**ELK日志系统详解**

数工研发部：舒全亮、易金金

2018年12月05日

# ELK简介

ELK是三个开源软件的缩写，分别表示：Elasticsearch , Logstash, Kibana , 它们都是开源软件。适合于在各个服务器上搜集日志后传输给Logstash，官方也推荐此工具。

Elasticsearch是个开源分布式搜索引擎，提供搜集、分析、存储数据三大功能。它的特点有：分布式，零配置，自动发现，索引自动分片，索引副本机制，restful风格接口，多数据源，自动搜索负载等。

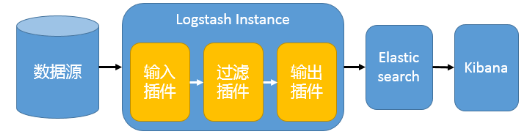
Logstash 主要是用来日志的搜集、分析、过滤日志的工具，支持大量的数据获取方式。一般工作方式为c/s架构，client端安装在需要收集日志的主机上，server端负责将收到的各节点日志进行过滤、修改等操作在一并发往elasticsearch上去。

Kibana 也是一个开源和免费的工具，Kibana可以为 Logstash 和 ElasticSearch 提供的日志分析友好的 Web 界面，可以帮助汇总、分析和搜索重要数据日志。

# 原理介绍

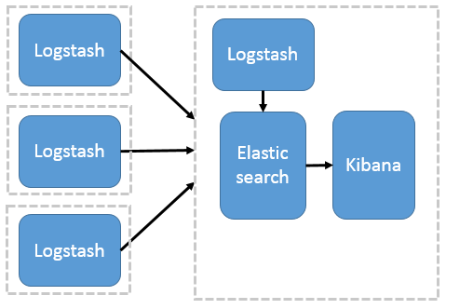
## ELK架构

**架构图一**：**最简单架构**



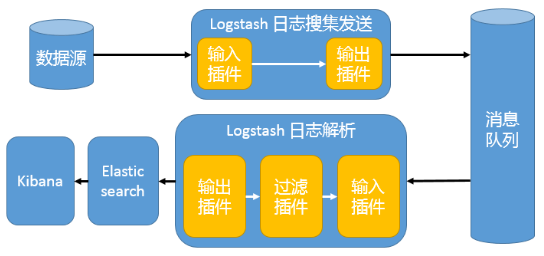
这种架构非常简单，使用场景也有限。初学者可以搭建这个架构，了解 ELK 如何工作。

**架构图二：Logstash 作为日志搜集器**



这种结构因为需要在各个服务器上部署 Logstash，而它比较消耗 CPU 和内存资源，所以比较适合计算资源丰富的服务器，否则容易造成服务器性能下降，甚至可能导致无法正常工作。

**架构图三：引入消息队列机制的架构**

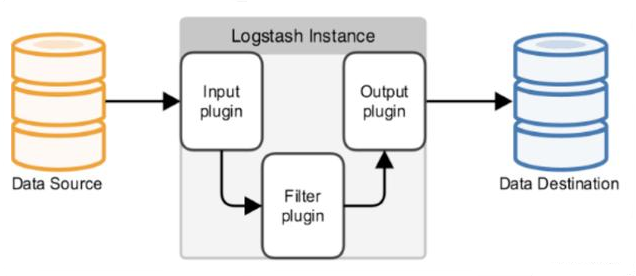


这种架构使用 Logstash 从各个数据源搜集数据，然后经消息队列输出插件输出到消息队列中。目前 Logstash 支持 Kafka、Redis、RabbitMQ 等常见消息队列。然后 Logstash 通过消息队列输入插件从队列中获取数据，分析过滤后经输出插件发送到 Elasticsearch，最后通过 Kibana 展示。详见图

这种架构适合于日志规模比较庞大的情况。但由于 Logstash 日志解析节点和 Elasticsearch 的负荷比较重，可将他们配置为集群模式，以分担负荷。引入消息队列，均衡了网络传输，从而降低了网络闭塞，尤其是丢失数据的可能性，但依然存在 Logstash 占用系统资源过多的问题。

## Logstash工作原理

Logstash事件处理有三个阶段：inputs → filters → outputs。是一个接收，处理，转发日志的工具。支持系统日志，webserver日志，错误日志，应用日志，总之包括所有可以抛出来的日志类型。



**Input：输入数据到logstash。**

一些常用的输入为：

file：从文件系统的文件中读取，类似于tial -f命令

syslog：在514端口上监听系统日志消息，并根据RFC3164标准进行解析

redis：从redis service中读取

beats：从filebeat中读取

**Filters：数据中间处理，对数据进行操作。**

一些常用的过滤器为：

grok：解析任意文本数据，Grok 是 Logstash 最重要的插件。它的主要作用就是将文本格式的字符串，转换成为具体的结构化的数据，配合正则表达式使用。内置120多个解析语法。

官方提供的grok表达式：

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

grok在线调试：https://grokdebug.herokuapp.com/

mutate：对字段进行转换。例如对字段进行删除、替换、修改、重命名等。

drop：丢弃一部分events不进行处理。

clone：拷贝 event，这个过程中也可以添加或移除字段。

geoip：添加地理信息(为前台kibana图形化展示使用)

