ELK 搭建

摘要： https://blog.51cto.com/zero01/2082794

**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，是当前流行的企业级搜索引擎。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。

**Logstash:**

用于管理日志和事件的工具，你可以用它去收集日志、转换日志、解析日志并将他们作为数据提供给其它模块调用，例如搜索、存储等。

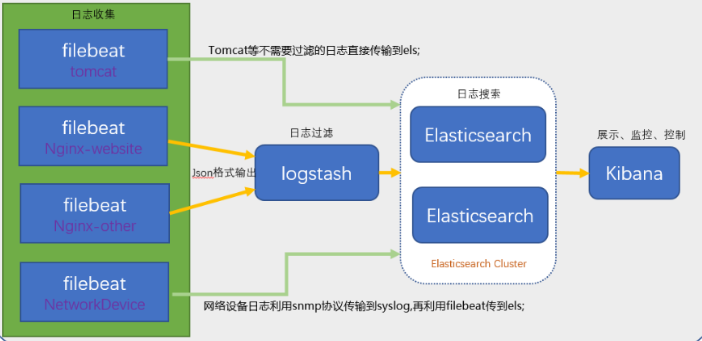
**Kibana:**

是一个优秀的前端日志展示框架，它可以非常详细的将日志转化为各种图表，为用户提供强大的数据可视化支持。

**Filebeat:**

类似于 “agent” 装在被监控端上（数据源），用来实时收集日志文件数据。

**ELK架构图：**



1. 下载包

#wget https://artifacts.elastic.co/downloads/elasticsearch/elasticsearch-6.4.0.tar.gz

#wget https://artifacts.elastic.co/downloads/logstash/logstash-6.4.0.tar.gz

#wget https://artifacts.elastic.co/downloads/kibana/kibana-6.4.0-linux-x86\_64.tar.gz

#wget <https://artifacts.elastic.co/downloads/beats/filebeat/filebeat-6.4.0-linux-x86_64.tar.gz>

#wget <https://nodejs.org/dist/v6.10.2/node-v6.10.2-linux-x64.tar.xz>

（由于head插件本质上还是一个nodejs的工程，因此需要安装node，使用npm来安装依赖的包如果wget不了直接下载）

1. 安装jdk

Pass

# 安装elasticsearch

1. 解压elasticsearch

[root@qishengkun ~]# tar xf elasticsearch-6.4.0.tar.gz -C /usr/local/

1. 将其改名为elasticsearch

[root@qishengkun ~]# mv /usr/local/elasticsearch-6.4.0 /usr/local/elasticsearch

1. 修改elasticsearch配置

[root@qishengkun elasticsearch]# cd /usr/local/elasticsearch/config/

[root@qishengkun config]# vim elasticsearch.yml

17 cluster.name: es

33 path.data: /data/elk/data

37 path.logs: /data/elk/logs

43 bootstrap.memory\_lock: false

44 bootstrap.system\_call\_filter: false （CenterOS6版本需要在Memory里添加）

56 network.host: 192.168.176.128 （本地IP）

60 http.port: 9200

1. 创建目录

[root@qishengkun config]# mkdir -p /data/elk/{data,logs}

1. 添加普通账号（默认必须普通用户启动）

[root@qishengkun config]# useradd qi

1. 给普通账号创建密码

[root@qishengkun config]# passwd qi

1. 给目录添加权限

[root@qishengkun config]# cd

[root@qishengkun ~]# chown -R qi.qi elasticsearch/

[root@qishengkun ~]# chown -R qi.qi /data/

1. 登陆普通账号

[root@qishengkun ~]# su – qi

1. 使用普通账户启动elasticsearch

[qi@qishengkun ~]$ cd /usr/local/elasticsearch/bin/

[qi@qishengkun bin]$ ./elasticsearch

1. 报错
2. max file descriptors [65535] for elasticsearch process is too low, increase to at least [65536]

编辑 /etc/security/limits.conf，追加以下内容：

\* soft nofile 65536

\* hard nofile 65536

(此文件修改后需要重新登录用户，才会生效)

1. max virtual memory areas vm.max\_map\_count [65530] is too low, increase to at least [262144]

编辑 /etc/sysctl.conf，追加以下内容：

vm.max\_map\_count=655360

保存后，执行：

[root@qishengkun ~]# sysctl -p

1. bootstrap checks failed

编辑 /etc/security/limits.d/90-nproc.conf，内容

[root@qishengkun ~]# vim /etc/security/limits.d/90-nproc.conf

\* soft nproc 4096

root soft nproc unlimited

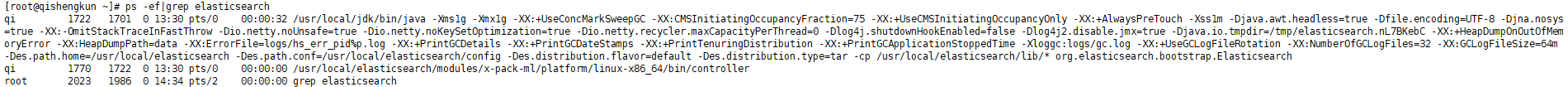
1. 在后台启动elasticsearch

[qi@qishengkun ~]$ cd /usr/local/elasticsearch/bin/

[qi@qishengkun bin]$ ./elasticsearch -d

1. 测试启动情况

[root@qishengkun ~]# ps -ef|grep elasticsearch



下面主要讲解下elasticsearch.yml这个文件中可配置的东西  
cluster.name:elasticsearch  
配置es的集群名称，默认是elasticsearch，es会自动发现在同一网段下的es，如果在同一网段下有多个集群，就可以用这个属性来区分不同的集群。  
node.name:”FranzKafka”  
节点名，默认随机指定一个name列表中名字，该列表在es的jar包中config文件夹里name.txt文件中，其中有很多作者添加的有趣名字。  
node.master:true  
指定该节点是否有资格被选举成为node，默认是true，es是默认集群中的第一台机器为master，如果这台机挂了就会重新选举master。  
node.data:true  
指定该节点是否存储索引数据，默认为true。  
index.number\_of\_shards:5  
设置默认索引分片个数，默认为5片。  
index.number\_of\_replicas:1  
设置默认索引副本个数，默认为1个副本。  
path.conf:/path/to/conf  
设置配置文件的存储路径，默认是es根目录下的config文件夹。  
path.data:/path/to/data  
设置索引数据的存储路径，默认是es根目录下的data文件夹，可以设置多个存储路径，用逗号隔开，例：  
path.data:/path/to/data1,/path/to/data2  
path.work:/path/to/work  
设置临时文件的存储路径，默认是es根目录下的work文件夹。  
path.logs:/path/to/logs  
设置日志文件的存储路径，默认是es根目录下的logs文件夹  
path.plugins:/path/to/plugins  
设置插件的存放路径，默认是es根目录下的plugins文件夹  
bootstrap.mlockall:true  
设置为true来锁住内存。因为当jvm开始swapping时es的效率会降低，所以要保证它不swap，可以把ES\_MIN\_MEM和ES\_MAX\_MEM两个环境变量设置成同一个值，并且保证机器有足够的内存分配给es。同时也要允许elasticsearch的进程可以锁住内存，linux下可以通过`ulimit-lunlimited`命令。  
network.bind\_host:192.168.0.1  
设置绑定的ip地址，可以是ipv4或ipv6的，默认为0.0.0.0。network.publish\_host:192.168.0.1  
设置其它节点和该节点交互的ip地址，如果不设置它会自动判断，值必须是个真实的ip地址。  
network.host:192.168.0.1  
这个参数是用来同时设置bind\_host和publish\_host上面两个参数。  
transport.tcp.port:9300  
设置节点间交互的tcp端口，默认是9300。  
transport.tcp.compress:true  
设置是否压缩tcp传输时的数据，默认为false，不压缩。  
http.port:9200  
设置对外服务的http端口，默认为9200。  
http.max\_content\_length:100mb  
设置内容的最大容量，默认100mb  
http.enabled:false  
是否使用http协议对外提供服务，默认为true，开启。  
gateway.type:local  
gateway的类型，默认为local即为本地文件系统，可以设置为本地文件系统，分布式文件系统，hadoop的HDFS，和amazon的s3服务器，其它文件系统的设置方法下次再详细说。  
gateway.recover\_after\_nodes:1  
设置集群中N个节点启动时进行数据恢复，默认为1。  
gateway.recover\_after\_time:5m  
设置初始化数据恢复进程的超时时间，默认是5分钟。  
gateway.expected\_nodes:2  
设置这个集群中节点的数量，默认为2，一旦这N个节点启动，就会立即进行数据恢复。  
cluster.routing.allocation.node\_initial\_primaries\_recoveries:4  
初始化数据恢复时，并发恢复线程的个数，默认为4。  
cluster.routing.allocation.node\_concurrent\_recoveries:2  
添加删除节点或负载均衡时并发恢复线程的个数，默认为4。  
indices.recovery.max\_size\_per\_sec:0  
设置数据恢复时限制的带宽，如入100mb，默认为0，即无限制。  
indices.recovery.concurrent\_streams:5  
设置这个参数来限制从其它分片恢复数据时最大同时打开并发流的个数，默认为5。  
discovery.zen.minimum\_master\_nodes:1  
设置这个参数来保证集群中的节点可以知道其它N个有master资格的节点。默认为1，对于大的集群来说，可以设置大一点的值（2-4）  
discovery.zen.ping.timeout:3s  
设置集群中自动发现其它节点时ping连接超时时间，默认为3秒，对于比较差的网络环境可以高点的值来防止自动发现时出错。  
discovery.zen.ping.multicast.enabled:false  
设置是否打开多播发现节点，默认是true。  
discovery.zen.ping.unicast.hosts:[“host1″,”host2:port”,”host3[portX-portY]”]  
设置集群中master节点的初始列表，可以通过这些节点来自动发现新加入集群的节点

1. 使用xz命令解压xz包，解压出来的结果是tar包node-v6.10.2-linux-x64.tar.xz

[root@qishengkun ~]# xz -d node-v6.10.2-linux-x64.tar.xz

1. 如果xz命令没有的话直接yum安装

[root@qishengkun ~]# yum -y install xz

1. 解压tar包

[root@qishengkun ~]# tar -xvf node-v6.10.2-linux-x64.tar -C /usr/local

1. 将其重命名

[root@qishengkun ~]# cd /usr/local/

[root@qishengkun local]# mv node-v6.10.2-linux-x64 /usr/local/node

1. 配置全局变量

[root@qishengkun local]# vim /etc/profile

NODE\_HOME=/usr/local/node

PATH=$PATH:$NODE\_HOME/bin

export PATH NOTH\_HOME

1. 重新加载环境变量配置文件

[root@qishengkun local]# source /etc/profile

1. 验证node.js是否安装成功，如果能查看到版本号说明安装成功

[root@qishengkun local]# node -v



1. 验证npm工具是否安装成功，如果能查看到版本号说明安装成功

[root@qishengkun local]# npm -v



1. 安装git库

[root@qishengkun local]# yum install –y git

1. 下载head插件

[root@qishengkun local]# git clone git://github.com/mobz/elasticsearch-head.git

《25.26 和 27.28操作一种就可以，建议使用后者》

1. 安装grunt工具，由于在国内连接国外的镜像速度奇慢无比，所以我们使用了国内taobao的镜像。

[root@qishengkun local]# cd elasticsearch-head

[root@qishengkun elasticsearch-head]#

npm install -g grunt --registry=https://registry.npm.taobao.org

1. 编译elasticsearch-head源码

[root@qishengkun elasticsearch-head]# npm install

使用npm编译es-head源码，但是建议大家不要用这个这种方式安装，因为使用npm编译源码会下载很多依赖，这些依赖默认到国外镜像下载，所以速度会很慢

1. 安装cnpm 命令，这是链接中国的镜像

[root@qishengkun elasticsearch-head]#

npm install -g cnpm --registry=https://registry.npm.taobao.org

1. 使用cnpm代替npm编译es-head源码

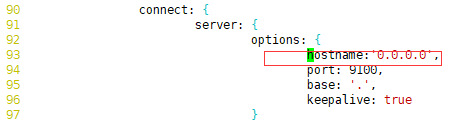
[root@qishengkun elasticsearch-head]# cnpm install

