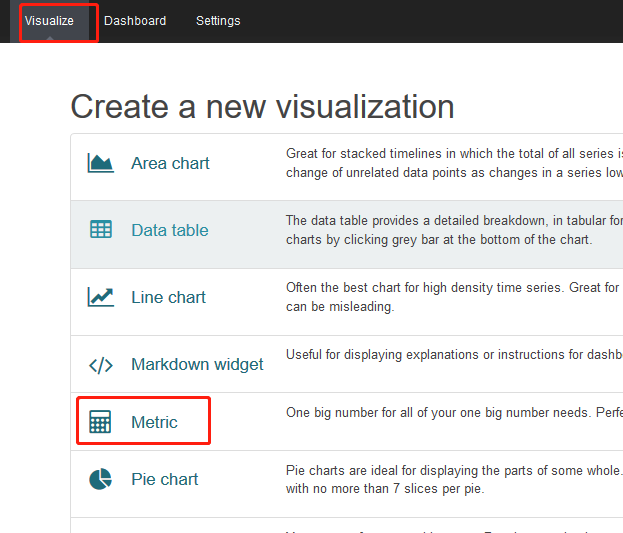


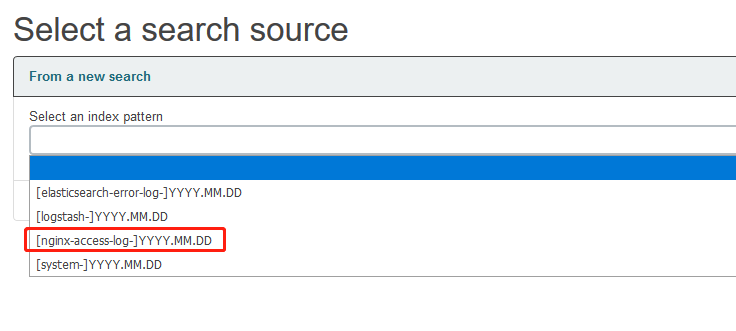
### 2 dashboard 添加上面的值班表

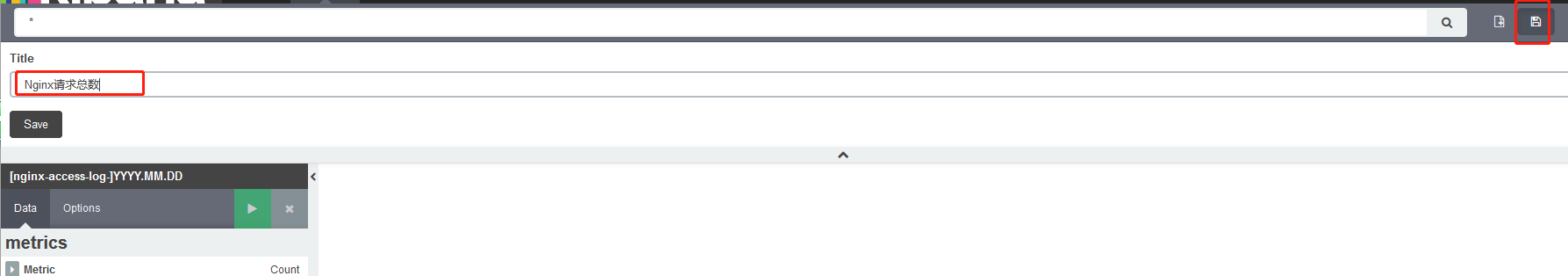




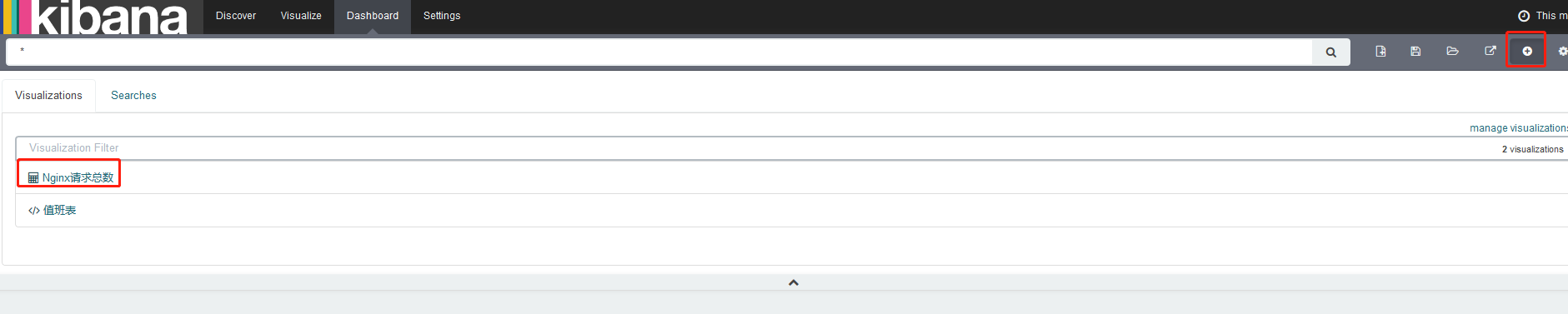
### 3 metric 添加nginx的请求总数



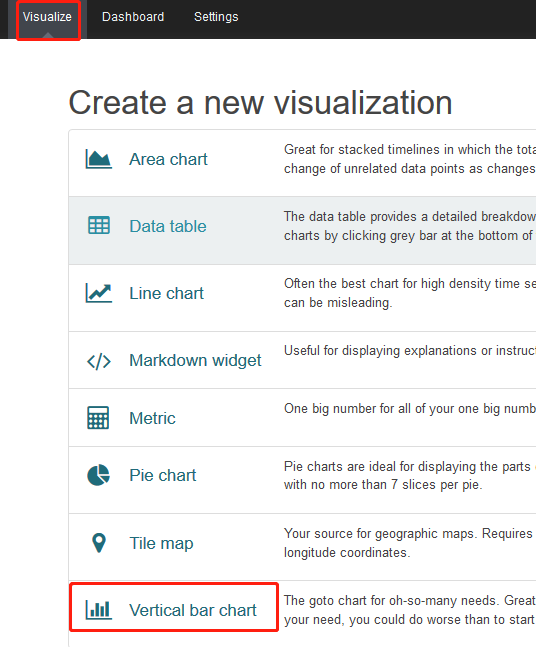


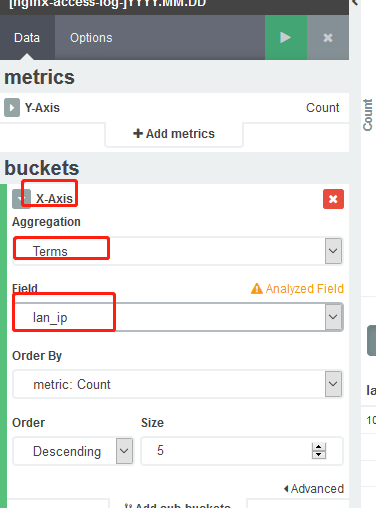