**Outputs：outputs是logstash处理管道的最末端组件。一个event可以在处理过程中经过多重输出，但是一旦所有的outputs都执行结束，这个event也就完成生命周期。**

一些常见的outputs为：

elasticsearch：可以高效的保存数据，并且能够方便和简单的进行查询。

file：将event数据保存到文件中。

graphite：将event数据发送到图形化组件中，一个很流行的开源存储图形化展示的组件。

**Codecs：codecs 是基于数据流的过滤器，它可以作为input，output的一部分配置。**

Codecs可以帮助你轻松的分割发送过来已经被序列化的数据。

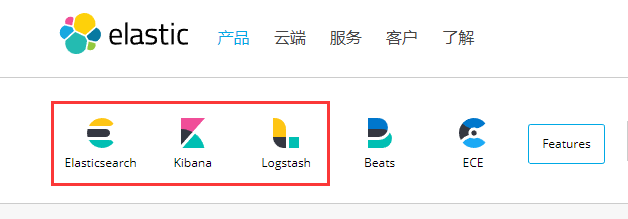
一些常见的codecs：

json：使用json格式对数据进行编码/解码。

multiline：将汇多个事件中数据汇总为一个单一的行。比如：java异常信息和堆栈信息。

# 下载ELK

下载地址：https://www.elastic.co/cn/products



下载完后上传到linux服务器 或者 直接在linux服务器上下载

Linux下载：

执行下面命令：

wget https://artifacts.elastic.co/downloads/logstash/logstash-6.5.1.tar.gz

wget https://artifacts.elastic.co/downloads/kibana/kibana-6.5.1-linux-x86\_64.tar.gz

wget https://artifacts.elastic.co/downloads/elasticsearch/elasticsearch-6.5.1.tar.gz

# 安装与配置

## 安装前说明：

采用架构一来搭建

ELK依赖JDK，因此要先安装JDK，采用JDK1.8版本

创建目录：elkhome，把下载的文件移动到该目录下

在elkhome目录解压：tar -xvf XXX解压即可

## JDK安装与配置

下载jdk

解压JDK：tar -zxvf jdk-8u144-linux-x64.tar.gz

配置JDK：

编辑：vi /etc/profile

增加下列配置：(JAVA\_HOME是自己的JDK安装目录)

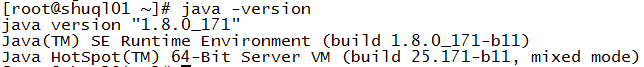
export JAVA\_HOME=/usr/java/jdk1.8.0\_144

export CLASSPATH=.:${JAVA\_HOME}/jre/lib/rt.jar:${JAVA\_HOME}/lib/dt.jar:${JAVA\_HOME}/lib/tools.jar

export PATH=$PATH:${JAVA\_HOME}/bin

生效新配置：source /etc/profile

验证是否安装成功：java –version



看到java版本信息说明安装成功了。

## 配置Elasticsearch

Elasticsearch不能以root用户来执行，先来创建一个elsearch用户

groupadd elsearch

useradd elsearch -g elsearch

passwd elsearch

usermod -s /bin/bash elsearch

设置目录 归属组和用户

chown -R elsearch:elsearch /home/elkhome

用新添加用户 elsearch 来登录服务器

编辑vi ./elasticsearch-6.5.1/config/elasticsearch.yml

1. cluster.name: shuql01

2. node.name: shuql01-node-1

3. bootstrap.memory\_lock: false

5. network.host: 192.168.22.126

6.http.port: 9200

启动 cd /elsearch/elasticsearch-6.5.1/bin/

./elasticsearch 启动elsearch

如果出现下面错误



解决：切换root

[1].编辑vi /etc/security/limits.conf，增加下列配置：

\* soft nofile 65536

\* hard nofile 131072

\* soft nproc 2048

\* hard nproc 4096

[2].编辑vi /etc/security/limits.d/90-nproc.conf

\* soft nproc 1024 #修改为 2048

如果报2048不够，在加到4096 依次类推

[3].编辑vi /etc/sysctl.conf，增加下列配置：

vm.max\_map\_count=655360

执行：sysctl -p 重新启动

再使用elsearch用户来执行启动命令，如果需要后台执行，则使用

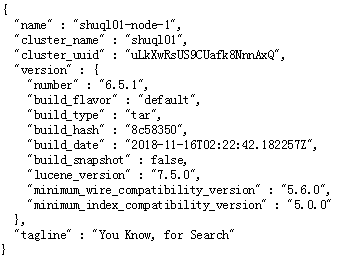
./elasticsearch –d

curl http://192.168.22.126:9200/?pretty

或者

浏览器访问

出现下列信息，说明成功了



## 配置kibana

kibana的配置文件在

/elsearch/kibana-6.5.1-linux-x86\_64/config/kibana.yml

elasticsearch.url: "http://192.168.22.126:9200" //访问elsearch的地址

server.port: 5601 //kibana的访问端口

server.host: "192.168.22.126" //kibana的访问地址。改为本机的IP

并去掉之前的 #

server.host: "192.168.22.126"，其它的就按默认文件配置。

启动 kibana

cd /elsearch/kibana-6.5.1-linux-x86\_64/bin/

nohup ./kibana &

## 配置Logstash

**宏观的配置格式如下：**

#　输入

input {

...

}

# 过滤器

filter {

...

}

# 输出

output {

...

}

**客观配置如下：**

进入config下：

编辑vi ./logstack.conf 增加下列配置

input {

#配置tcp连接方式，还可以配置其他的

tcp {

host => "192.168.22.126" #访问地址

port => 9220 #访问端口

codec => json\_lines #日志格式json

}

}

filter{

#过滤规则

}

output {

#配置日志写入es

elasticsearch {

hosts => "192.168.22.126:9200"