编译好后es-head根目录下会出现一个叫node\_modules的目录，该目录就是存放源码编译后的可执行文件

1. 设置插件管理界面跨主机访问

插件默认是只有本机的IP才能访问的，也就是127.0.0.1，这样我们就无法跨主机访问head 插件的管理界面，所以需要把它改成所有IP地址都能访问。该配置在head插件安装目录根目录下，文件名为Gruntfile.js

[root@qishengkun elasticsearch-head]# vim Gruntfile.js



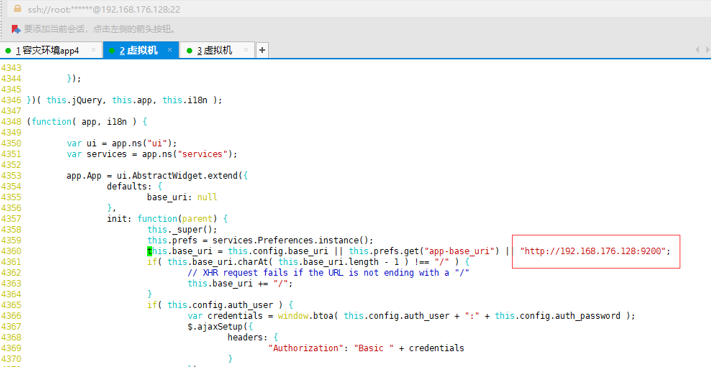
在该配置文件中connect-server-options下添加hostname: '0.0.0.0',这个配置，这样就不限制IP地址的访问了

1. 设置连接elasticsearch的地址

head插件默认是连接本机的elasticsearch的，如果你的elasticsearch和head插件是安装在同一台主机上，那么就不需要修改配置，如果不是安装在同一台主机的，就必须修改配置了，配置文件在head插件安装目录的\_site目录下，文件名为app.js

[root@qishengkun elasticsearch-head]# vim \_site/app.js

把this.base\_uri = this.config.base\_uri || this.prefs.get("app-base\_uri") || "http://localhost:9200";这行配置中的localhost改成你elasticsearch服务所在IP地址（如果安装在同一台主机就不需要修改）



1. elasticsearch配置允许跨域访问

[root@qishengkun ~]# cd /usr/local/elasticsearch/config/

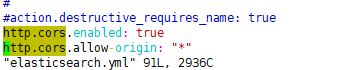
[root@qishengkun config]# vim elasticsearch.yml

# 是否支持跨域

http.cors.enabled: true

# \*表示支持所有域名

http.cors.allow-origin: "\*"



1. 在后台启动head插件服务

[root@qishengkun ~]# cd /usr/local/elasticsearch-head/node\_modules/grunt/bin

[root@qishengkun bin]# nohup grunt server &

1. 启动成功



# 安装kibana

1. 解压Kibana

[root@qishengkun ~]# tar zxf kibana-6.4.0-linux-x86\_64.tar.gz -C /usr/local/

1. 将其改名为kibana

[root@qishengkun ~]# cd /usr/local/

[root@qishengkun local]# mv kibana-6.4.0-linux-x86\_64/ kibana

1. 将其设置为开启自启动

[root@qishengkun ~]# vim /etc/rc.local

/usr/local/kibana/bin/kibana >/var/log/kibana.log 2>&1 &

1. 配置kibana

[root@qishengkun config]# vim /usr/local/kibana/config

server.port: 5601

server.host: "192.168.176.128"

elasticsearch.url: <http://192.168.176.128:9200>

1. 在后台启动kibana

[root@qishengkun config]# cd /usr/local/kibana/bin

[root@qishengkun bin]# nohup ./kibana &

1. 测试界面如下：

http://192.168.176.128:5601



1. 将nginx日志转成json

[root@qishengkun ~]# vim /usr/local/nginx/conf/nginx.conf

http {

include mime.types;

default\_type application/octet-stream;

charset utf-8;

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

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

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

log\_format log\_json '{ "@timestamp": "$time\_local", '

'"remote\_addr": "$remote\_addr", '

'"referer": "$http\_referer", '

'"request": "$request", '

'"status": $status, '

'"bytes": $body\_bytes\_sent, '

'"agent": "$http\_user\_agent", '

'"x\_forwarded": "$http\_x\_forwarded\_for", '

'"up\_addr": "$upstream\_addr",'

'"up\_host": "$upstream\_http\_host",'

'"up\_resp\_time": "$upstream\_response\_time",'

'"request\_time": "$request\_time"'

' }';

access\_log logs/access.log log\_json;

(省略内容)

}

1. 重启nginx

[root@qishengkun ~]# /usr/local/nginx/sbin/nginx -s reload

# 安装Logstash

1. 解压logstash

[root@qishengkun ~]# tar zxf logstash-6.4.0.tar.gz -C /usr/src/

1. 新建一个/usr/local/logstash-6.4.0/config/syslog.conf

[root@qishengkun ~]# cd /usr/local/logstash-6.4.0/config/

1. 在配置文件里输入以下内容

[root@qishengkun config]# vim syslog.conf

input { # 定义日志源

syslog {

type => "system-syslog" # 定义类型

port => 10514 # 定义监听端口

}

}

output { # 定义日志输出

stdout {

codec => rubydebug # 将日志输出到当前的终端上显示

}

}

1. 检测配置文件是否有错

[root@qishengkun config]# cd ../bin

[root@qishengkun bin]# ./logstash --path.settings /usr/local/logstash-6.4.0/config/ -f /usr/local/logstash-6.4.0/config/syslog.conf --config.test\_and\_exit

--path.settings 用于指定logstash的配置文件所在的目录

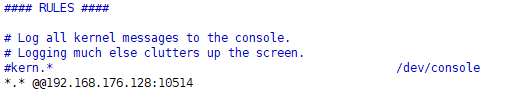
-f 指定需要被检测的配置文件的路径

--config.test\_and\_exit 指定检测完之后就退出，不然就会直接启动了

1. 配置kibana服务器的ip以及配置的监听端口

[root@qishengkun ~]# vim /etc/rsyslog.conf

\*.\* @@192.168.176.128:10514



1. 重启rsyslog服务

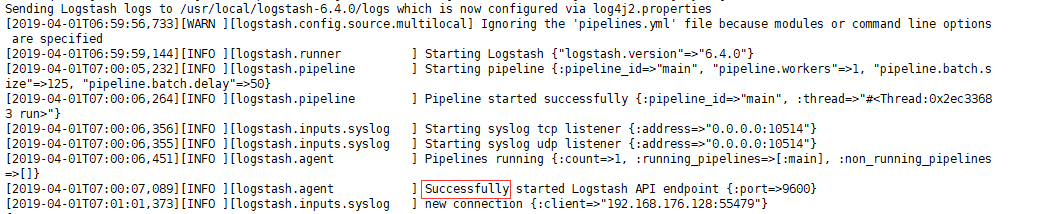
[root@qishengkun ~]# /etc/init.d/rsyslog restart

1. 指定配置文件，启动logstash

[root@qishengkun bin]# cd /usr/local/logstash-6.4.0/bin/

[root@qishengkun bin]# ./logstash --path.settings /usr/local/logstash-6.4.0/config/ -f ../config/syslog.conf

1. 启动成功



…

1. 更改logstash 让收集的日志信息输出到es服务器中

[root@qishengkun config]# vim /usr/local/logstash-6.4.0/config/syslog.conf

input { # 定义日志源

syslog {

type => "system-syslog" # 定义类型

port => 10514 # 定义监听端口

}

}

output {

elasticsearch {

hosts => ["192.168.176.128:9200"] # 定义es服务器的ip

index => "system-syslog-%{+YYYY.MM}" # 定义索引

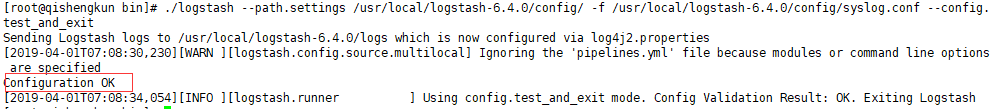
}

}

1. 检查配置文件有没有错

[root@qishengkun config]# cd ../bin

[root@qishengkun bin]# ./logstash --path.settings /usr/local/logstash-6.4.0/config/ -f /usr/local/logstash-6.4.0/config/syslog.conf --config.test\_and\_exit

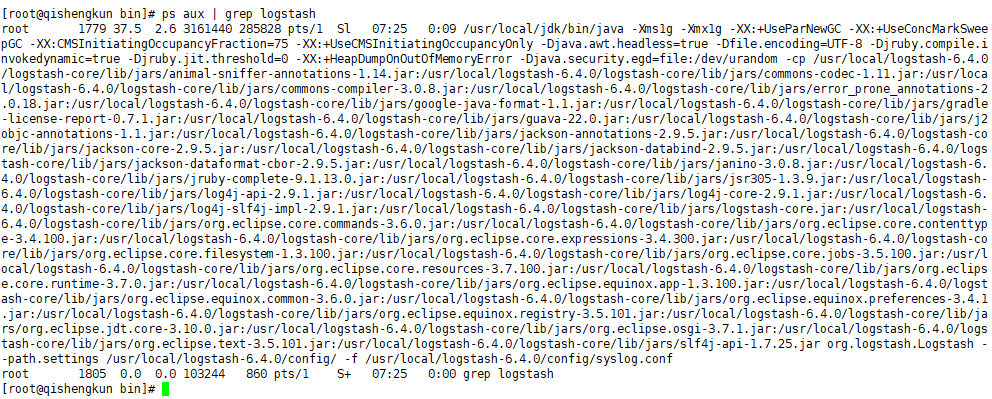


1. 启动logstash

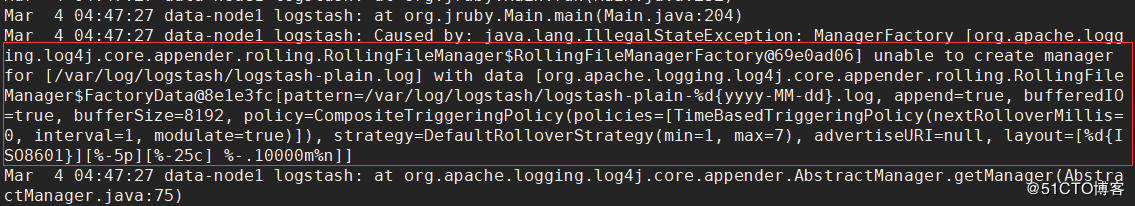
[root@qishengkun bin]# nohup ./logstash --path.settings /usr/local/logstash-6.4.0/config/ -f /usr/local/logstash-6.4.0/config/syslog.conf &

1. 检查启动情况

[root@qishengkun bin]# ps aux | grep logstash



1. 错误解决

我这里启动logstash后，进程是正常存在的，但是9600以及10514端口却没有被监听。于是查看logstash的日志看看有没有错误信息的输出，但是发现没有记录日志信息，那就只能转而去查看messages的日志，发现错误信息如下：  


这是因为权限不够，既然是权限不够，那就设置权限即可：

[root@qishengkun logs]# chown logstash /usr/local/logstash-6.4.0/logs/logstash-slowlog-plain.log