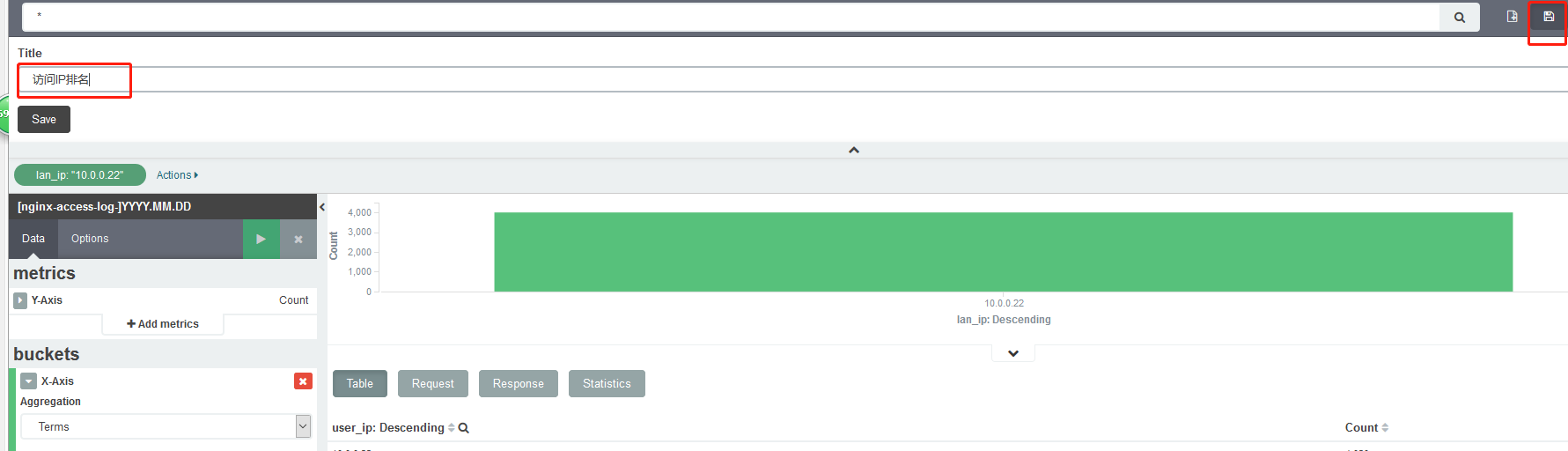
加入dashboard



### 4 Vertical bar chart 显示 IP排名



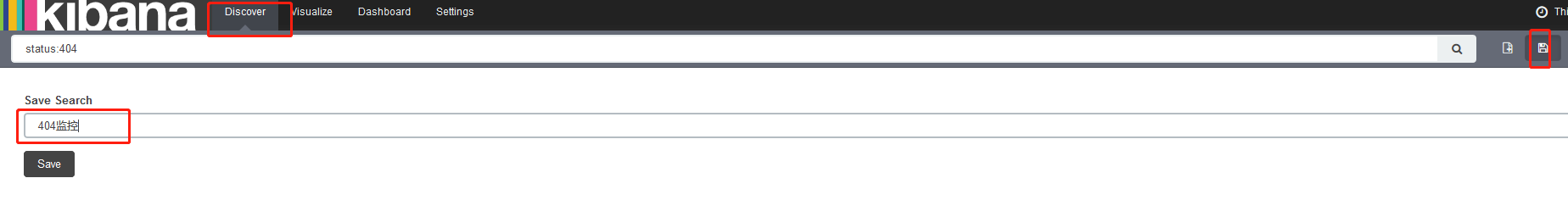


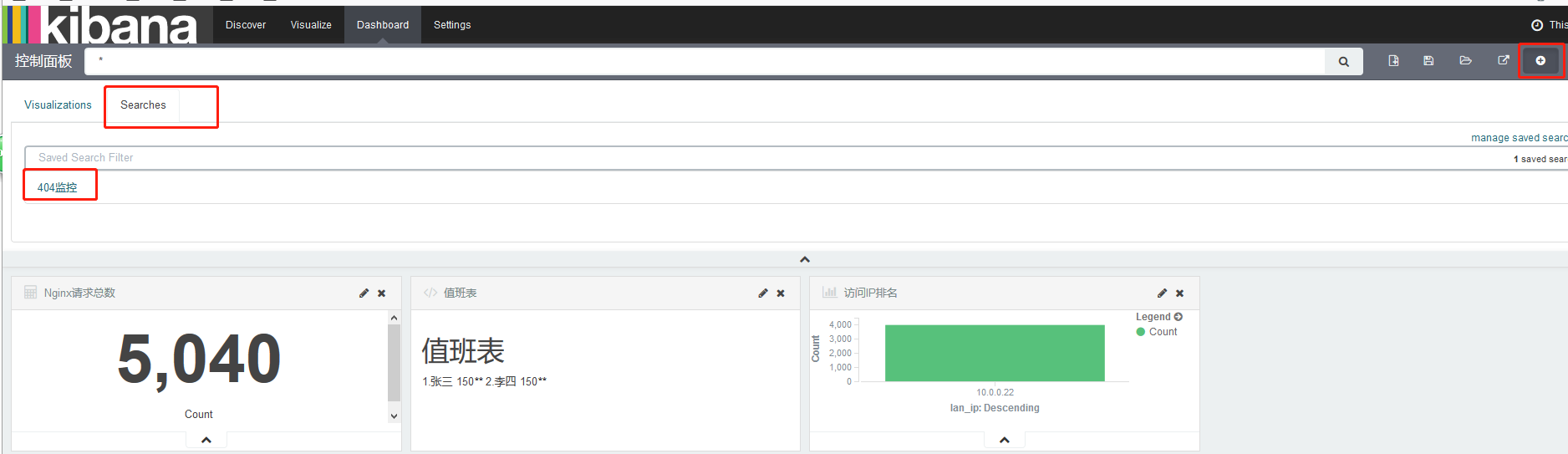


放入dashboard

### 5 一些常用错误日志

先在discover中搜索，然后在dashboard中显示





### 6 补充

生产kibana面板可以：

统计报表、mysql slow日志、nginx 500 错误、访问数、值班表....