index => "%{[appname]} " #配置索引，采用应用名称

}

#配置控制台打印

stdout {

codec => json\_lines

}

}

启动./bin/logstash -f ./config/logstash.conf &

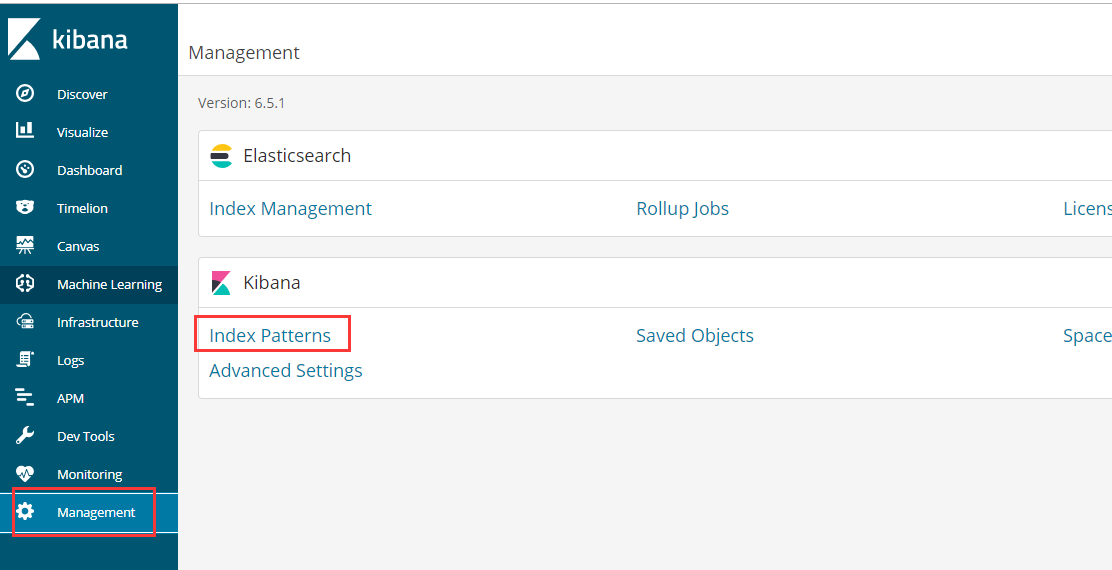
后台运行：

nohup ./elkhome/logstash-6.5.1/bin/logstash -f ./elkhome/logstash-6.5.1/config/logstash.conf -w 10 -l ./elkhome/logstash-6.5.1/logs/logstash.log &

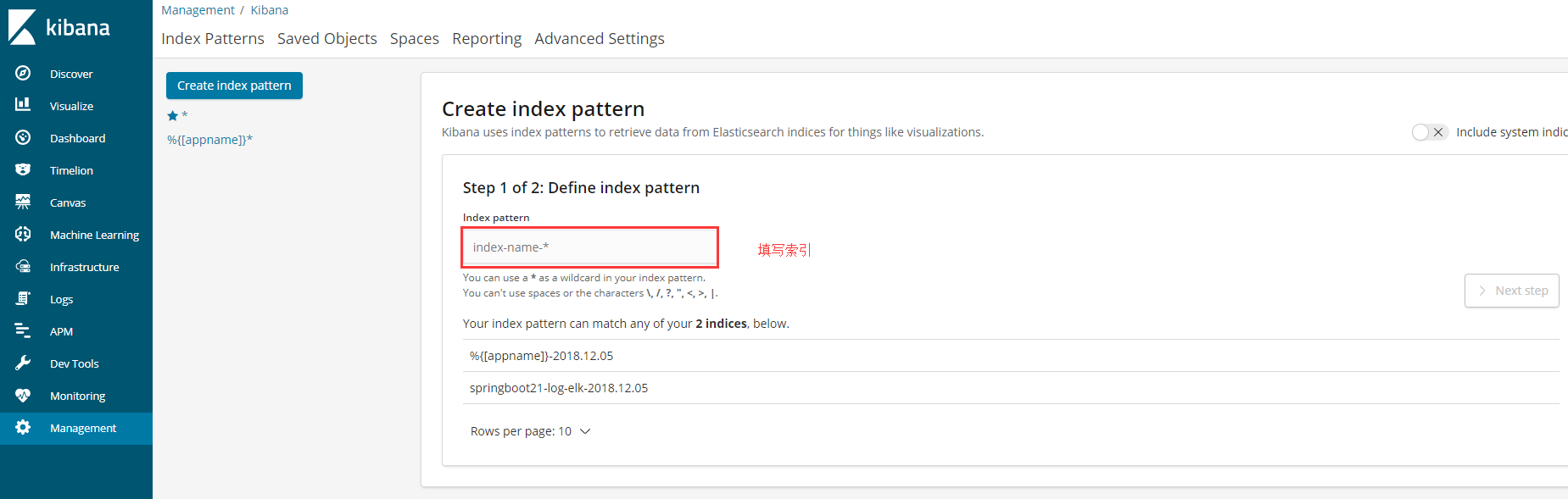
nohup /usr/local/logstash/bin/logstash -f /etc/logstash/logstash.conf -w 10 -l /var/log/logstash/logstash.log &