如果还是不可以的话

[root@qishengkun ~]# chown -R logstash /usr/local/logstash-6.4.0/

没有账号的话创建logstash

[root@qishengkun logs]# useradd -s /sbin/nologin -M logstash

再次启动就可以了

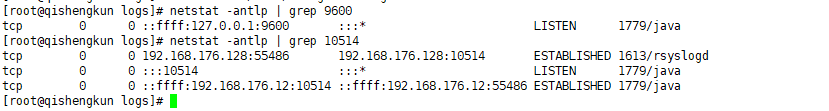
[root@qishengkun bin]# nohup ./logstash --path.settings /usr/local/logstash-6.4.0/config/ -f /usr/local/logstash-6.4.0/config/syslog.conf &

1. 查看端口号

[root@qishengkun logs]# netstat -antlp | grep 9600

[root@qishengkun logs]# netstat -antlp | grep 10514

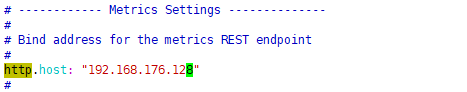
（有监听就可以）



1. 但是可以看到，logstash的监听ip是127.0.0.1这个本地ip，本地ip无法远程通信，所以需要修改一下配置文件，配置一下监听的ip：

[root@qishengkun config]# vim /usr/local/logstash-6.4.0/config/logstash.yml

http.host: "192.168.176.128"

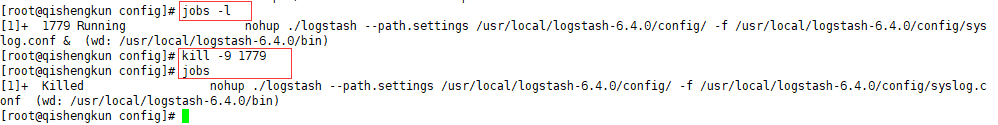


1. 杀死nohup进程

[root@qishengkun config]# jobs -l

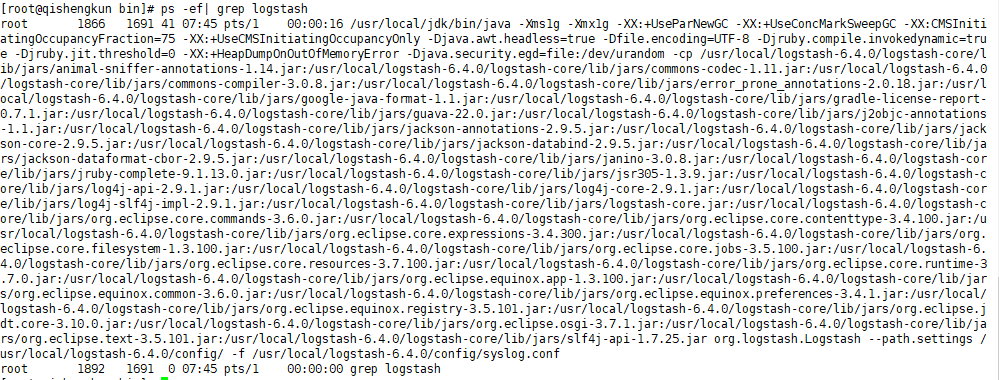
[root@qishengkun config]# kill -9 PID号

[root@qishengkun config]# jobs



1. 启动服务

[root@qishengkun bin]# nohup ./logstash --path.settings /usr/local/logstash-6.4.0/config/ -f /usr/local/logstash-6.4.0/config/syslog.conf &

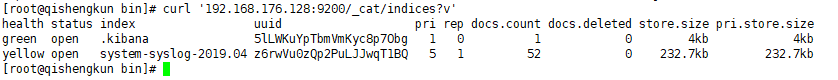


# 配置ELK

1. kibana上查看日志（登陆kibana服务器）

完成了logstash服务器的搭建之后，回到kibana服务器上查看日志，执行以下命令可以获取索引信息

[root@qishengkun ~]# curl '192.168.176.128:9200/\_cat/indices?v'

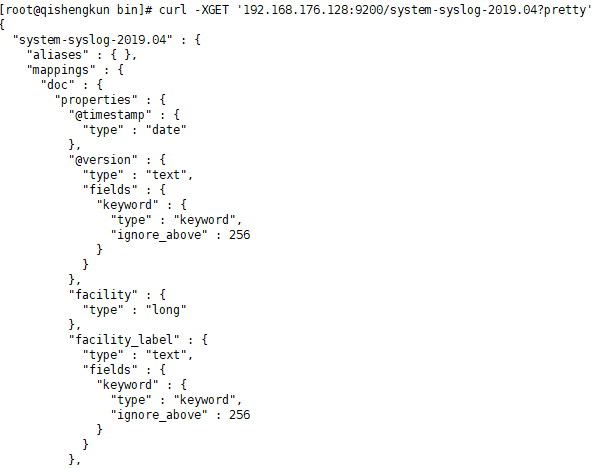


如上，可以看到，在logstash配置文件中定义的system-syslog索引成功获取到了，证明配置没问题，logstash与es通信正常。

1. 获取指定索引详细信息：（此命令需要修改ip和后面的时间）

[root@qishengkun bin]#

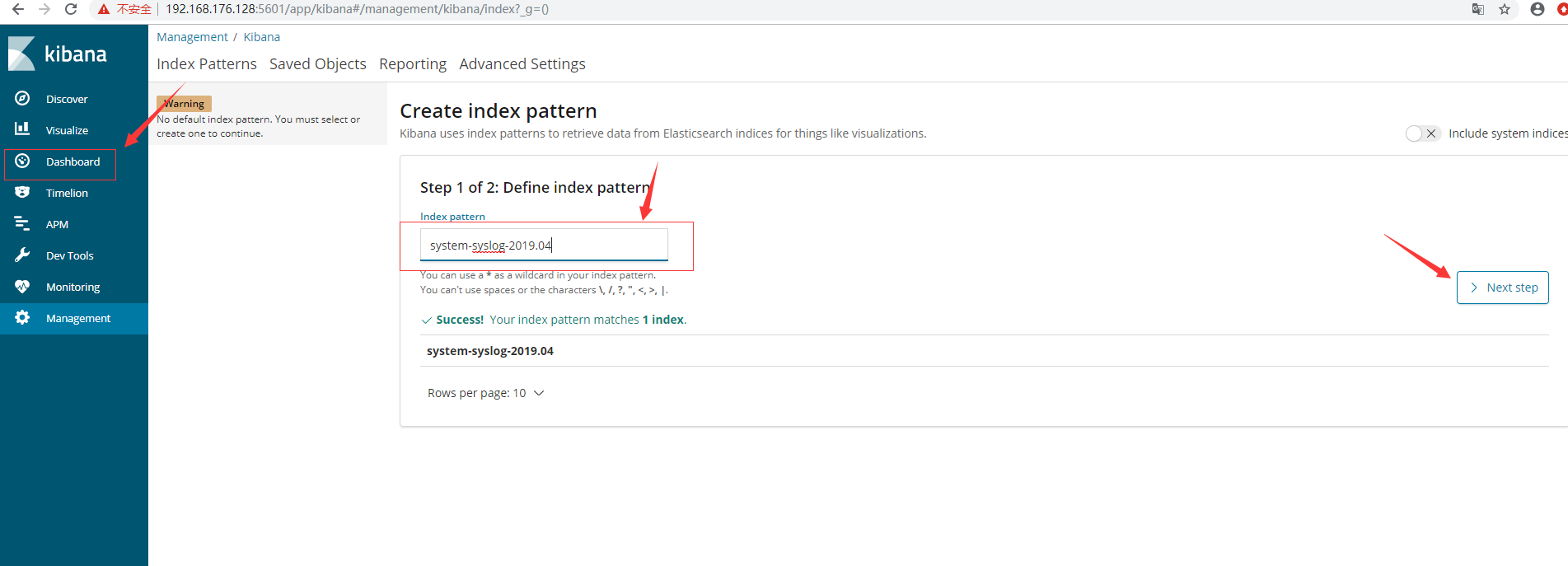
curl -XGET '192.168.176.128:9200/system-syslog-2019.04?pretty'



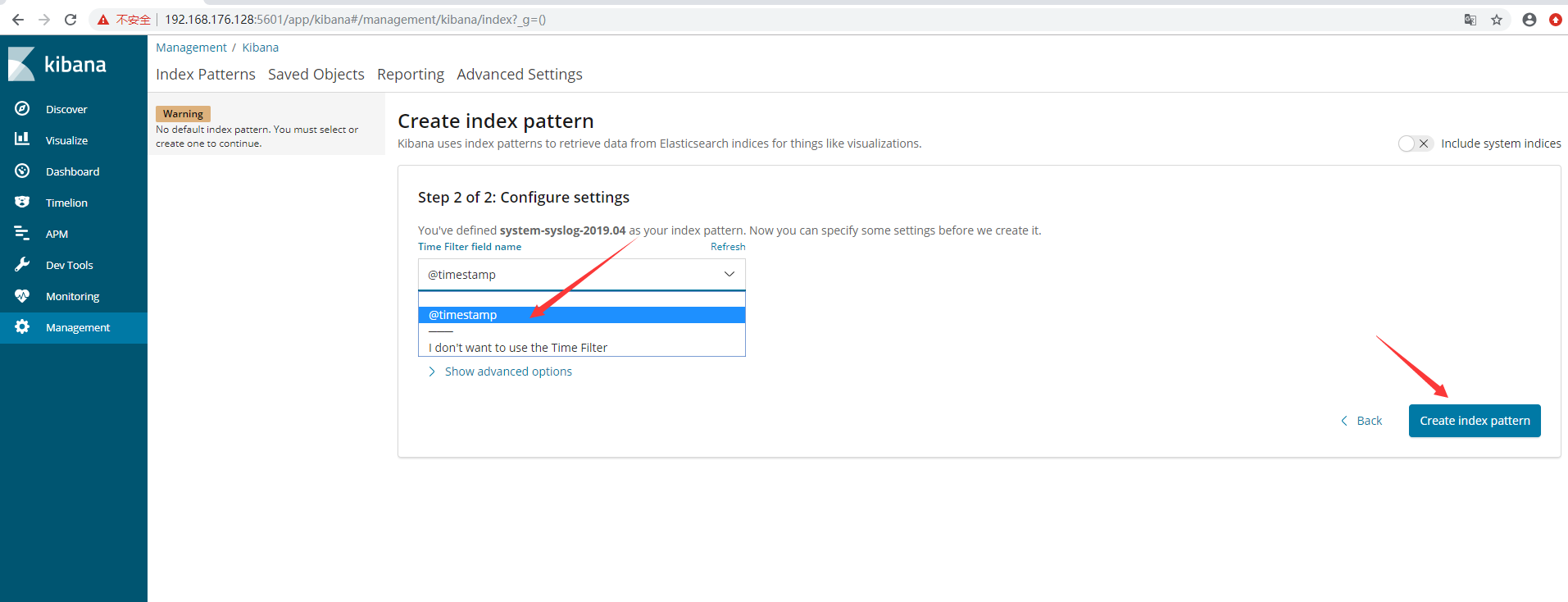
1. 如果如何需要删除索引的话，使用一些命令可以删除指定索引

[root@qishengkun bin]# curl -XDELETE '192.168.176.128:9200/system-syslog-2019.04'