1 每个ES上面都启动一个kibana

2 kibana都连当前节点的ES

3 前端用nginx负载均衡 + 验证（用Nginx自带的auth\_basic，Nginx访问认证等）

# 7 logstash收集nginx、syslog和tcp日志实践

## 7.1 收集nginx的访问日志

说明：要收集文件信息，都必须要在收集的节点上装logstash插件

比如node1节点要收集node2节点上的日志信息；操作如下：

1 nginx环境准备

[root@linux-node2 src]# yum install nginx -y

[root@linux-node2 src]# /etc/init.d/nginx start

2 创造访问日志

用apache的ab压力测试插件来创造

yum -y install httpd-tools

[root@linux-node2 src]# ab -n 1000 -c 1 <http://10.0.0.22/>

# 单个并发访问1000次

[root@linux-node2 src]# ll /var/log/nginx/

总用量 96

-rw-r--r-- 1 root root 95000 7月 11 18:10 access.log

-rw-r--r-- 1 root root 0 7月 11 18:05 error.log

3 配置nginx的配置文件nginx.conf的访问日志以json输出

[root@linux-node2 ~]# sed -n '18,33p' /etc/nginx/nginx.conf

http {

log\_format main '$remote\_addr - $remote\_user [$time\_local] "$request" '

'$status $body\_bytes\_sent "$http\_referer" '

'"$http\_user\_agent" "$http\_x\_forwarded\_for"';

log\_format access\_log\_json '{ "user\_ip": "$http\_x\_real\_ip","lan\_ip": "$remote\_addr","log\_time":"$time\_iso8601", "user\_req": "$request", "status": "$status", "body\_bytes\_sent": "$body\_bytes\_sent", "req\_time": "$request\_time", "user\_ua":"$http\_user\_agent"}';

access\_log /var/log/nginx/access\_log\_json.log access\_log\_json;

#access\_log /var/log/nginx/access.log main;

sendfile on;

tcp\_nopush on;

tcp\_nodelay on;

keepalive\_timeout 65;

types\_hash\_max\_size 2048;

include /etc/nginx/mime.types;

default\_type application/octet-stream;

检查语法启动Nginx

[root@linux-node2 ~]# nginx -t

nginx: the configuration file /etc/nginx/nginx.conf syntax is ok

nginx: configuration file /etc/nginx/nginx.conf test is successful

[root@linux-node2 ~]# /etc/init.d/nginx restart

停止 nginx： [确定]

正在启动 nginx： [确定]

[root@linux-node2 ~]# netstat -lntup|grep 80

查看新的日志文件验证：

[root@linux-node2 ~]# tail -f /var/log/nginx/access\_log\_json.log

新开窗口:

[root@linux-node2 ~]# ab -n 1000 -c 1 <http://10.0.0.22/>

日志文件如下，配置成功

{ "user\_ip": "-","lan\_ip": "10.0.0.22","log\_time":"2018-07-11T23:12:17+08:00", "user\_req": "GET / HTTP/1.0", "status": "200", "body\_bytes\_sent": "3698", "req\_time": "0.000", "user\_ua":"ApacheBench/2.3"}

{ "user\_ip": "-","lan\_ip": "10.0.0.22","log\_time":"2018-07-11T23:12:17+08:00", "user\_req": "GET / HTTP/1.0", "status": "200", "body\_bytes\_sent": "3698", "req\_time": "0.000", "user\_ua":"ApacheBench/2.3"}

{ "user\_ip": "-","lan\_ip": "10.0.0.22","log\_time":"2018-07-11T23:12:17+08:00", "user\_req": "GET / HTTP/1.0", "status": "200", "body\_bytes\_sent": "3698", "req\_time": "0.000", "user\_ua":"ApacheBench/2.3"}

{ "user\_ip": "-","lan\_ip": "10.0.0.22","log\_time":"2018-07-11T23:12:17+08:00", "user\_req": "GET / HTTP/1.0", "status": "200", "body\_bytes\_sent": "3698", "req\_time": "0.000", "user\_ua":"ApacheBench/2.3"}

....

4 编写logstash的conf文件，将logstash收集到的信息输出到控制台检测

[root@linux-node2 ~]# cat nginx.conf

input {

file {

path => "/var/log/nginx/access\_log\_json.log"

codec => "json"

}

}

filter {

}

output {

stdout {

codec => rubydebug

}

}

启动logstash收集日志

[root@linux-node2 ~]# /opt/logstash/bin/logstash -f nginx.conf

新窗口添加nginx的访问日志

[root@linux-node2 ~]# ab -n 10 -c 1 <http://10.0.0.22/>

logstash输出的日志如下：

[root@linux-node2 ~]# /opt/logstash/bin/logstash -f nginx.conf

Settings: Default filter workers: 1

Logstash startup completed

{

"user\_ip" => "-",

"lan\_ip" => "10.0.0.22",

"log\_time" => "2018-07-11T23:19:52+08:00",

"user\_req" => "GET / HTTP/1.0",

"status" => "200",

"body\_bytes\_sent" => "3698",

"req\_time" => "0.002",

"user\_ua" => "ApacheBench/2.3",

"@version" => "1",

"@timestamp" => "2018-07-11T15:19:53.204Z",

"path" => "/var/log/nginx/access\_log\_json.log",

"host" => "linux-node2"

}

测试ok

5 在logstash的/etc/logstash/conf.d/下增加nginx日志收集的配置

[root@linux-node2 ~]# mv nginx.conf /etc/logstash/conf.d/

编辑收集nginx日志的logstash的配置文件如下

[root@linux-node2 ~]# cat /etc/logstash/conf.d/nginx.conf

input {

file {

path => "/var/log/nginx/access\_log\_json.log"

codec => "json"

type => "nginx-access-log"

}

}

filter {

}

output {

elasticsearch {

hosts => ["10.0.0.11:9200"]

index => "nginx-access-log-%{+YYYY.MM.dd}"

}

}

测试配置文件

[root@linux-node2 ~]# /opt/logstash/bin/logstash -t -f /etc/logstash/conf.d/nginx.conf

Configuration OK

6 删除测试时候运行的.sincedb\_XXXX的文件

注意测试的时候编写的logstash收集nginx.conf的文件在/root下，所以在/root下删除

操作如下：

[root@linux-node2 ~]# ll -a

总用量 88

dr-xr-x---. 3 root root 4096 7月 11 23:26 .

dr-xr-xr-x. 23 root root 4096 7月 11 21:20 ..

-rw-------. 1 root root 1086 6月 26 01:03 anaconda-ks.cfg

-rw-------. 1 root root 1504 7月 11 16:25 .bash\_history

-rw-r--r--. 1 root root 18 5月 20 2009 .bash\_logout

-rw-r--r--. 1 root root 176 5月 20 2009 .bash\_profile

-rw-r--r--. 1 root root 176 9月 23 2004 .bashrc

-rw-r--r--. 1 root root 100 9月 23 2004 .cshrc

-rw-r--r--. 1 root root 23614 6月 26 01:03 install.log

-rw-r--r--. 1 root root 6240 6月 26 01:02 install.log.syslog

drwxr----- 3 root root 4096 7月 11 17:13 .pki

-rw-r--r-- 1 root root 21 7月 11 23:22 .sincedb\_79929b4271e205874b942fde4119e723

-rw-r--r--. 1 root root 129 12月 4 2004 .tcshrc

-rw------- 1 root root 6454 7月 11 23:26 .viminfo

删除

[root@linux-node2 ~]# rm -rf .sincedb\_79929b4271e205874b942fde4119e723

7 启动logstash

[root@linux-node2 ~]# /etc/init.d/logstash restart

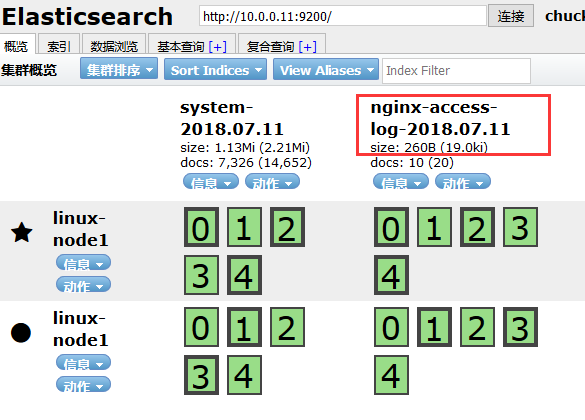
[root@linux-node2 ~]# /opt/logstash/bin/logstash -f /etc/logstash/conf.d/nginx.conf