刷新 http://192.168.22.126:5601

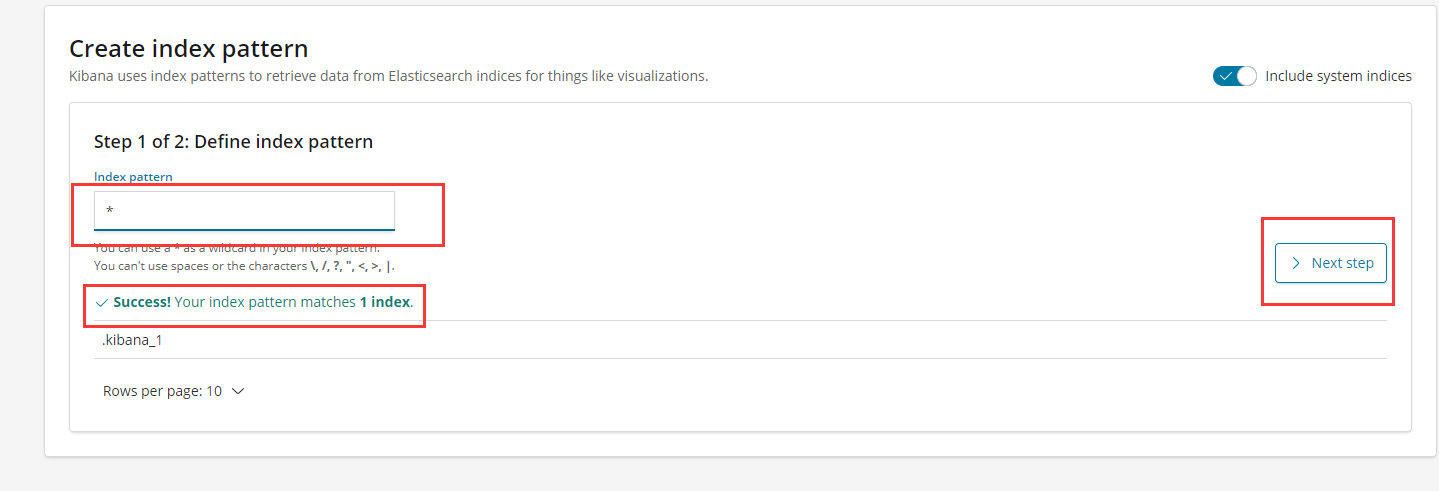
1. 点击Management
2. 点击Index Patterns

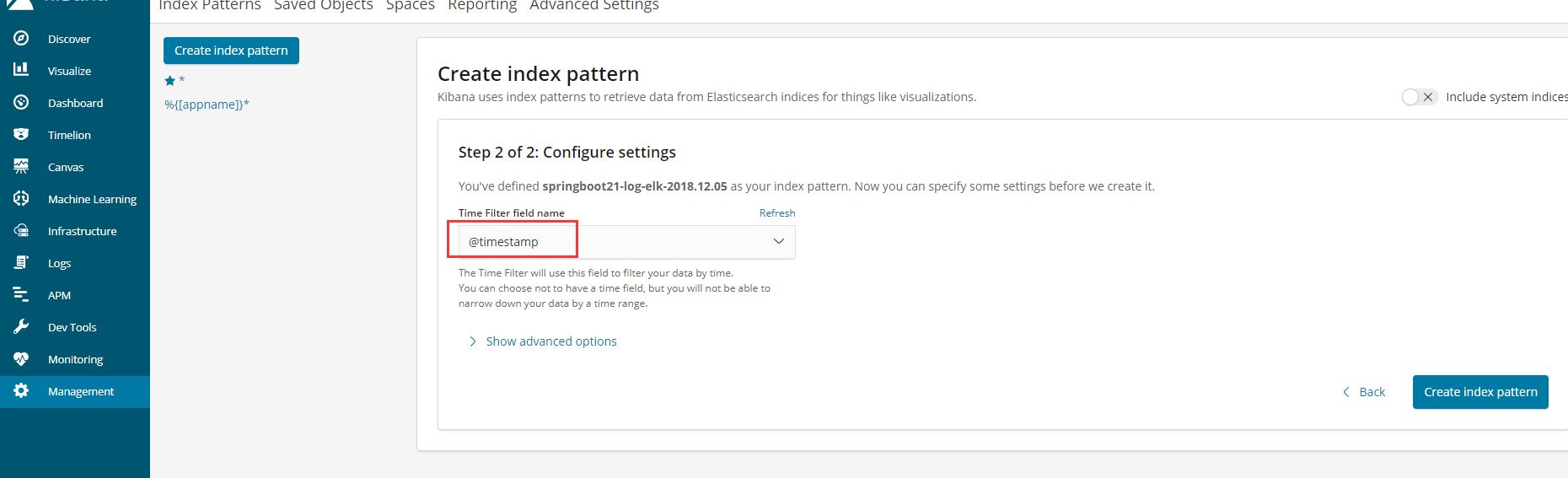


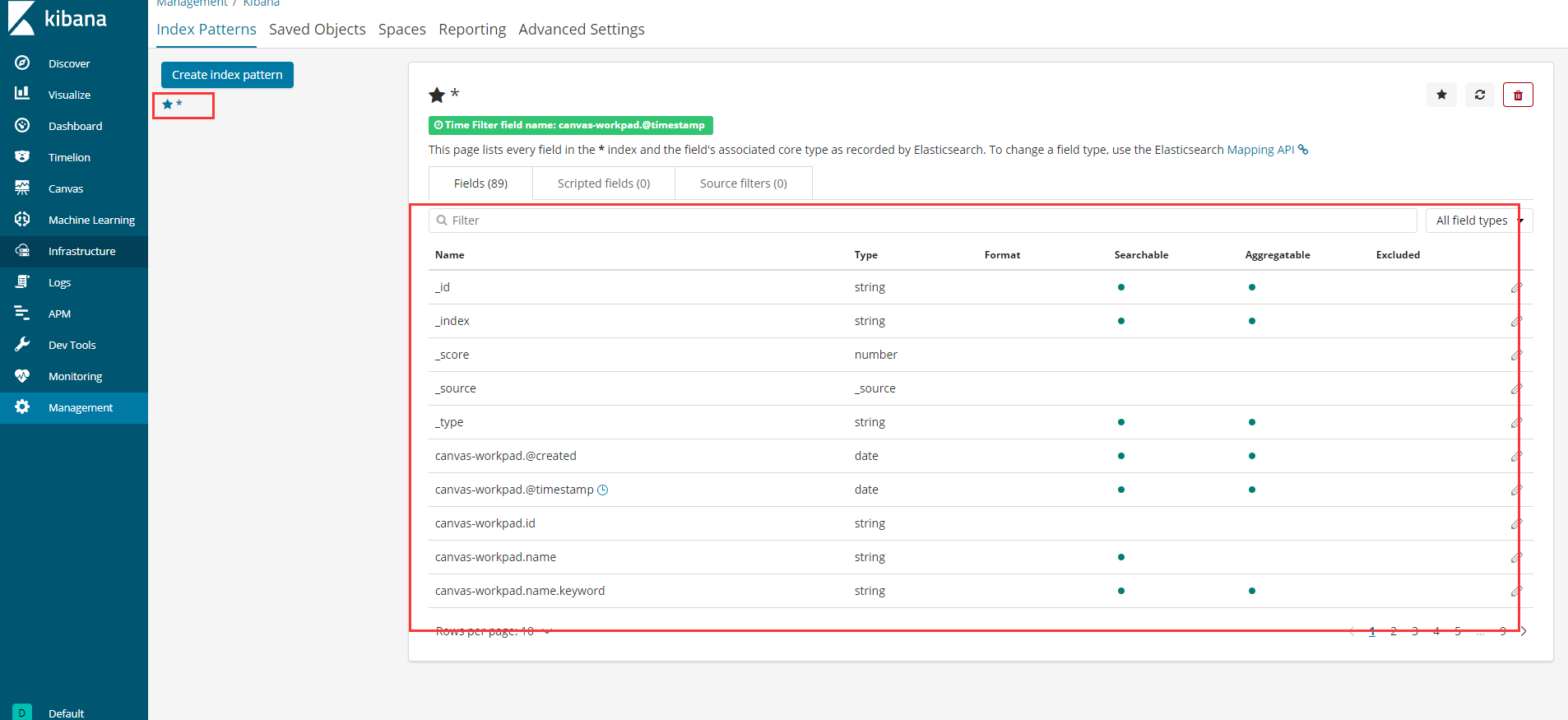
1. 创建pattern



1. 搜索 \* ，然后点击 Next step

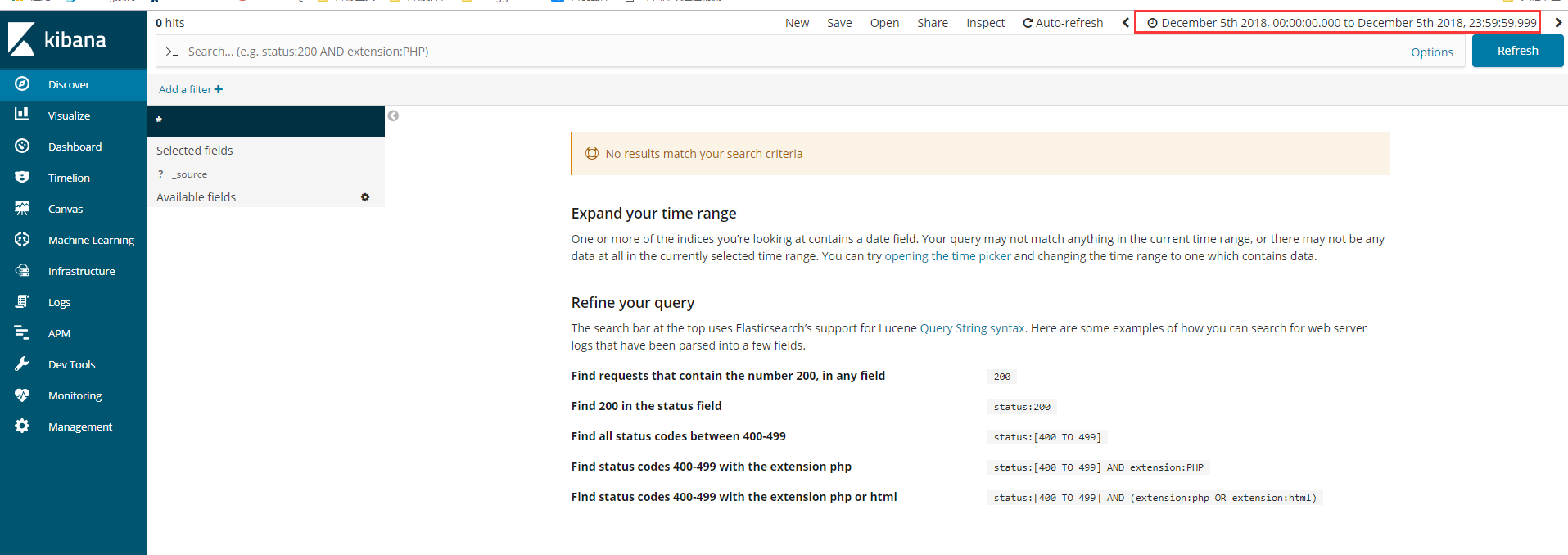