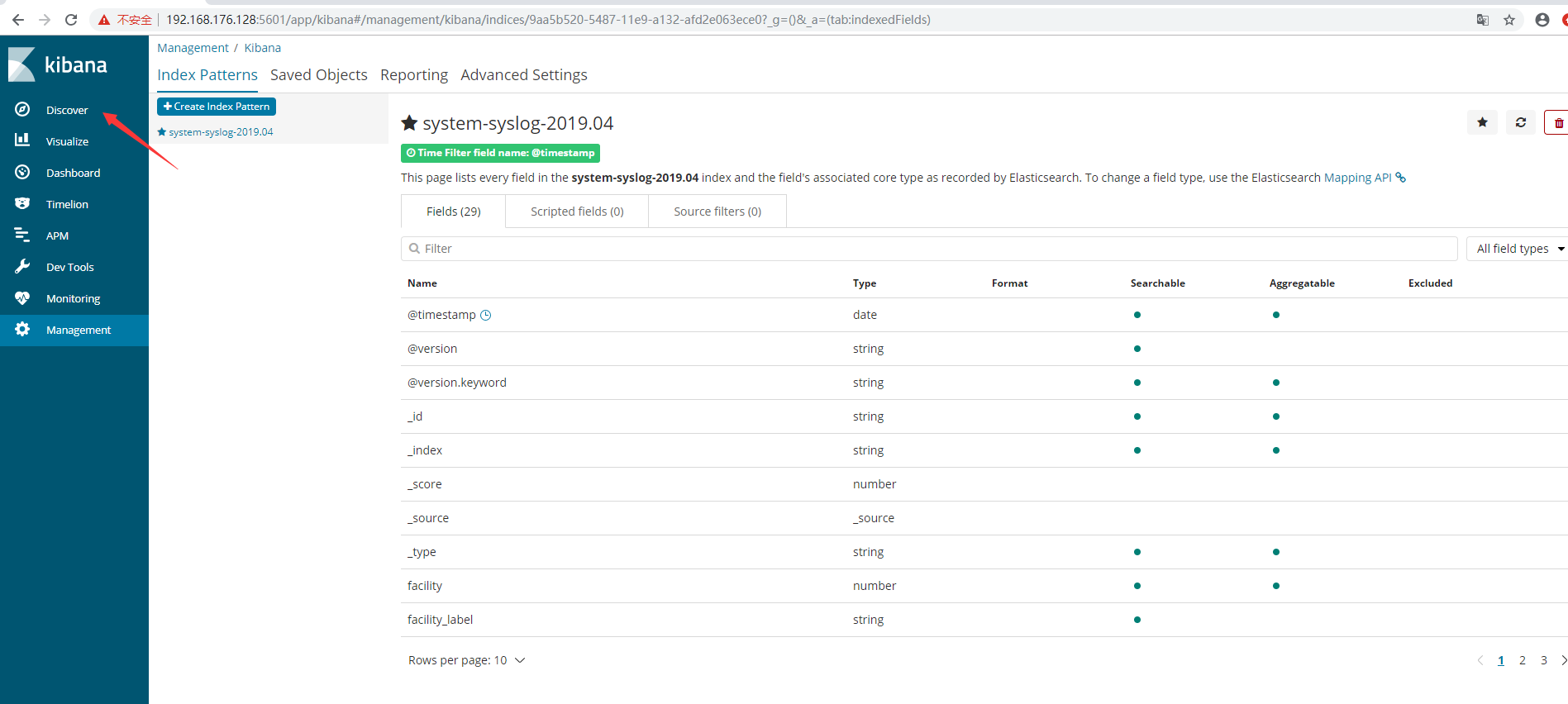
1. es与logstash能够正常通信后就可以去配置kibana了，浏览器访问192.168.176.128:5601，到kibana页面上配置索引：



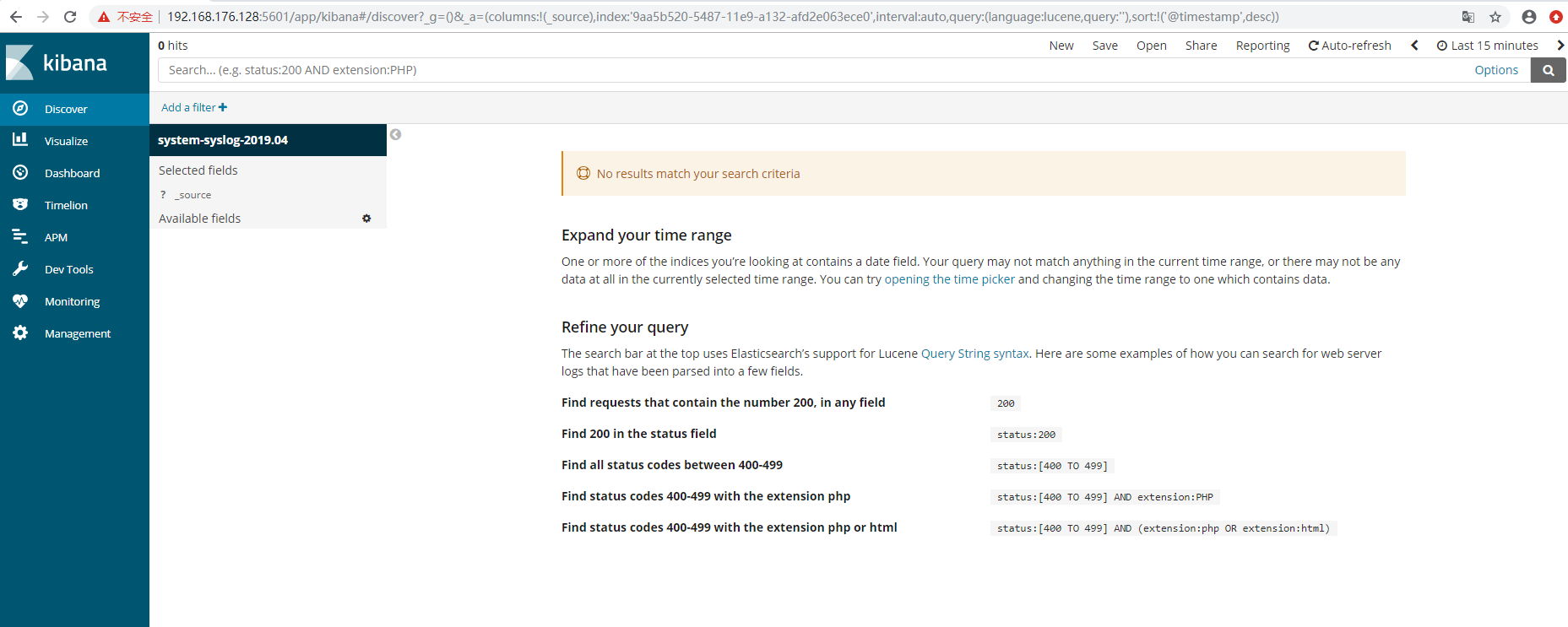
选择timestamp



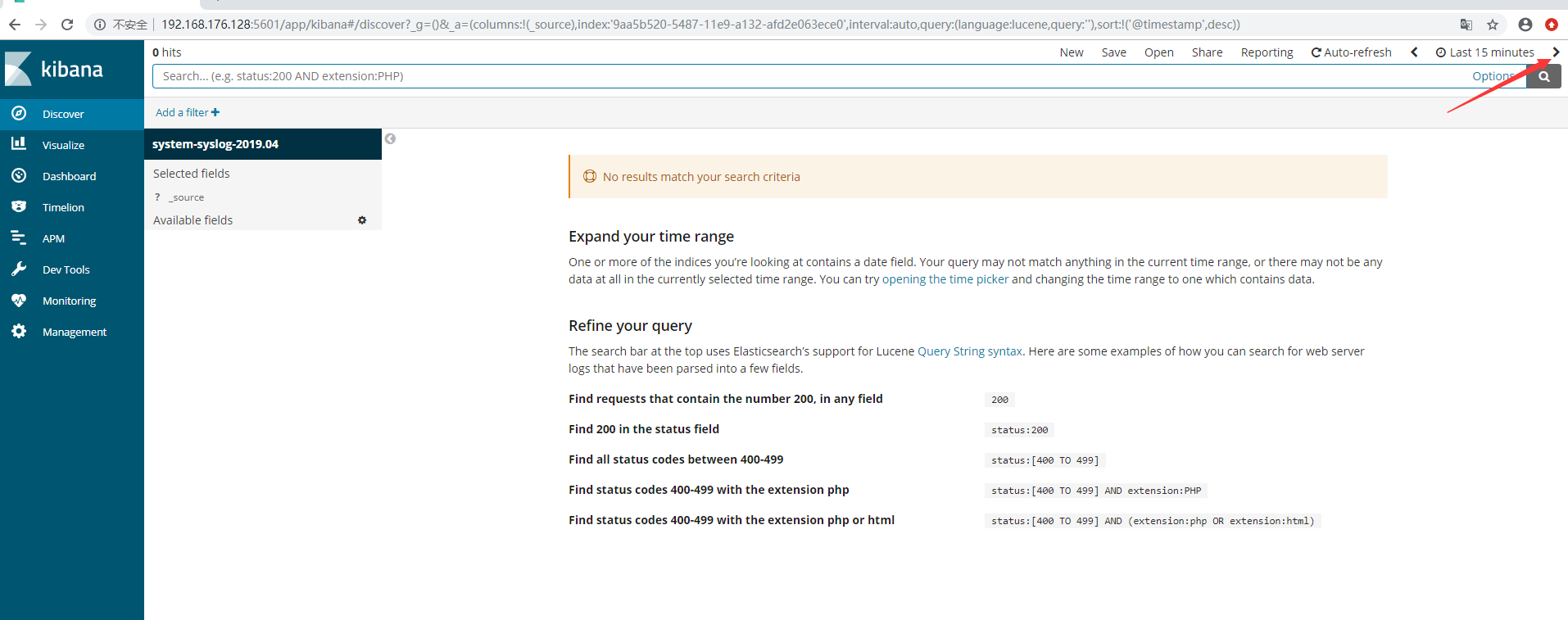
配置成功后点击 “Discover” ：



进入 “Discover” 页面后如果出现以下提示，则是代表无法查找到日志信息：



这种情况一般是时间的问题，点击右上角切换成查看当天的日志信息：



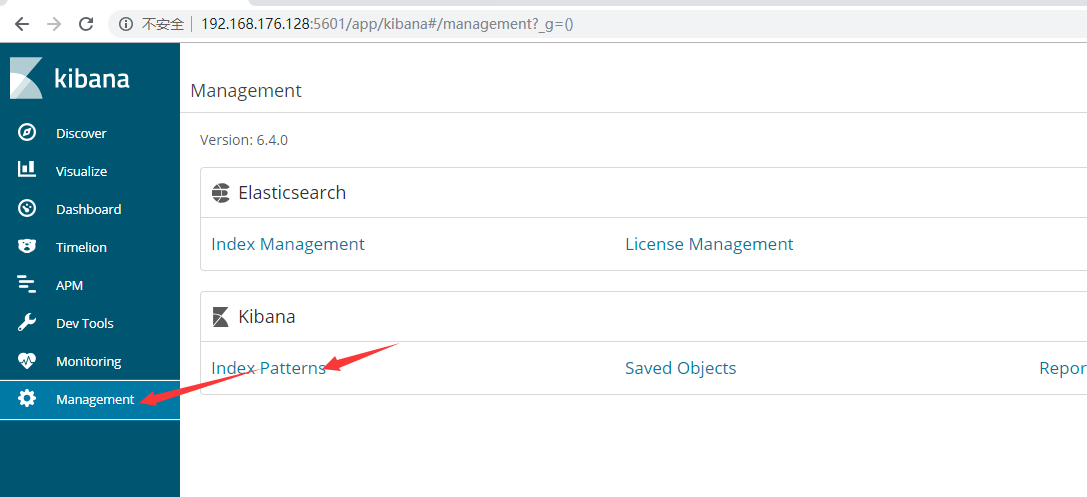
如果还是不行的话，就换几个时间试试，换了几个时间都不行的话，就在浏览器中直接访问es服务器看看是否有反馈出信息：