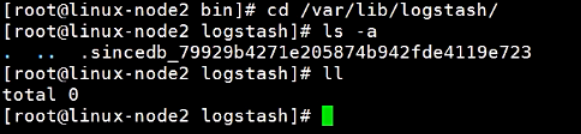
8 测试

添加nginx访问日志

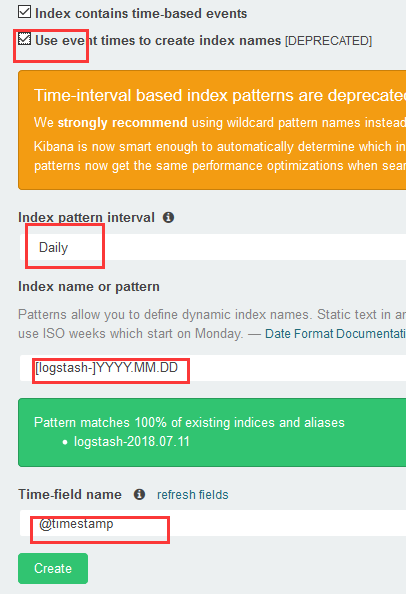
[root@linux-node2 ~]# ab -n 10 -c 1 http://10.0.0.22/

在head中查看

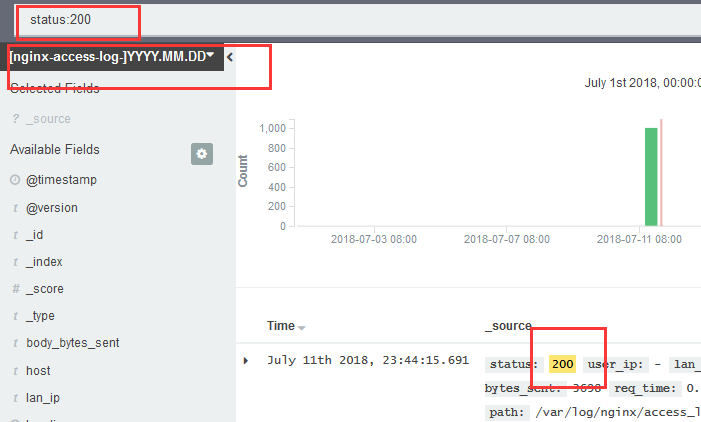




9 kibana加上nginx的index



搜索测试



## 7.2 收集系统Rsyslog日志

前文中已经使用文件file的形式收集了系统日志/var/log/messages，但是实际生产环境是需要使用syslog插件直接收集

修改syslog的配置文件，把日志信息发送到514端口上

官方文档：https://www.elastic.co/guide/en/logstash/2.4/plugins-inputs-syslog.html

1 编写logstash的conf文件，将logstash收集到的信息输出到控制台检测

[root@linux-node2 ~]# vim /etc/logstash/conf.d/syslog.conf

input {

syslog {

type => "system-syslog"

port => 514

}

}

output {

stdout {

codec => rubydebug

}

}

对上面的采集文件进行执行：

[root@linux-node2 ~]# /opt/logstash/bin/logstash -f /etc/logstash/conf.d/syslog.conf

重新开启一个窗口，查看服务是否启动：

[root@linux-node2 ~]# netstat -lnup

Active Internet connections (only servers)

Proto Recv-Q Send-Q Local Address Foreign Address State PID/Program name

udp 0 0 :::514 :::\* 30112/java

编译系统的rsyslog.conf

[root@linux-node2 ~]# vim /etc/rsyslog.conf

#\*.\* @@remote-host:514 # 在这行下面添加如下内容

\*.\* @@10.0.0.22:514

重启

[root@linux-node2 ~]# /etc/init.d/rsyslog restart

回到原来的窗口(即上面采集文件的执行终端)，就会出现数据：查看日志

[root@linux-node2 ~]# /opt/logstash/bin/logstash -f /etc/logstash/conf.d/syslog.conf

Settings: Default filter workers: 1

Logstash startup completed

{

"message" => "imklog 5.8.10, log source = /proc/kmsg started.\n",

"@version" => "1",

"@timestamp" => "2000-07-12T01:34:21.000Z",

"type" => "system-syslog",

"host" => "10.0.0.22",

"priority" => 6,

"timestamp" => "Jul 12 09:34:21",

"logsource" => "linux-node2",

"program" => "kernel",

"severity" => 6,

"facility" => 0,

"facility\_label" => "kernel",

"severity\_label" => "Informational"

}

...

2 修改配置，写入es中

[root@linux-node2 ~]# vim /etc/logstash/conf.d/syslog.conf

input {

syslog {

type => "system-syslog"

port => 514

}

}

output {

elasticsearch {

hosts => ["10.0.0.11:9200"]

index => "system-syslog-%{+YYYY.MM}"

}

}

检测配置文件语法

[root@linux-node2 ~]# /opt/logstash/bin/logstash -t -f /etc/logstash/conf.d/syslog.conf

Configuration OK

执行logstash配置文件

[root@linux-node2 ~]# /opt/logstash/bin/logstash -f /etc/logstash/conf.d/syslog.conf

3 向日志中添加测试数据

[root@linux-node2 ~]# logger "hehe"