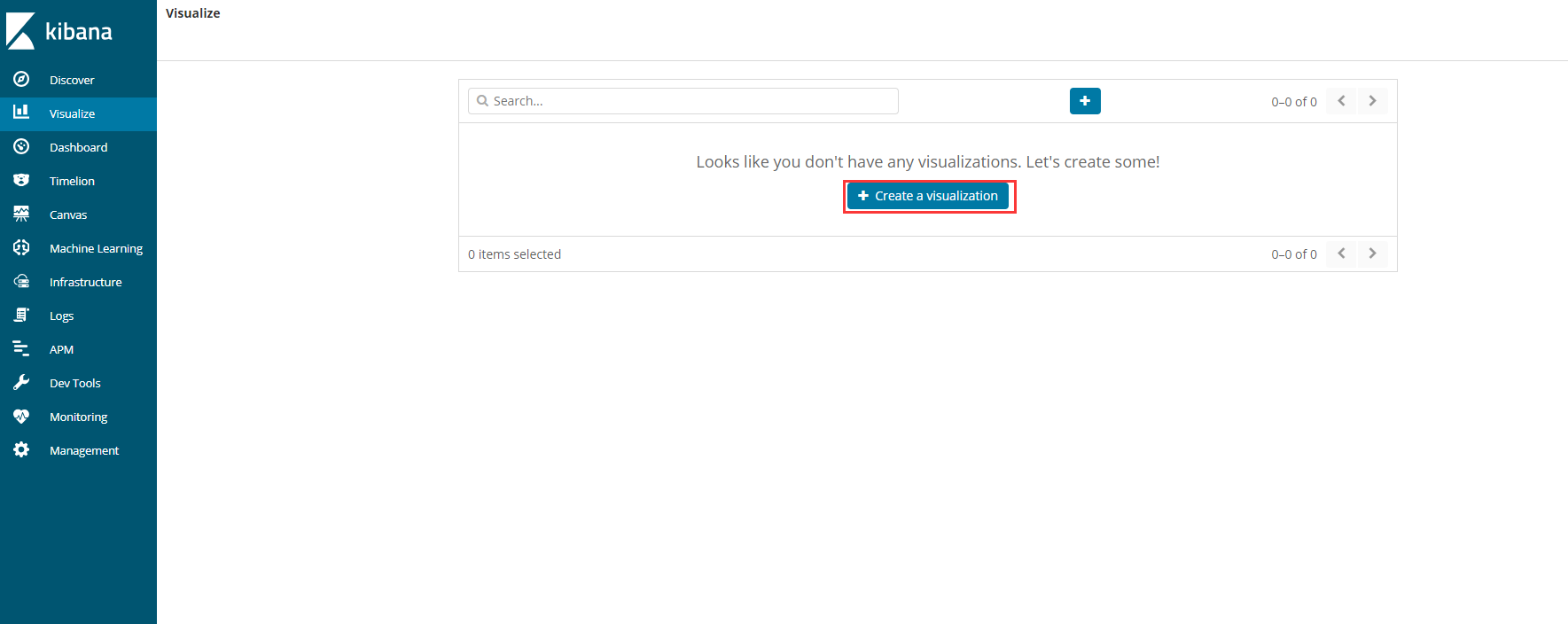


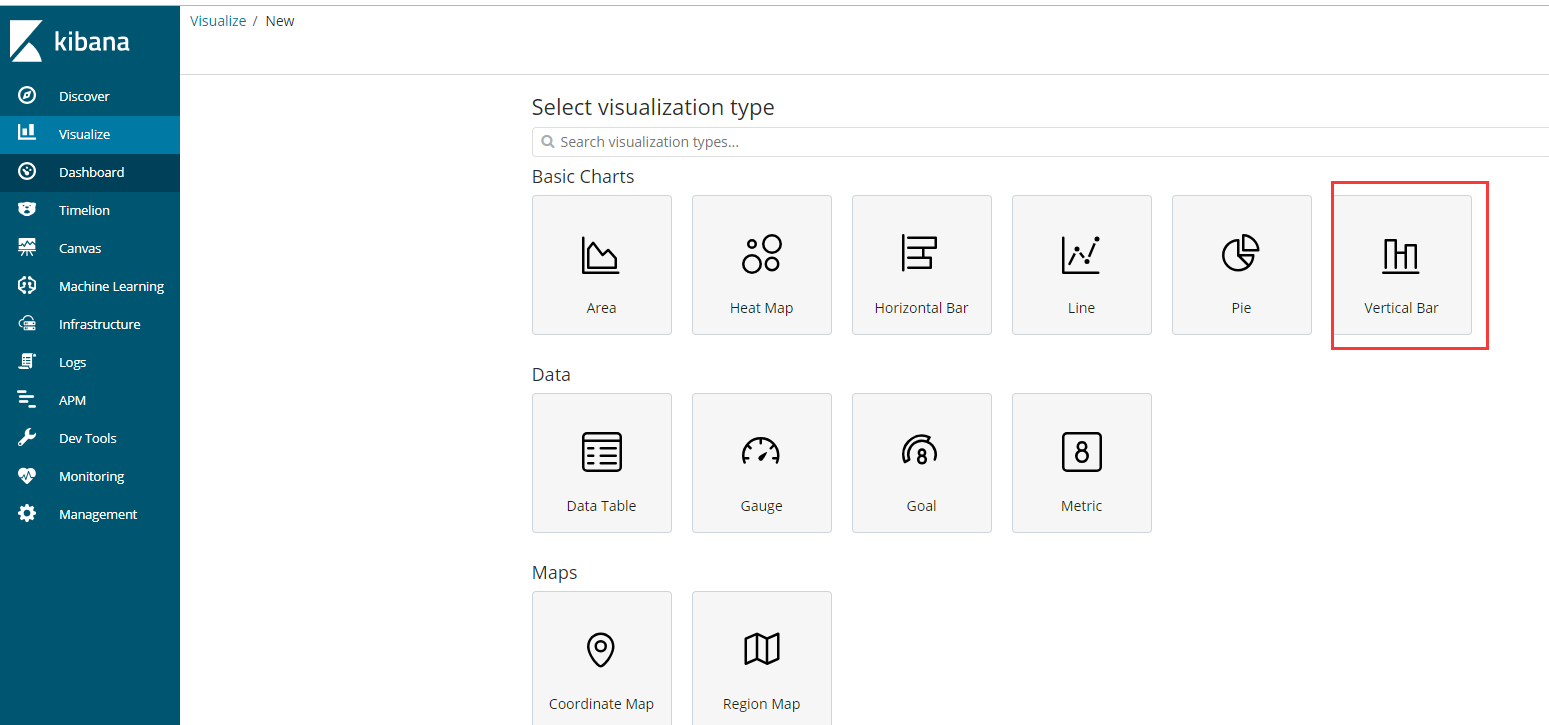
就可以看到日志导入elsearch中的一些信息

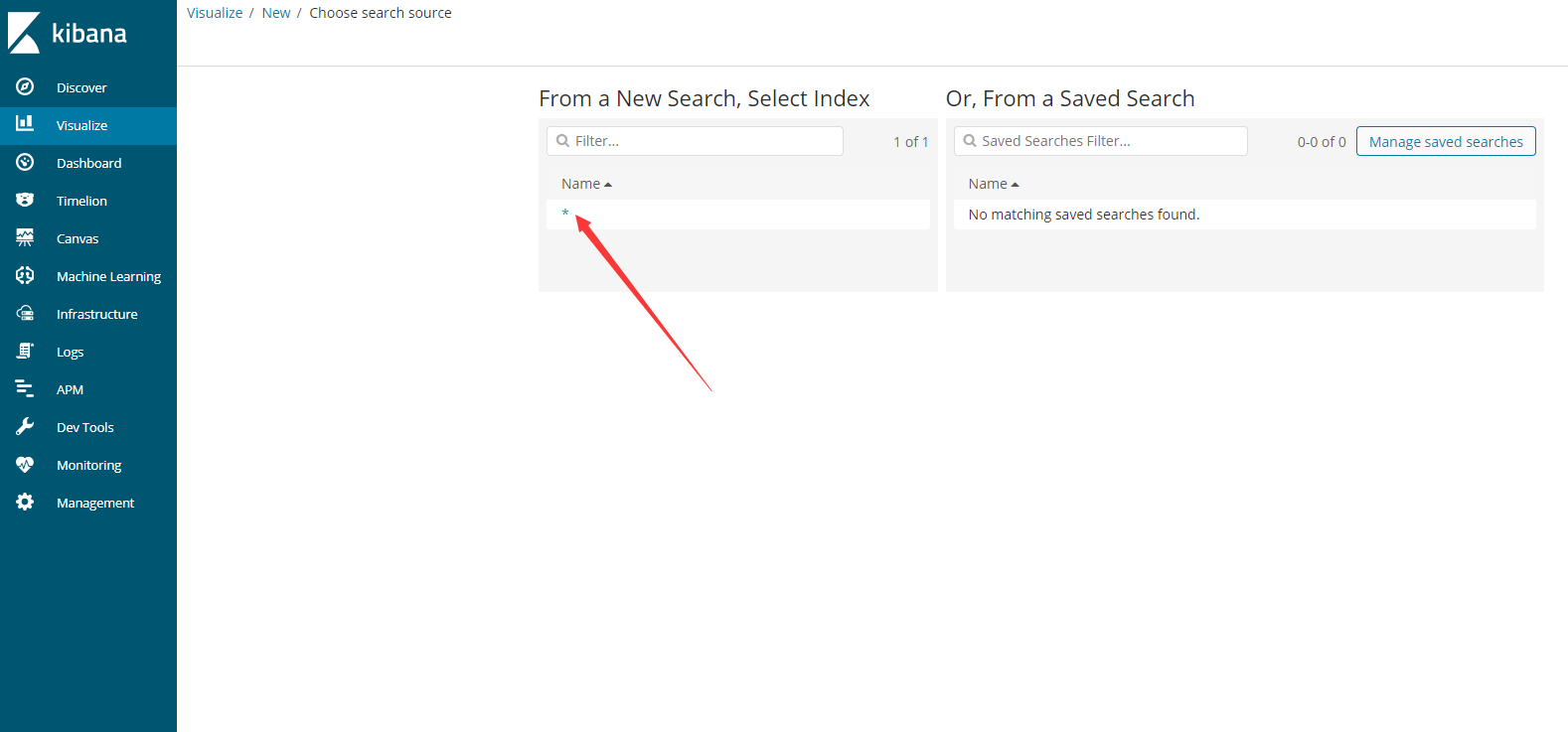
1. 点击左侧栏的 Discover,然后在右上角选择日志的时间段，因为我这里只有一部分测试数据，所以会比较少



1. 点击左侧栏的 Visualize,然后点击 create