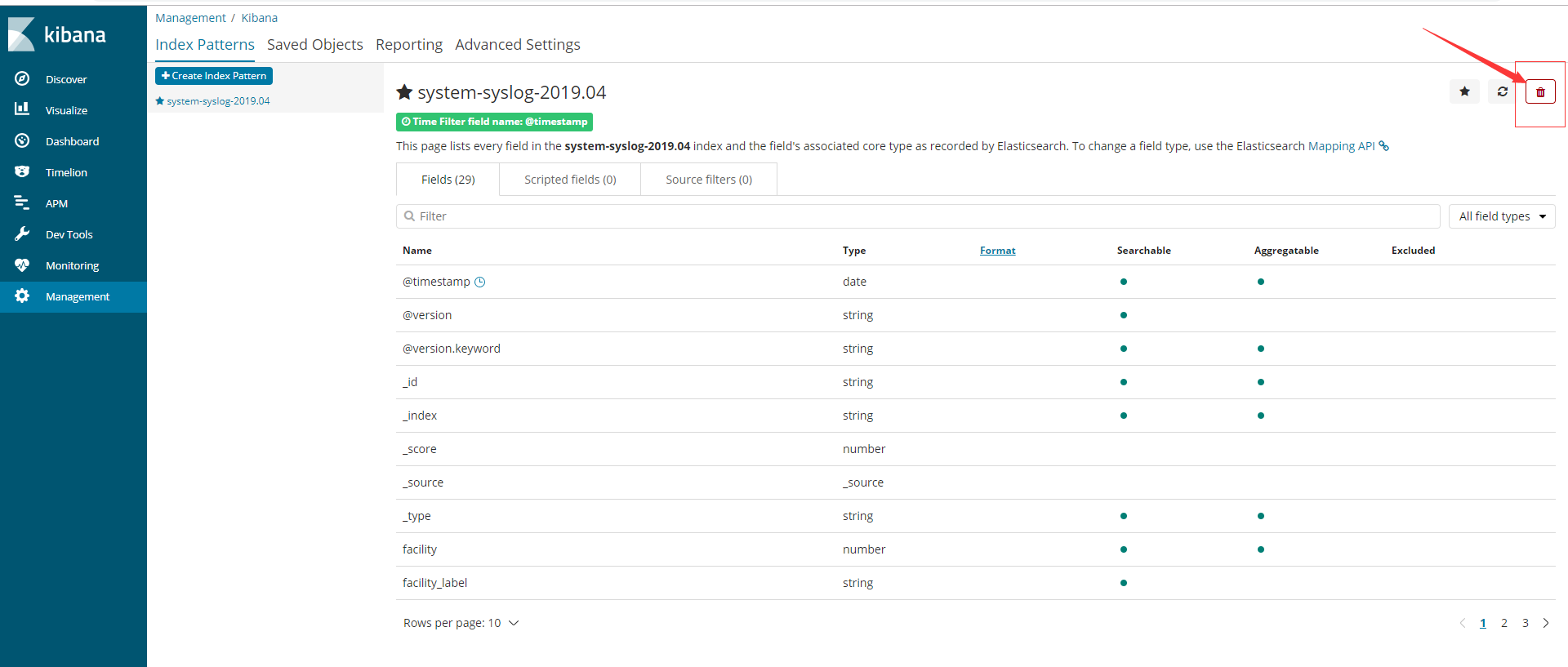
http://192.168.176.128:9200/system-syslog-2019.04/\_search?q=\*