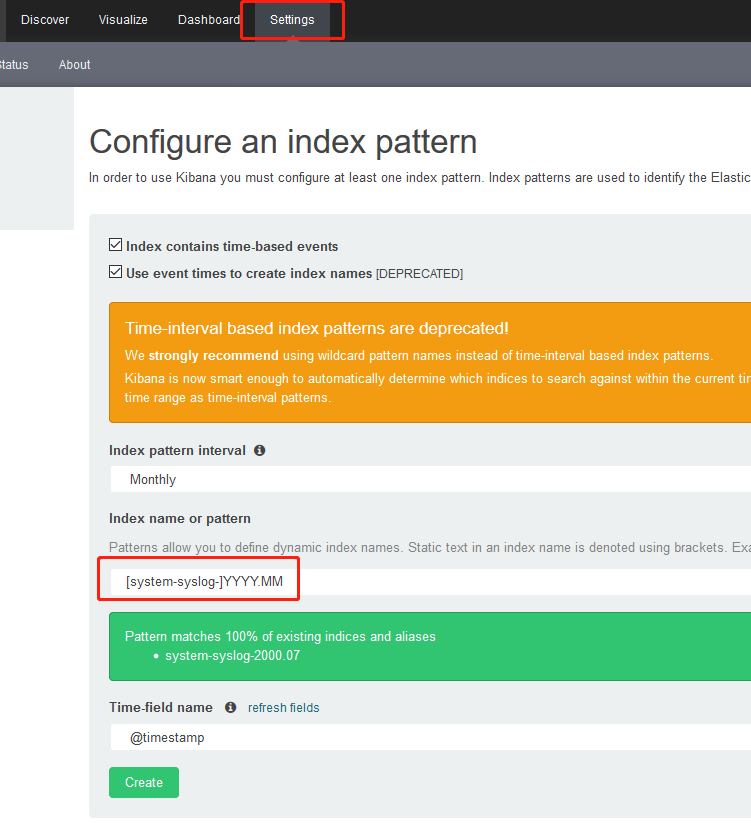
[root@linux-node2 ~]# logger "hehe2"

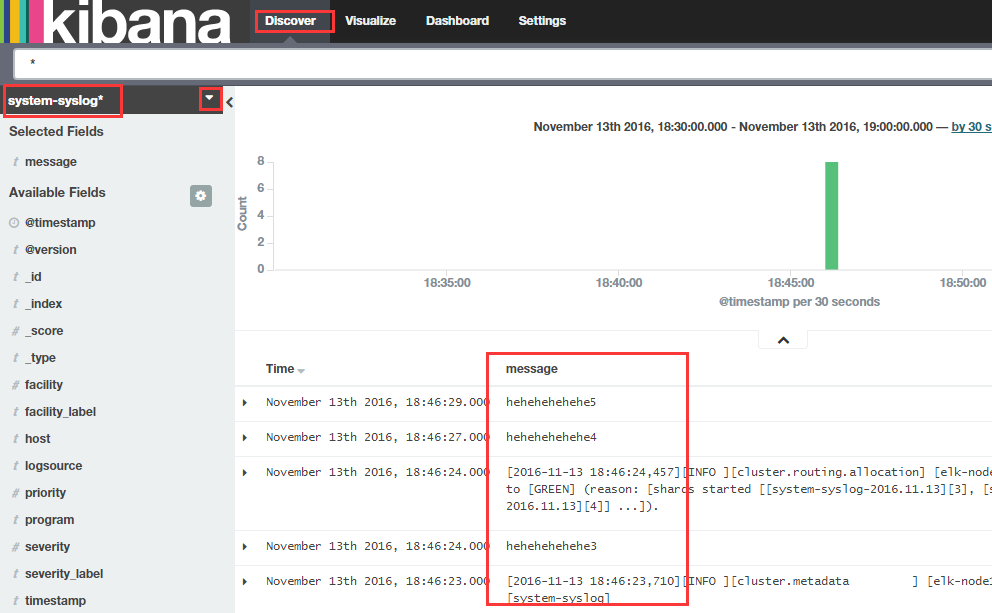
[root@linux-node2 ~]# logger "hehe3"

在head中验证查看



4 添加到kibana界面中





## 7.3 tcp日志的收集

1 编写logstash的conf文件，将logstash收集到的信息输出到控制台检测

[root@linux-node2 ~]# vim /etc/logstash/conf.d/tcp.conf

input {

tcp {

type => "tcp"

port => "6666"

mode => "server"

}

}

output {

stdout {

codec => rubydebug

}

}

[root@linux-node2 ~]# /opt/logstash/bin/logstash -f /etc/logstash/conf.d/tcp.conf

开启另外一个窗口，测试一（安装nc命令：yum install -y nc）：

[root@linux-node2 ~]# nc 192.168.1.160 6666 </etc/resolv.conf

[root@linux-node2 ~]# netstat -lntp

Active Internet connections (only servers)

Proto Recv-Q Send-Q Local Address Foreign Address State PID/Program name

tcp 0 0 0.0.0.0:80 0.0.0.0:\* LISTEN 28454/nginx

tcp 0 0 0.0.0.0:22 0.0.0.0:\* LISTEN 1323/sshd

tcp 0 0 :::6666 :::\* LISTEN 30814/java

tcp 0 0 :::80 :::\* LISTEN 28454/nginx

tcp 0 0 :::9200 :::\* LISTEN 27345/java

tcp 0 0 :::9300 :::\* LISTEN 27345/java

tcp 0 0 :::22 :::\* LISTEN 1323/sshd

给tcp发送内容

[root@linux-node2 ~]# echo "hehe" | nc 10.0.0.22 6666

回到原来窗口

[root@linux-node2 ~]# /opt/logstash/bin/logstash -f /etc/logstash/conf.d/tcp.conf

Settings: Default filter workers: 1

Logstash startup completed

{

"message" => "hehe",

"@version" => "1",

"@timestamp" => "2018-07-12T03:36:33.379Z",

"host" => "10.0.0.22",

"port" => 37125,

"type" => "tcp"

}

补充：

1. 发送某个文件

[root@linux-node2 ~]# nc 10.0.0.22 6666 < /etc/resolv.conf

2）伪设备的方式 发送数据

[root@linux-node2 ~]# echo "hehe2" > /dev/tcp/10.0.0.22/6666

2 配置logstash通过TCP收集输出到elasticsearch

input {

tcp {

type => "tcp"

port => "6666"

mode => "server"

}

}

output {

elasticsearch {

hosts => ["10.0.0.11:9200"]

index => "tcp-log-%{+YYYY.MM}"

}

}

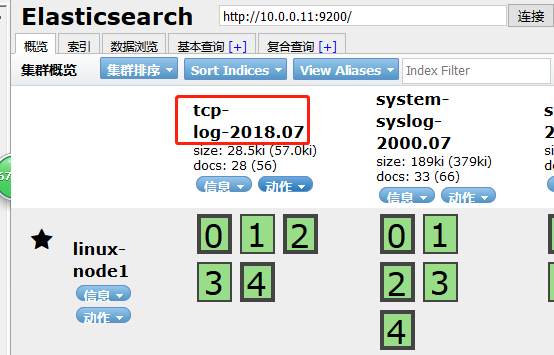
执行logstash配置文件

[root@linux-node2 ~]# /opt/logstash/bin/logstash -f /etc/logstash/conf.d/tcp.conf

通过tcp写入数据

[root@linux-node2 ~]# nc 10.0.0.22 6666 < /etc/passwd

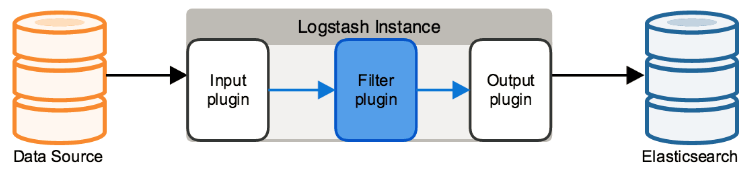
head查看



加入kibana

# 8 logstash的fliter插件

## 8.1 grok插件



filter插件有很多，在这里就学习grok插件，使用正则匹配日志里的域来拆分。在实际生产中，apache日志不支持jason，就只能使用grok插件匹配；mysql慢查询日志也是无法拆分，只能石油grok正则表达式匹配拆分。