如果有数据的话就会看到柱状图像

# SpringBoot集成elk

## 创建springboot项目

## 添加依赖

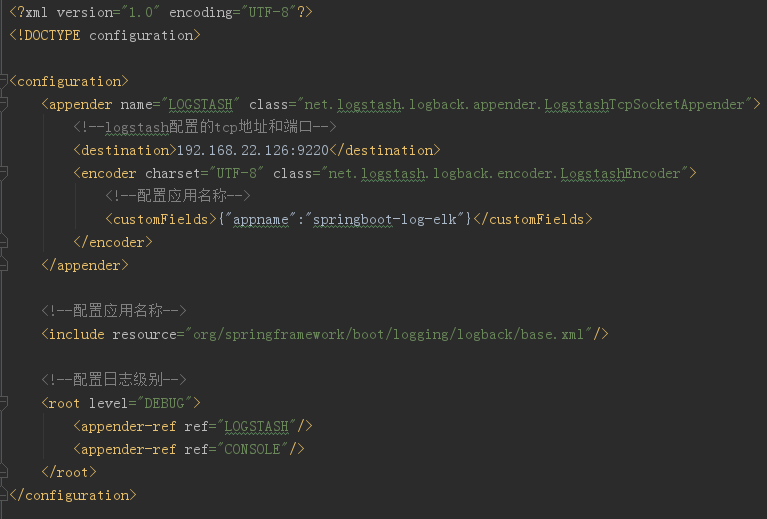
在pom.xml文件添加下面依赖

<dependency>  
 <groupId>net.logstash.logback</groupId>  
 <artifactId>logstash-logback-encoder</artifactId>  
 <version>5.1</version>  
</dependency>

## 添加日志配置文件

在resources资源文件路面创建logstash.xml

添加下列配置：



复制下面：

<?xml version="1.0" encoding="UTF-8"?>

<!DOCTYPE configuration>

<configuration>

<appender name="LOGSTASH" class="net.logstash.logback.appender.LogstashTcpSocketAppender">