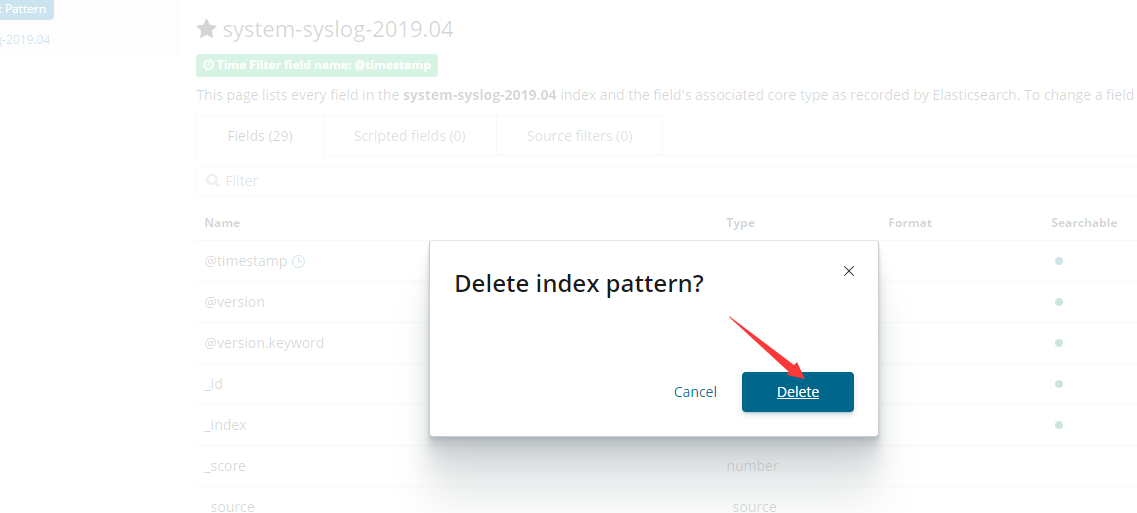
如下，这是正常返回信息的情况，如果有问题的话是会返回error的：



如果es服务器正常返回信息，但是 “Discover” 页面却依旧显示无法查找到日志信息的话，就使用另一种方式，进入设置删除掉索引：

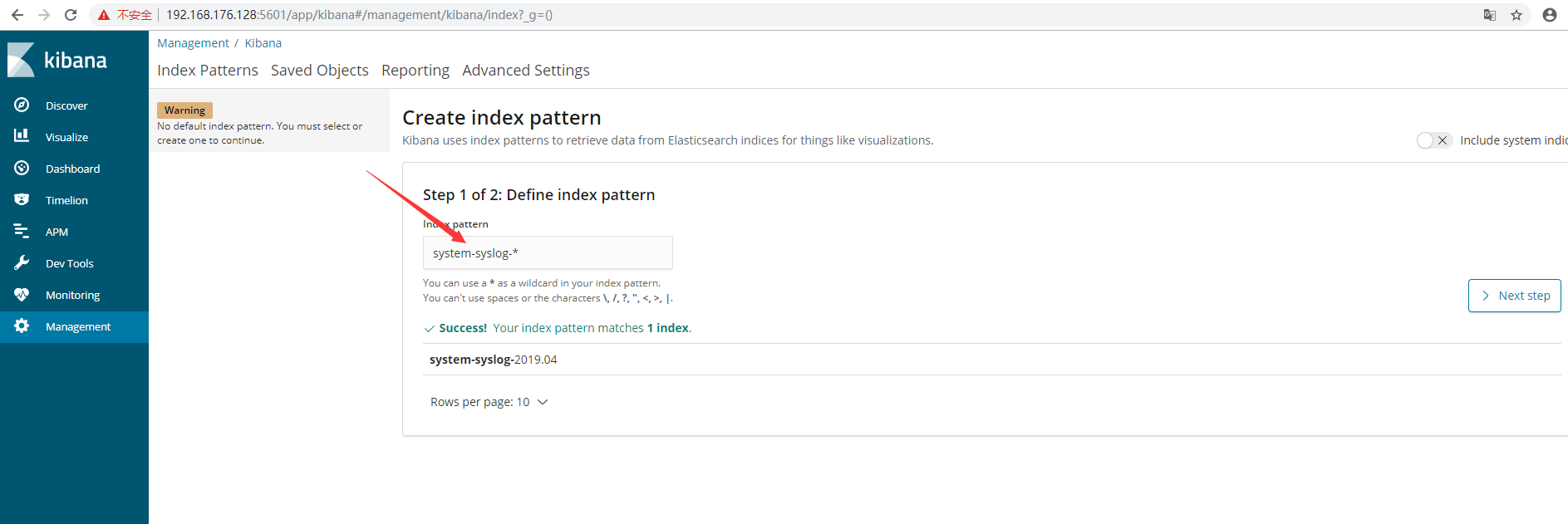


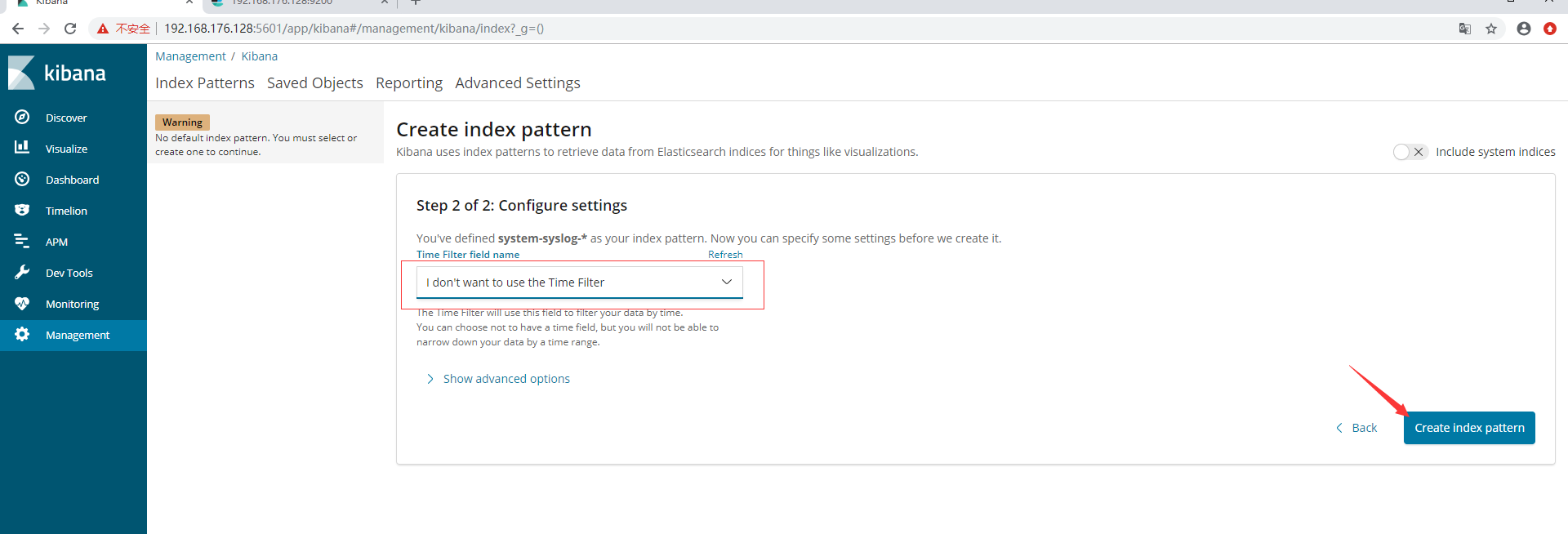




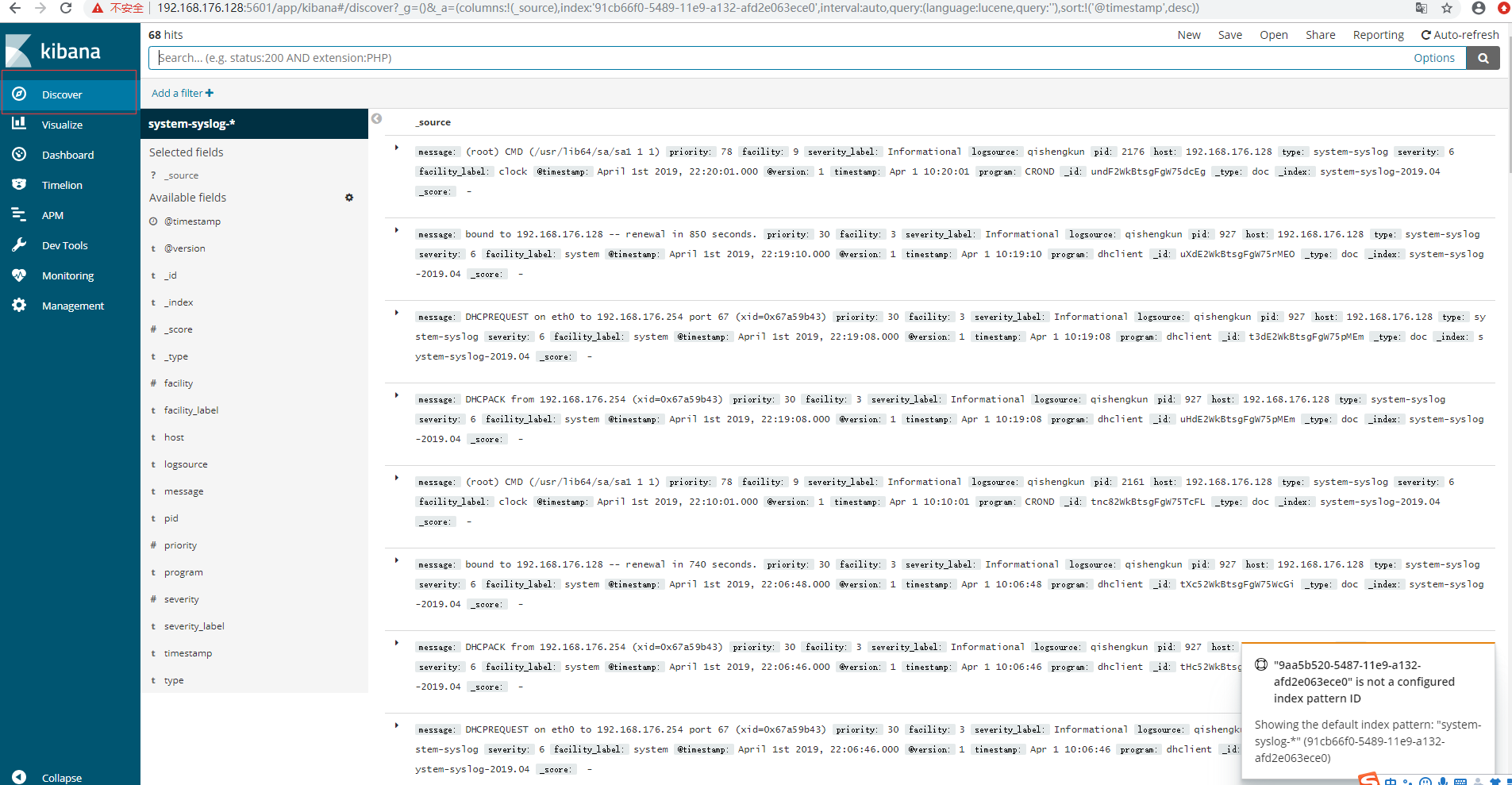
重新添加索引，但是这次不要选择 @timestampe 了：

system-syslog-\*





但是这种方式只能看到数据，没有可视化的柱状图：



其实这里显示的日志数据就是 /var/log/messages 文件里的数据，因为logstash里配置的就是收集messages 文件里的数据。

以上这就是如何使用logstash收集系统日志，输出到es服务器上，并在kibana的页面上进行查看。

logstash收集nginx日志

和收集syslog一样，首先需要编辑配置文件，这一步在logstash服务器上完成：

[root@qishengkun ~]# /usr/local/logstash-6.4.0/config

[root@qishengkun config]# vim nginx.conf

input {

file { # 指定一个文件作为输入源

path => "/tmp/elk\_access.log" # 指定文件的路径

start\_position => "beginning" # 指定何时开始收集

type => "nginx" # 定义日志类型，可自定义

}

}

filter { # 配置过滤器

grok {

match => { "message" => "%{IPORHOST:http\_host} %{IPORHOST:clientip} - %{USERNAME:remote\_user} \[%{HTTPDATE:timestamp}\] \"(?:%{WORD:http\_verb} %{NOTSPACE:http\_request}(?: HTTP/%{NUMBER:http\_version})?|%{DATA:raw\_http\_request})\" %{NUMBER:response} (?:%{NUMBER:bytes\_read}|-) %{QS:referrer} %{QS:agent} %{QS:xforwardedfor} %{NUMBER:request\_time:float}"} # 定义日志的输出格式

}

geoip {

source => "clientip"

}

}

output {

stdout { codec => rubydebug }

elasticsearch {

hosts => ["192.168.176.128:9200"]

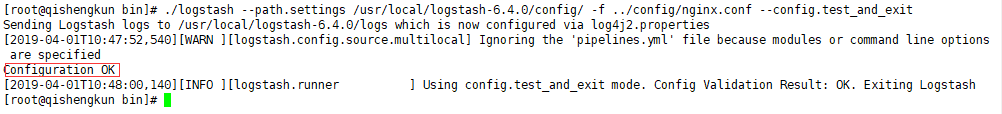
index => "nginx-test-%{+YYYY.MM.dd}"

}

同样的编辑完配置文件之后，还需要检测配置文件是否有错

[root@qishengkun bin]# cd /usr/local/logstash-6.4.0/bin/

[root@qishengkun bin]# ./logstash --path.settings /usr/local/logstash-6.4.0/config/ -f ../config/nginx.conf --config.test\_and\_exit



修改nginx配置文件

配置nginx的主配置文件，因为需要配置日志格式，在 log\_format combined\_realip

[root@qishengkun ~]# vim /usr/local/nginx/conf/nginx.conf

worker\_processes 1;

events {

worker\_connections 1024;

}

http {

log\_format main2 '$http\_host $remote\_addr - $remote\_user [$time\_local] "$request" '

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

'"$http\_user\_agent" "$upstream\_addr" $request\_time';

include mime.types;

default\_type application/octet-stream;

sendfile on;

keepalive\_timeout 65;

server {

listen 80;

server\_name elk.test.com;

location / {

proxy\_pass http://192.168.176.128:5601;

proxy\_set\_header Host $host;

proxy\_set\_header X-Real-IP $remote\_addr;

proxy\_set\_header X-Forwarded-For $proxy\_add\_x\_forwarded\_for;

}

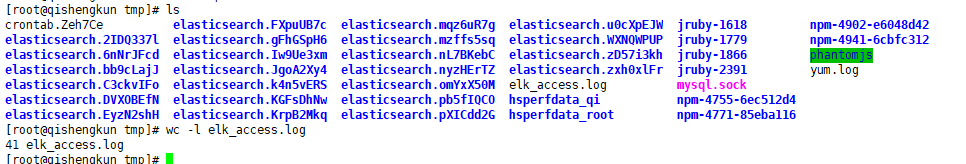
access\_log /tmp/elk\_access.log main2;

}

}

访问成功后，查看生成的日志文件：

[root@qishengkun tmp]# wc -l /tmp/elk\_access.log

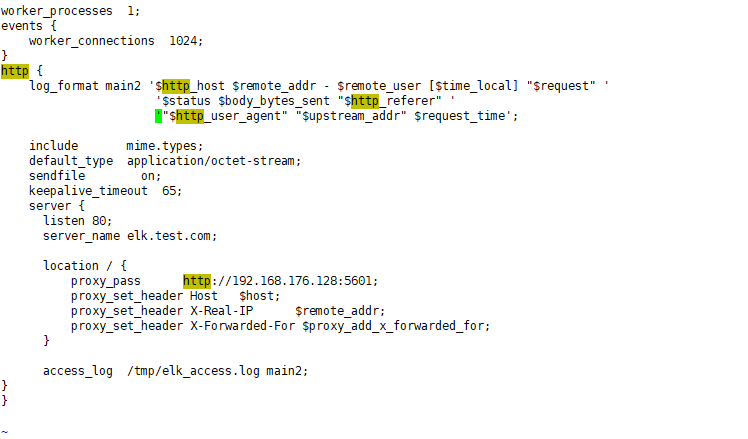


重新启动logstash

[root@qishengkun bin]# cd /usr/local/logstash-6.4.0/bin/

[root@qishengkun bin]# nohup ./logstash --path.settings /usr/local/logstash-6.4.0/config/ &

重启完成后，在es服务器上检查是否有nginx-test开头的索引生成



测试nginx配置文件

[root@qishengkun conf]# /usr/local/nginx/sbin/nginx –t

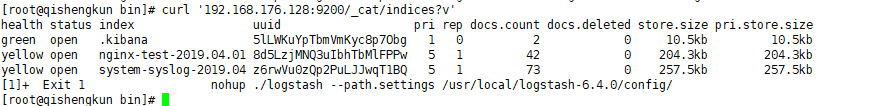


重启nginx

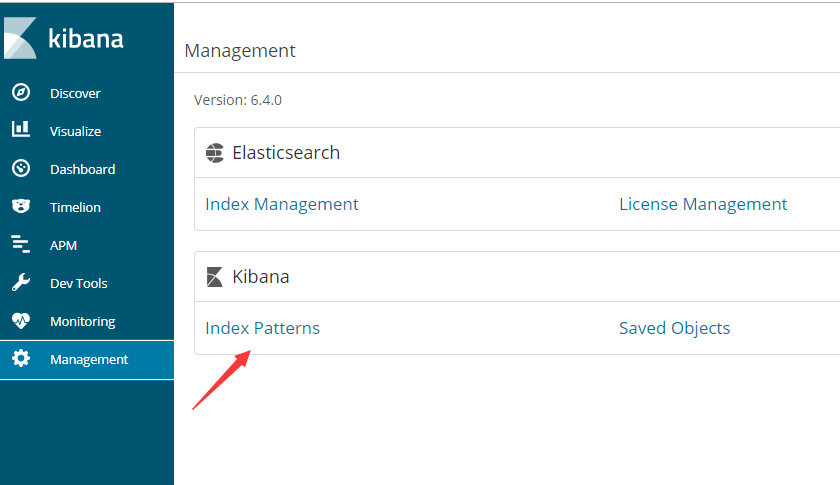
[root@qishengkun conf]# /usr/local/nginx/sbin/nginx -s reload

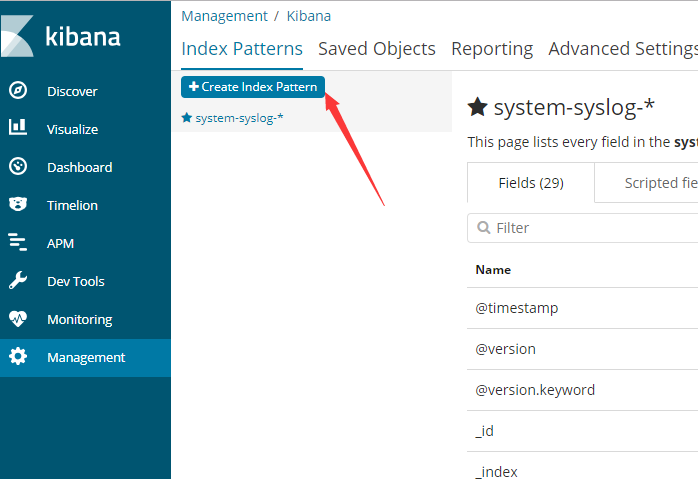
重启完成后，在es服务器上检查是否有nginx-test开头的索引生成：

[root@qishengkun bin]# curl '192.168.176.128:9200/\_cat/indices?v'