在如下链接，github上有很多写好的grok模板，可以直接引用

<https://github.com/logstash-plugins/logstash-patterns-core/blob/master/patterns/grok-patterns>

在装好的logstash中也会有grok匹配规则，直接可以引用，路径如下

[root@linux-node1 patterns]# pwd

/opt/logstash/vendor/bundle/jruby/1.9/gems/logstash-patterns-core-2.0.2/patterns

编写文件

[root@linux-node2 ~]# vim /etc/logstash/conf.d/grok.conf

input {

stdin {}

}

filter {

grok {

match => { "message" => "%{IP:client} %{WORD:method} %{URIPATHPARAM:request} %{NUMBER:bytes} %{NUMBER:duration}" }

}

}

output {

stdout {

codec => "rubydebug"

}

}

执行测试

[root@linux-node2 ~]# /opt/logstash/bin/logstash -f /etc/logstash/conf.d/grok.conf

Settings: Default filter workers: 1

Logstash startup completed

10.0.0.11 GET /index.html 15824 0.043 #模拟的数据

{

"message" => "10.0.0.11 GET /index.html 15824 0.043",

"@version" => "1",

"@timestamp" => "2018-07-12T05:32:10.025Z",

"host" => "linux-node2",

"client" => "10.0.0.11",

"method" => "GET",

"request" => "/index.html",

"bytes" => "15824",

"duration" => "0.043"

}

## 8.2 收集mysql慢查询

[root@linux-node2 ~]# vim /etc/logstash/conf.d/mysql-slow.conf

input {

file {

path => "/root/slow.log"

type => "mysql-slowlog"

codec => multiline {

pattern => "^# User@Host"

negate => true

what => "previous"

}

}

}

filter {

# drop sleep events

grok {

match => { "message" =>"SELECT SLEEP" }

add\_tag => [ "sleep\_drop" ]

tag\_on\_failure => [] # prevent default \_grokparsefailure tag on real records

}

if "sleep\_drop" in [tags] {

drop {}

}

grok {

match => [ "message", "(?m)^# User@Host: %{USER:user}\[[^\]]+\] @ (?:(?<clienthost>\S\*) )?\[(?:%{IP:clientip})?\]\s+Id: %{NUMBER:row\_id:int}\s\*# Query\_time: %{NUMBER:query\_time:float}\s+Lock\_time: %{NUMBER:lock\_time:float}\s+Rows\_sent: %{NUMBER:rows\_sent:int}\s+Rows\_examined: %{NUMBER:rows\_examined:int}\s\*(?:use %{DATA:database};\s\*)?SET timestamp=%{NUMBER:timestamp};\s\*(?<query>(?<action>\w+)\s+.\*)\n#\s\*" ]

}

date {

match => [ "timestamp", "UNIX" ]

remove\_field => [ "timestamp" ]

}

}

output {

stdout {

codec =>"rubydebug"

}

}

执行检测：

上面需要的/root/slow.log是自己上传的，然后自己插入数据保存后，会显示：

[root@linux-node2 ~]# /opt/logstash/bin/logstash -f /etc/logstash/conf.d/mysql-slow.conf

Settings: Default filter workers: 1

Logstash startup completed

{

"@timestamp" => "2016-11-14T06:53:54.100Z",

"message" => "# Time: 161114 11:05:18",

"@version" => "1",

"path" => "/root/slow.log",

"host" => "elk-node1",

"type" => "mysql-slowlog",

"tags" => [

[0] "\_grokparsefailure"

]

}

{

"@timestamp" => "2016-11-14T06:53:54.105Z",

"message" => "# User@Host: test[test] @ [124.65.197.154]\n# Query\_time: 1.725889 Lock\_time: 0.000430 Rows\_sent: 0 Rows\_examined: 0\nuse test\_zh\_o2o\_db;\nSET timestamp=1479092718;\nSELECT trigger\_name, event\_manipulation, event\_object\_table, action\_statement, action\_timing, DEFINER FROM information\_schema.triggers WHERE BINARY event\_object\_schema='test\_zh\_o2o\_db' AND BINARY event\_object\_table='customer';\n# Time: 161114 12:10:30",

"@version" => "1",

"tags" => [

[0] "multiline",

[1] "\_grokparsefailure"

],

"path" => "/root/slow.log",

"host" => "elk-node1",

"type" => "mysql-slowlog"

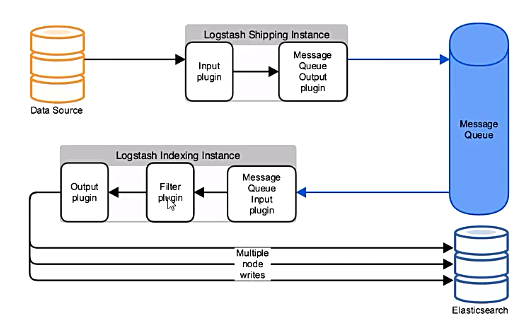
}

...

## 8.3 收集apache日志

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

# 9 优化-消息队列redis引入



数据源Datasource把数据写到input插件中，output插件使用消息队列把消息写入到消息队列Message Queue中，Logstash indexing Instance启动logstash使用input插件读取消息队列中的信息，Fliter插件过滤后在使用output写入到elasticsearch中。

如果生产环境中不适用正则grok匹配，可以写Python脚本从消息队列中读取信息，输出到elasticsearch中

优点：

解耦，松耦合

解除了由于网络原因不能直接连elasticsearch的情况

方便架构演变，增加新内容

消息队列可以使用rabbitmq，zeromq等，也可以使用redis，kafka（消息不删除，但是比较重量级）

## 9.1 安装配置redis

[root@linux-node1 ~]# yum install redis

1. redis的配置和启动

[root@linux-node1 ~]# vim /etc/redis.conf

daemonize yes

bing 10.0.0.11

[root@linux-node1 ~]# /etc/init.d/redis start

启动 ： [确定]

[root@linux-node1 ~]# netstat -lntup|grep 6379

tcp 0 0 10.0.0.11:6379 0.0.0.0:\* LISTEN 30889/redis-server

1. 编写从Client端收集数据的文件--测试的conf

官方文档：<https://www.elastic.co/guide/en/logstash/2.4/plugins-inputs-redis.html>

[root@linux-node1 ~]# vim redis-out.conf

input {

stdin {}

}

output {

redis {

host => "10.0.0.11"

port => "6379"

db => "6"

data\_type => "list"

key => "demo"

}

}

1. 执行收集数据的文件，并输入数据hello redis 测试

[root@linux-node1 ~]# /opt/logstash/bin/logstash -f redis-out.conf

Settings: Default filter workers: 1

Logstash startup completed

hello redis

4）在redis中查看数据

10.0.0.11:6379> info

# Server

......