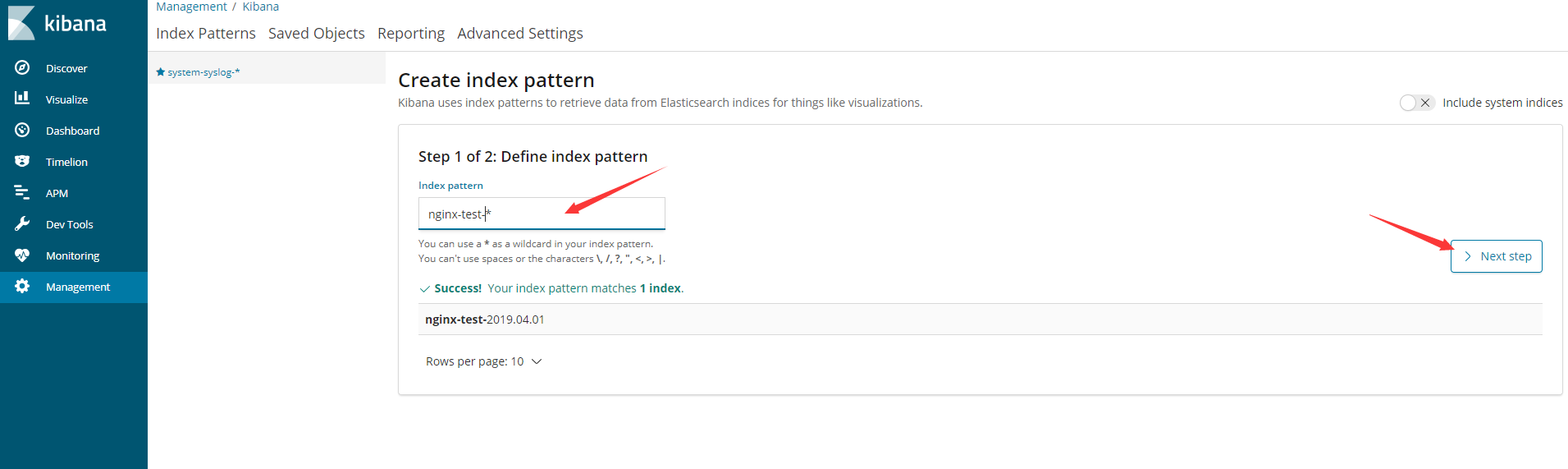


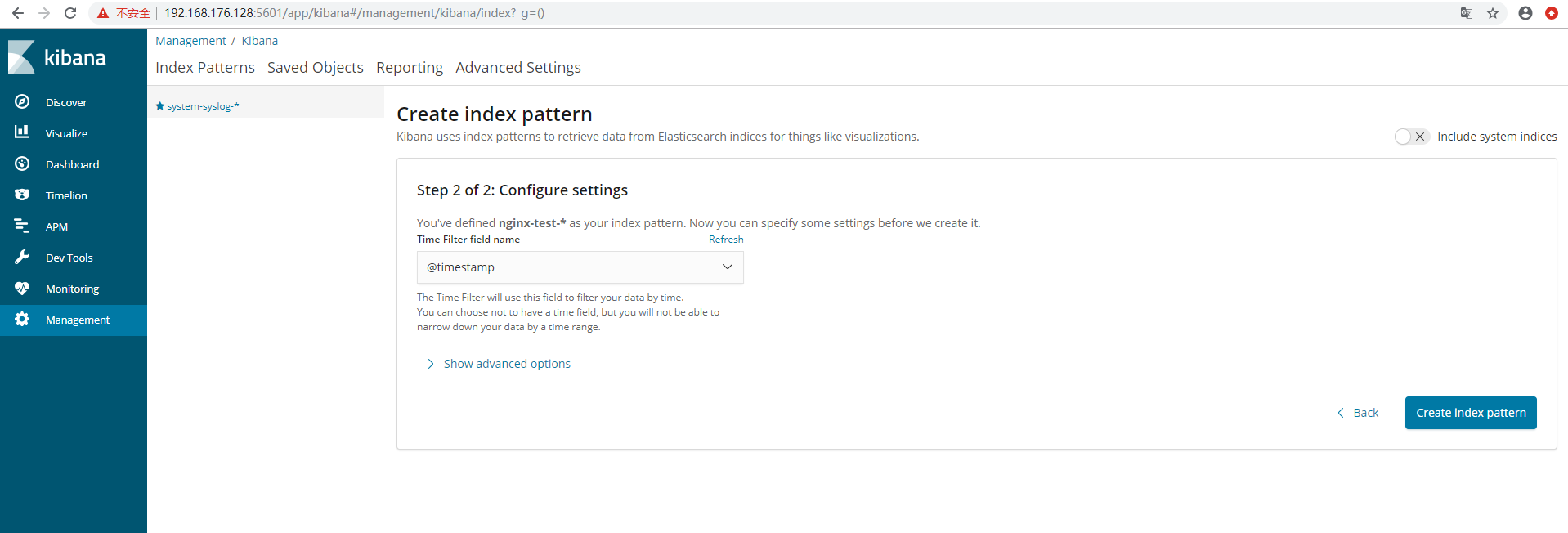
可以看到，nginx-test索引已经生成了，那么这时就可以到kibana上配置该索引：



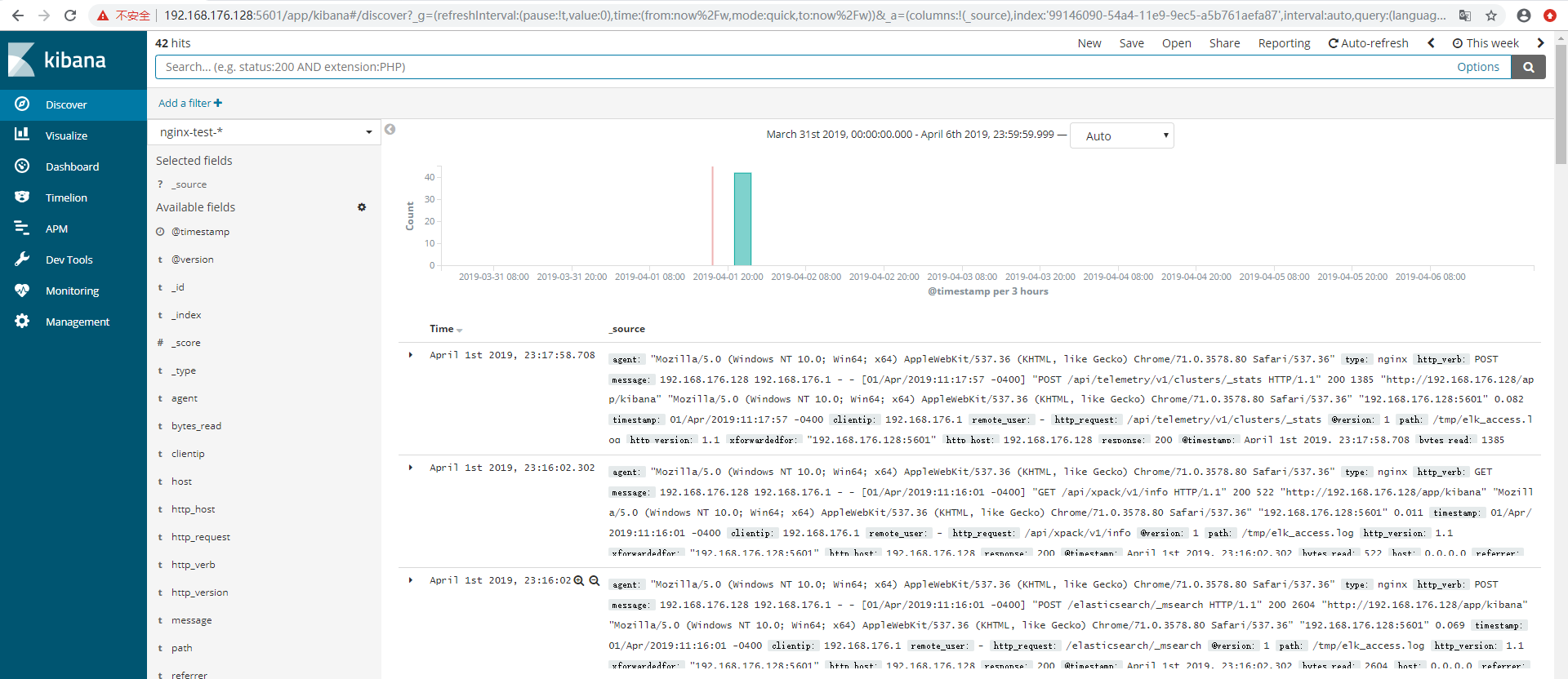


nginx-test-\*





配置完成之后就可以在 “Discover” 里进行查看nginx的访问日志数据了：



# 搭建filebeat

**agent端搭建占用少量cpu和内存**

1. 解缩

[root@qishengkun ~]# tar zxf filebeat-6.4.0-linux-x86\_64.tar.gz -C /usr/local

1. 重命名

[root@qishengkun ~]# cd /usr/local/

[root@qishengkun local]# mv filebeat-6.4.0-linux-x86\_64/ filebeat

1. 修改配置文件

[root@qishengkun local]# cd filebeat/

[root@qishengkun filebeat]# vim filebeat.yml

#output.console: 把这两句注释掉

# enable: true

# 把这两句的注释去掉

output.elasticsearch:

# Array of hosts to connect to.

hosts: ["192.168.77.128:9200"] # 并配置es服务器的ip地址

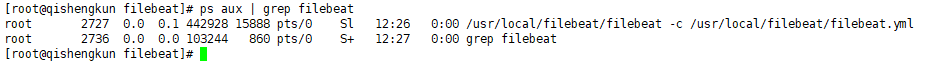
~

1. 看看是否有在终端中打印日志数据，有打印则代表filebeat能够正常收集日志数据： [root@qishengkun filebeat]#

nohup /usr/local/filebeat/filebeat -c /usr/local/filebeat/filebeat.yml &

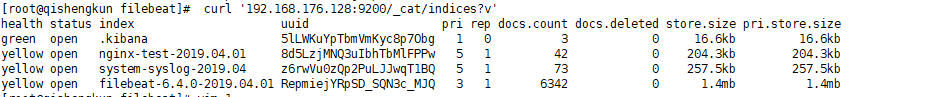
1. 检查进程

[root@qishengkun filebeat]# ps aux | grep filebeat

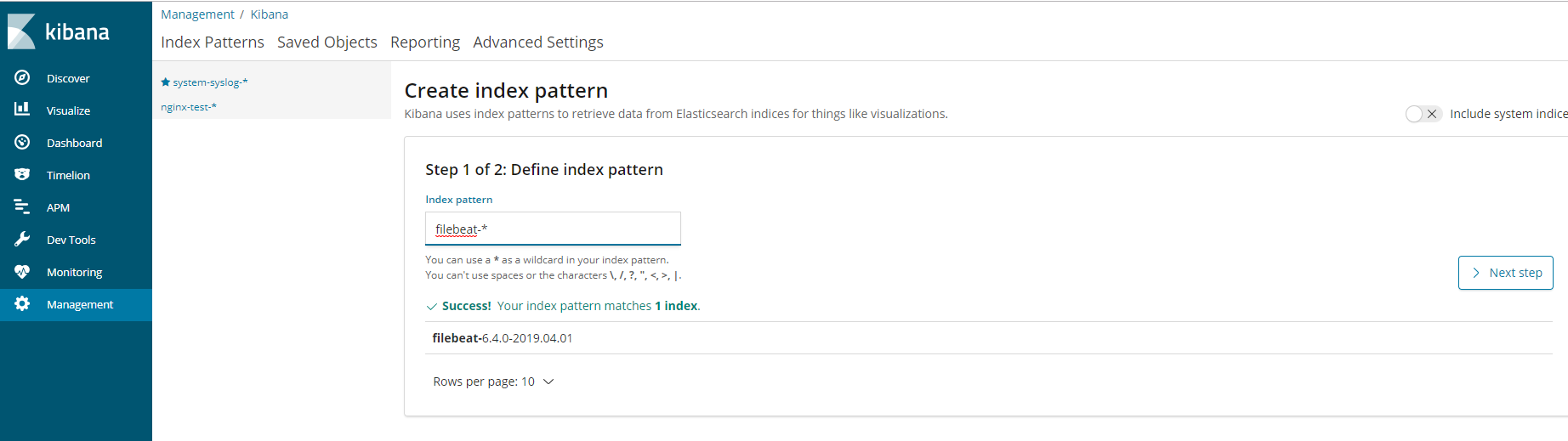


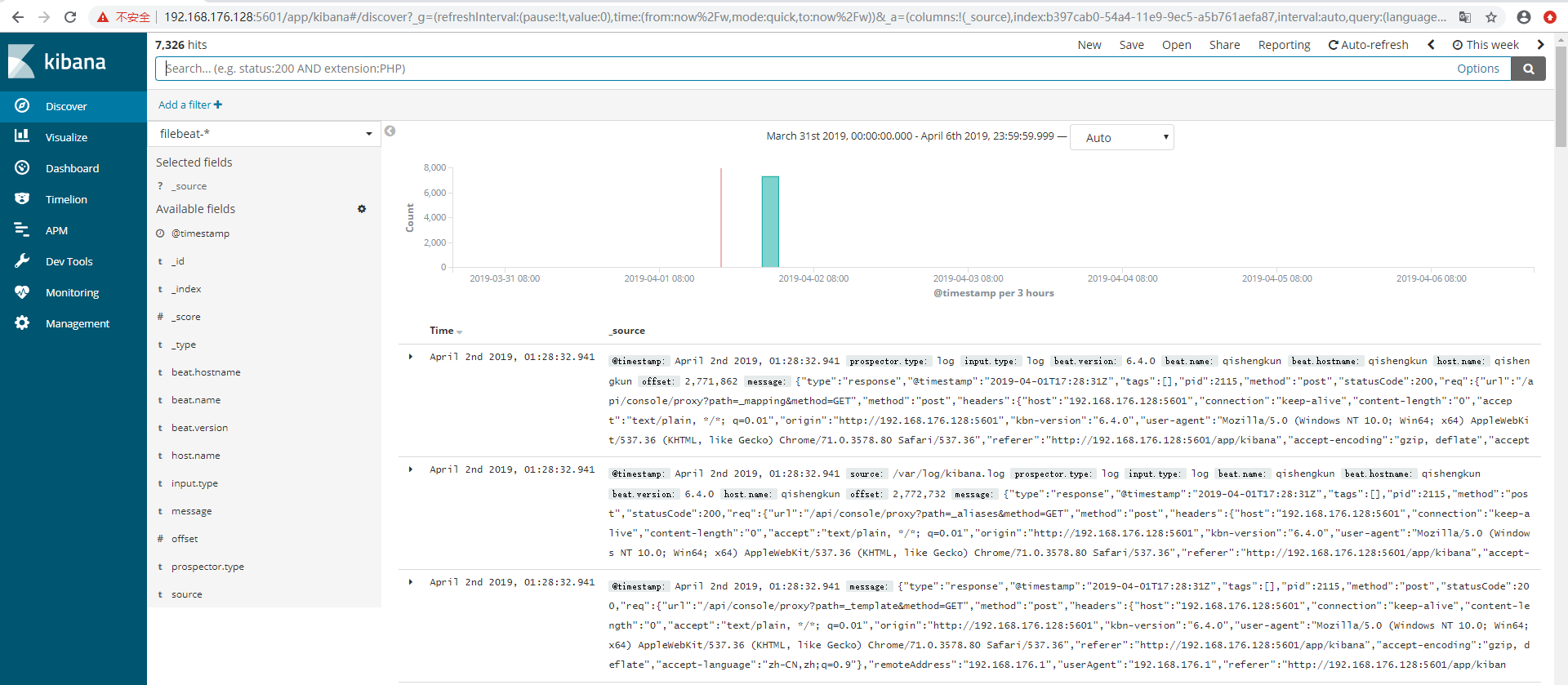
1. 启动成功后，到es服务器上查看索引，可以看到新增了一个以filebeat-6.0.0开头的索引，这就代表filesbeat和es能够正常通信了：

[root@qishengkun filebeat]# curl '192.168.176.128:9200/\_cat/indices?v'



1. es服务器能够正常获取到索引后，就可以到kibana上配置这个索引了：



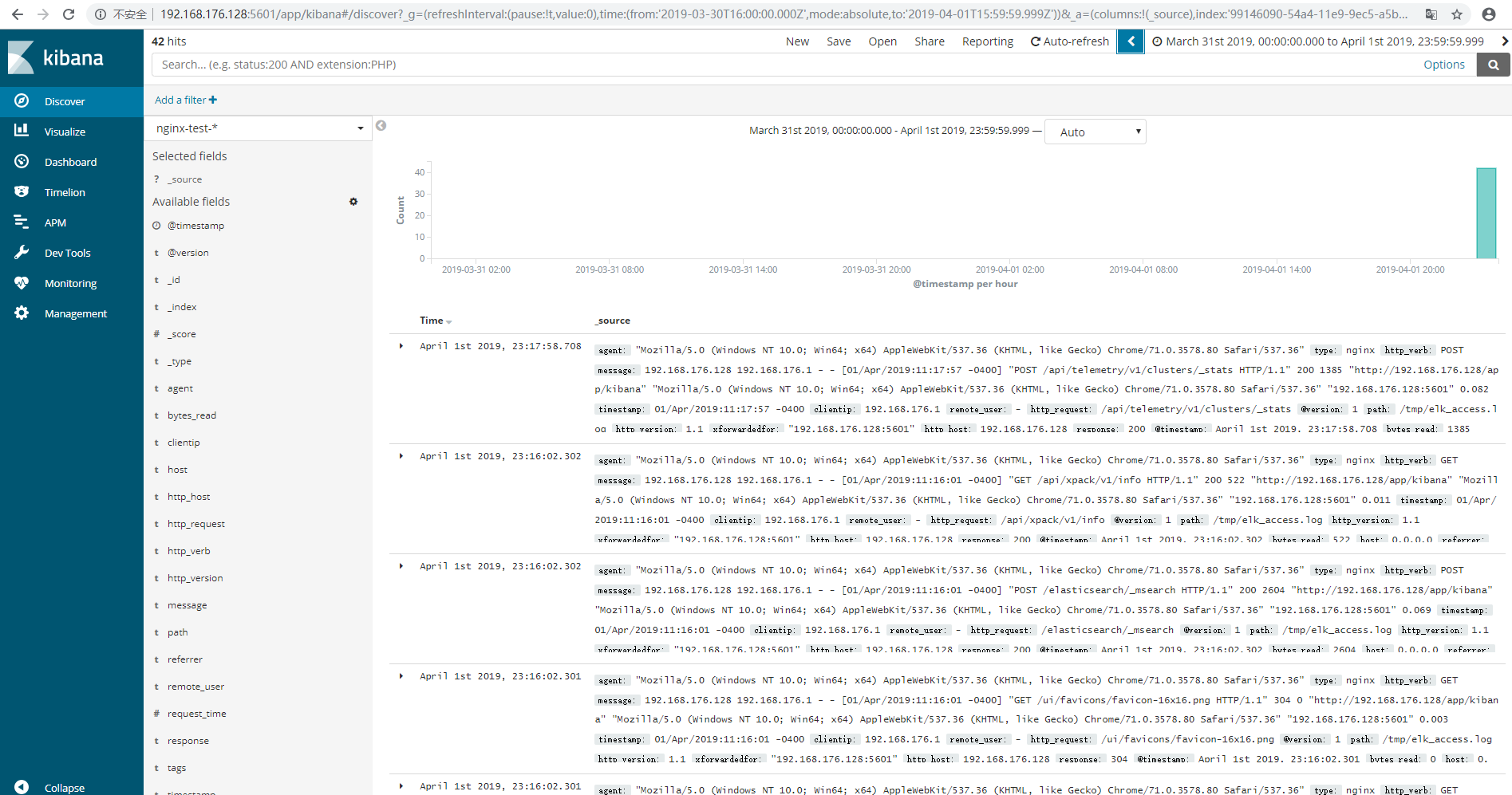


以上这就是如何使用filebeat进行日志的数据收集，可以看到配置起来比logstash要简单，而且占用资源还少。

# 错误timestamp

1. 可能造成的原因是kibana先启动了elasticsearch 后启动导致此原因

# Kibana介绍



## Visualize

可视化能使你创造你的Elasticsearch指标数据的可视化。然后你可以建立仪表板显示相关的可视化。Kibana的可视化是基于Elasticsearch查询。通过一系列的Elasticsearch聚合提取和处理您的数据，您可以创建图表显示你需要知道的关于趋势，峰值和骤降。您可以从搜索保存的搜索中创建可视化或从一个新的搜索查询开始。

## Dashboard

一个仪表板显示Kibana保存的一系列可视化。你可以根据需要安排和调整可视化，并保存仪表盘，可以被加载和共享。

## Monitoring

从图中可以发现，默认Kibana是没有该选项的。其实，Monitoring是由X-Pack集成提供的。  
该X-pack监控组件使您可以通过Kibana轻松地监控ElasticSearch。您可以实时查看集群的健康和性能，以及分析过去的集群、索引和节点度量。此外，您可以监视Kibana本身性能。当你安装X-pack在群集上，监控代理运行在每个节点上收集和指数指标从Elasticsearch。安装在X-pack在Kibana上，您可以查看通过一套专门的仪表板监控数据。

## Graph

X-Pack图的能力使你发现一个Elasticsearch索引项是如何相关联的。你可以探索索引条款之间的连接，看看哪些连接是最有意义的。从欺诈检测到推荐引擎，对各种应用中这都是有用的，例如，图的探索可以帮助你发现网站上黑客的目标的漏洞，所以你可以硬化你的网站。或者，您可以为您的电子商务客户提供基于图表的个性化推荐。X-pack提供简单，但功能强大的图形开发API，和Kibana交互式图形可视化工具。使用X-pack图有工作与开销与现有Elasticsearch指标你不需要任何额外的数据存储的特征。

## Timelion

Timelion是一个时间序列数据的可视化，可以结合在一个单一的可视化完全独立的数据源。它是由一个简单的表达式语言驱动的，你用来检索时间序列数据，进行计算，找出复杂的问题的答案，并可视化的结果。

这个功能由一系列的功能函数组成，同样的查询的结果，也可以通过Dashboard显示查看。

## Management

管理中的应用是在你执行你的运行时配置kibana，包括初始设置和指标进行配置模式，高级设置，调整自己的行为和Kibana，各种“对象”，你可以查看保存在整个Kibana的内容如发现页，可视化和仪表板。  
这部分是pluginable，除此之外，X-pack可以给Kibana增加额外的管理能力。  
你可以使用X-pack安全控制哪些用户可以访问Elasticsearch数据通过Kibana。当你安装X-pack，Kibana用户登录。他们需要有kibana\_user作用以及获得的指标，他们将在Kibana的工作。如果用户加载Kibana仪表板，访问数据的一个索引，他们未被授权查看，他们得到一个错误，表明指数不存在。X-pack安全目前并不提供一种方法来控制哪些用户可以负荷的仪表板。

## Dev Tools

原先的交互式控制台Sense，使用户方便的通过浏览器直接与Elasticsearch进行交互。从Kibana 5开始改名并直接内建在Kibana，就是Dev Tools选项。