# Keyspace

db6:keys=1,expires=0,avg\_ttl=0

10.0.0.11:6379> select 6 #选择db6

OK

10.0.0.11:6379[6]> keys \* #查看所有的索引

1) "demo"

10.0.0.11:6379[6]> LINDEX demo -1 #查看消息

"{\"message\":\"hello redis\",\"@version\":\"1\",\"@timestamp\":\"2018-07-12T07:10:44.462Z\",\"host\":\"linux-node1\"}"

10.0.0.11:6379[6]> llen demo #查看长度

(integer) 1

5）在 /opt/logstash/bin/logstash -f redis-out.conf 多增加些测试数据

[root@linux-node1 ~]# /opt/logstash/bin/logstash -f redis-out.conf

Settings: Default filter workers: 1

Logstash startup completed

hello redis

dhw

hdowhe

heowhe

hwoewhowu

eee

1. 将redis中的内容写到es中

[root@linux-node1 ~]# vim redis-in.conf

input {

redis {

host => "10.0.0.11"

port => "6379"

db => "6"

data\_type => "list"

key => "demo"

}

}

output {

elasticsearch {

hosts => ["10.0.0.11:9200"]

index => "redis-in-%{+YYYY.MM.dd}"

}

}

执行

[root@linux-node1 ~]# /opt/logstash/bin/logstash -t -f redis-in.conf

Configuration OK

[root@linux-node1 ~]# /opt/logstash/bin/logstash -f redis-in.conf

登录redis 查看数据是否还在

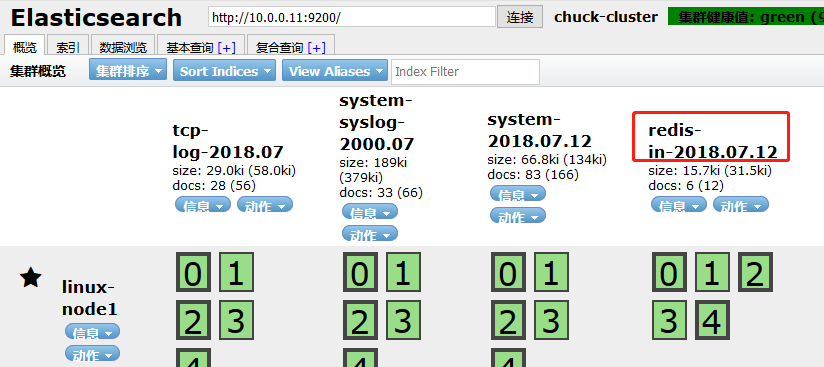
[root@linux-node1 ~]# redis-cli -h 10.0.0.11

10.0.0.11:6379> select 6

10.0.0.11:6379[6]> llen demo

(integer) 0 # 数据为空，说明被消耗掉了

在es集群中查看



## 9.2 改成全部经由redis

将收集到的所有日志写入到redis中。这了重新定义一个添加redis缓存后的总文件shipper.conf。（可以将之前执行的文件全部停掉）

[root@linux-node2 conf.d]# pwd

/etc/logstash/conf.d

[root@linux-node2 conf.d]# vim shipper.conf

input {

syslog {

type => "system-syslog"

host => "10.0.0.22"

port => "514"

}

tcp {

type => "tcp-6666"

host => "10.0.0.22"

port => "6666"

}

file {

path => "/var/log/messages"

type => "system"

start\_position => "beginning"

}

file {

path => "/var/log/nginx/access\_log\_json.log"

codec => json

start\_position => "beginning"

type => "nginx-access-log"

}

file {

path => "/var/log/elasticsearch/chuck-cluster.log"

type => "es-error"

start\_position => "beginning"

codec => multiline {

pattern => "^\["

negate => true

what => "previous"

}

}

}

output {

if [type] == "system" {

redis {

host => "10.0.0.11"

port => "6379"

db => "6"

data\_type => "list"

key => "system"

}

}

if [type] == "es-error" {

redis {

host => "10.0.0.11"

port => "6379"

db => "6"

data\_type => "list"

key => "es-error"

}

}

if [type] == "nginx-access-log" {

redis {

host => "10.0.0.11"

port => "6379"

db => "6"

data\_type => "list"

key => "nginx-access-log"

}

}

if [type] == "system-syslog" {

redis {

host => "10.0.0.11"

port => "6379"

db => "6"

data\_type => "list"

key => "system-syslog"

}

}

if [type] == "tcp-6666" {

redis {

host => "10.0.0.11"

port => "6379"

db => "6"

data\_type => "list"

key => "tcp-6666"

}

}

}

执行shipper.conf

[root@linux-node2 conf.d]# /opt/logstash/bin/logstash -t -f /etc/logstash/conf.d/shipper.conf

Configuration OK

[root@linux-node2 conf.d]# /opt/logstash/bin/logstash -f /etc/logstash/conf.d/shipper.conf

Settings: Default filter workers: 1

Logstash startup completed

登录redis查看

[root@linux-node1 ~]# redis-cli -h 10.0.0.11

10.0.0.11:6379> select 6

OK

10.0.0.11:6379[6]> keys \*

1) "nginx-access-log"

2) "es-error"

3) "system"

10.0.0.11:6379[6]> llen nginx-access-log

(integer) 6050

10.0.0.11:6379[6]> llen es-error

(integer) 42

10.0.0.11:6379[6]> llen system

(integer) 7219

# redis成功获取到了数据

另开一个窗口，添加点日志:

[root@elk-node2 ~]# logger "2222222"

查看redis

10.0.0.11:6379[6]> keys \*

1) "system-syslog"

2) "nginx-access-log"

3) "es-error"

4) "system"

## 9.3 将数据从redis读取到ES中

1 编写indexer.conf作为redis发送elasticsearch配置文件

说明：这里在哪个node上配置都是OK的，这里选择的是node2

[root@linux-node2 conf.d]# cat indexer.conf

input {

redis {

type => "system-syslog"

host => "10.0.0.11"

port => "6379"

db => "6"

data\_type => "list"

key => "system-syslog"

}

redis {

type => "tcp-6666"

host => "10.0.0.11"

port => "6379"

db => "6"

data\_type => "list"

key => "tcp-6666"

}

redis {

type => "system"

host => "10.0.0.11"

port => "6379"

db => "6"

data\_type => "list"

key => "system"

}

redis {

type => "nginx-access-log"

host => "10.0.0.11"

port => "6379"

db => "6"

data\_type => "list"

key => "nginx-access-log"

}

redis {

type => "es-error"

host => "10.0.0.11"

port => "6379"

db => "6"

data\_type => "list"

key => "es-error"

}

}

output {

if [type] == "system" {

elasticsearch {

hosts => "10.0.0.11"

index => "system-%{+YYYY.MM}"

}

}

if [type] == "es-error" {

elasticsearch {

hosts => "10.0.0.11"

index => "es-error-%{+YYYY.MM}"

}

}

if [type] == "nginx-access-log" {

elasticsearch {

hosts => "10.0.0.11"

index => "nginx-access-log-%{+YYYY.MM.dd}"

}

}

if [type] == "system-syslog" {

elasticsearch {

hosts => "10.0.0.11"

index => "system-syslog-%{+YYYY.MM}"

}

}

if [type] == "tcp-6666" {

elasticsearch {

hosts => "10.0.0.11"

index => "tcp-6666-%{+YYYY.MM}"

}

}

}

执行：

[root@linux-node2 conf.d]# /opt/logstash/bin/logstash -t -f /etc/logstash/conf.d/indexer.conf

Configuration OK

[root@linux-node2 conf.d]# /opt/logstash/bin/logstash -f /etc/logstash/conf.d/indexer.conf

登录redis查看

10.0.0.11:6379[6]> keys \*

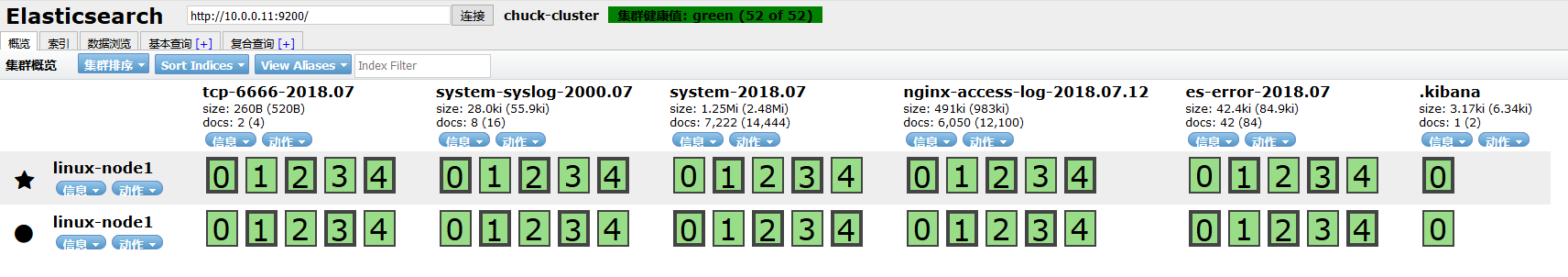
(empty list or set)

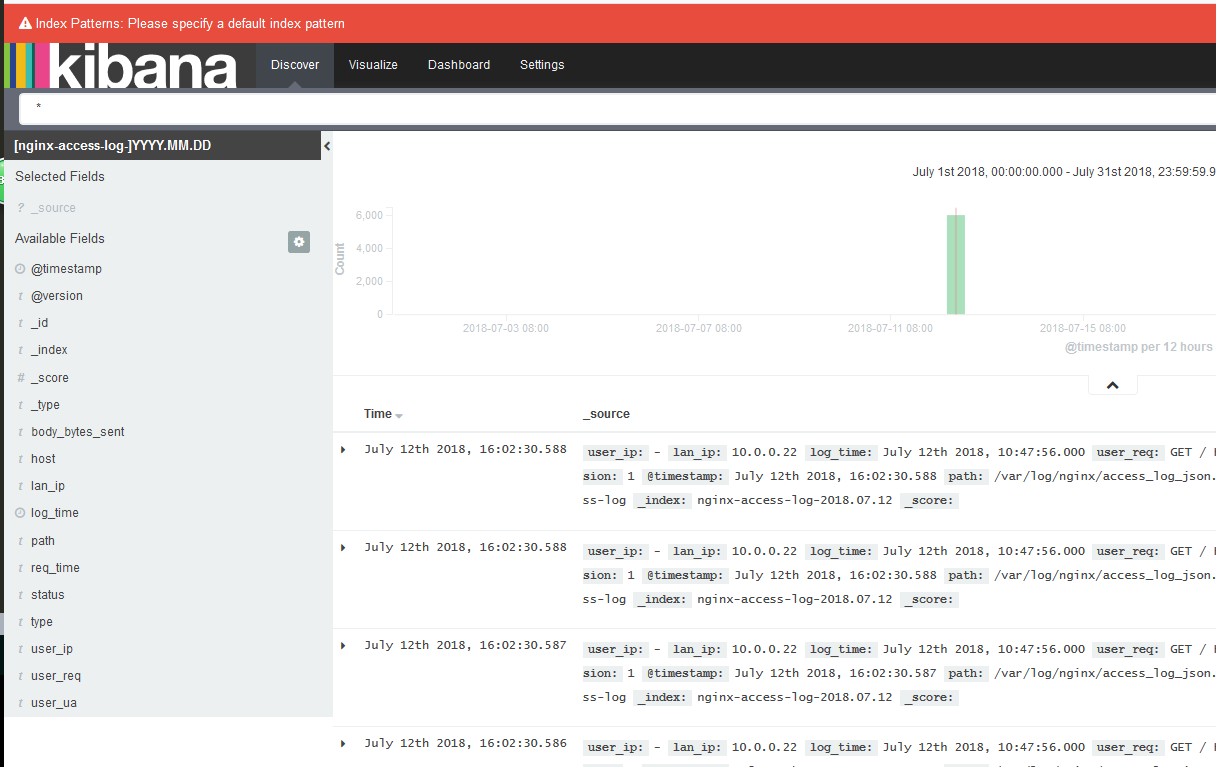
创造测试数据

[root@linux-node2 conf.d]# nc 10.0.0.22 6666 </etc/resolv.conf #测试tcp

[root@linux-node2 conf.d]# logger "eee"

用head插件查看kibana添加查看





# 10 生产实践

需求分析：

访问日志：apache访问日志、nginx访问日志、tomcat; 处理方式file -filter

错误日志：error log、java日志；处理方式直接获取，Java异常需要用multiline

系统日志：/var/logs/\*、syslog; 处理方式rsyslog

运行日志：程序写的（要求json）；处理方式file

网络日志：防火墙、交换机、路由器的日志 处理方式syslog

1 标准化：

日志路径：/data/logs/，格式要求是json

命名规则：/data/logs/access\_log /data/logs/error\_log /data/logs/runtime\_log

日志切割：access error用crontab定时用脚本进行切分；runtime\_log开发指定。

access按小时切分，error按天切分（根据业务具体而定）

日志备份：所有的原始文本，rsync到NAS，然后删除最近三天前的日志。

2 工具化：

用logstash处理。所有要收集的日志都放在一个配置文本

/etc/logstash/conf.d/shipper.conf # 获得信息写入redis

/etc/logstash/conf.d/indexer.conf # 获得redis信息写入es

也可以用提前的消息队列，比如：kafka rabbitmq等

参考：

<https://segmentfault.com/a/1190000011263254>

<http://blog.oldboyedu.com/elk/>

<https://www.cnblogs.com/kevingrace/p/5919021.html>