<!--logstash配置的tcp地址和端口-->

<destination>192.168.22.126:9220</destination>

<encoder charset="UTF-8" class="net.logstash.logback.encoder.LogstashEncoder">

<!--配置应用名称-->

<customFields>{"appname":"springboot-log-elk"}</customFields>

</encoder>

</appender>

<!--配置应用名称-->

<include resource="org/springframework/boot/logging/logback/base.xml"/>

<!-- 日志级别 Trace -> Debug-> Info -> Warn-> Error-> Fatal -->

<root level="DEBUG">

<appender-ref ref="LOGSTASH"/>

<appender-ref ref="CONSOLE"/>

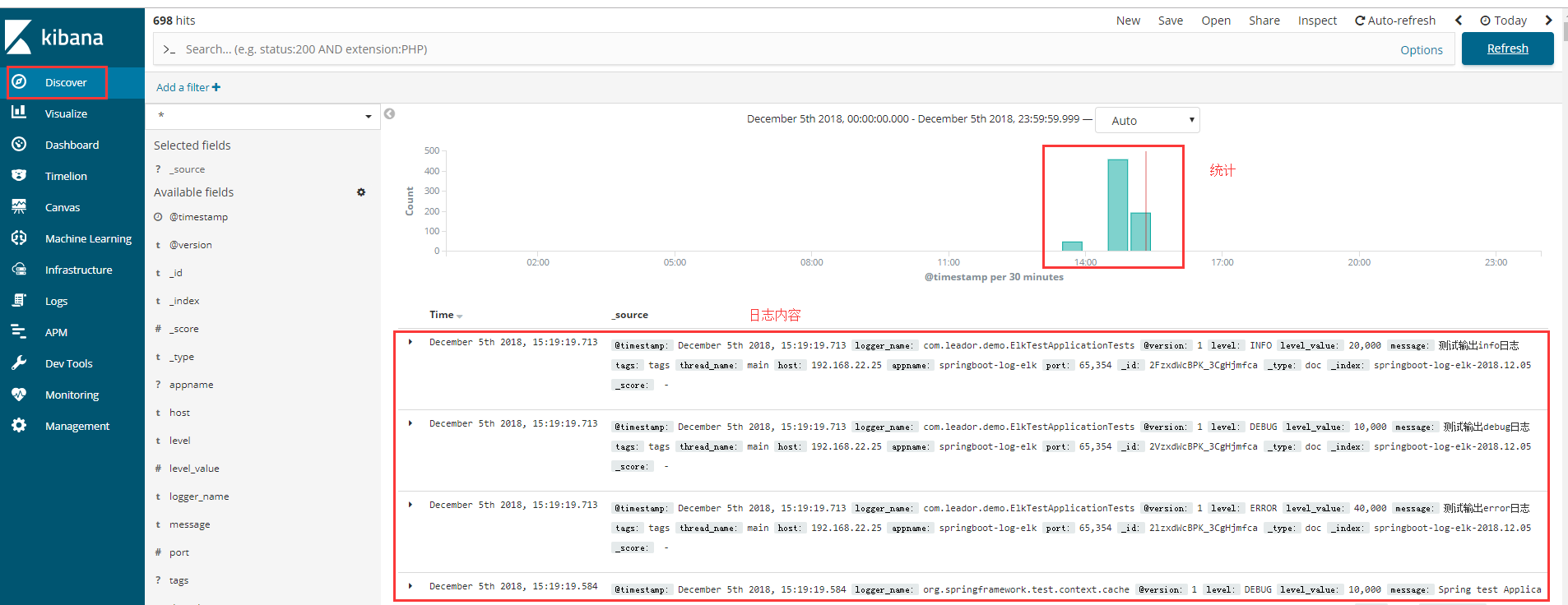
</root>

</configuration>

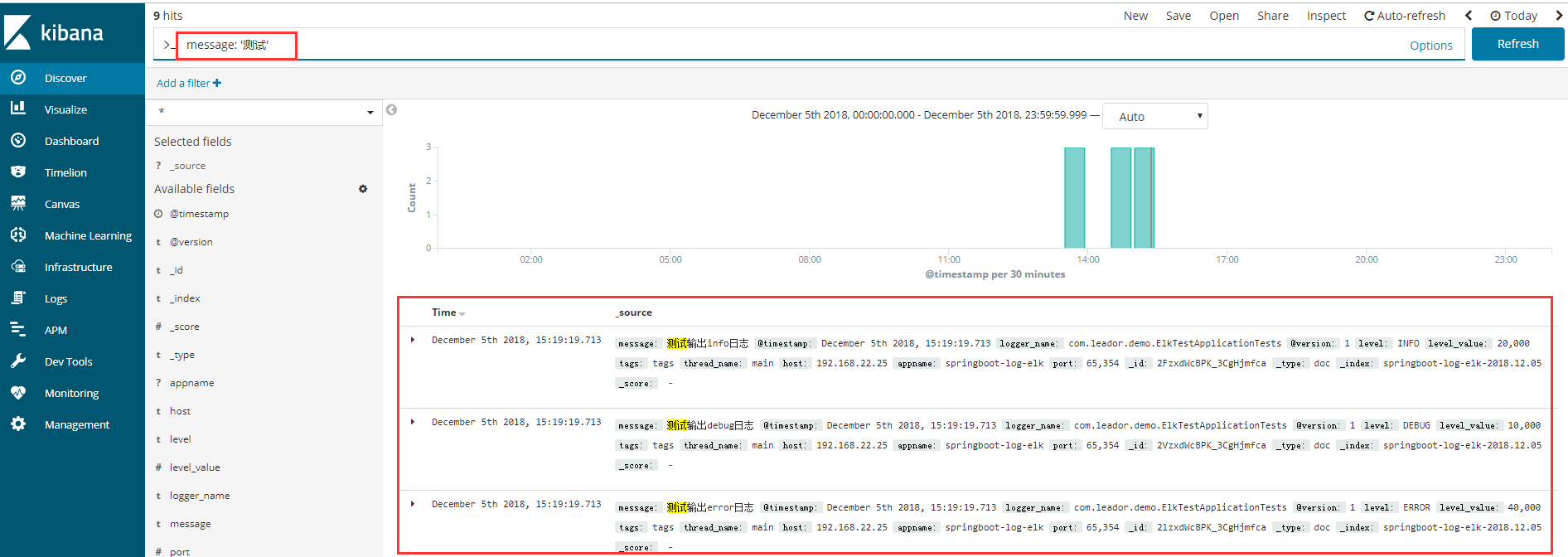
## 运行测试代码



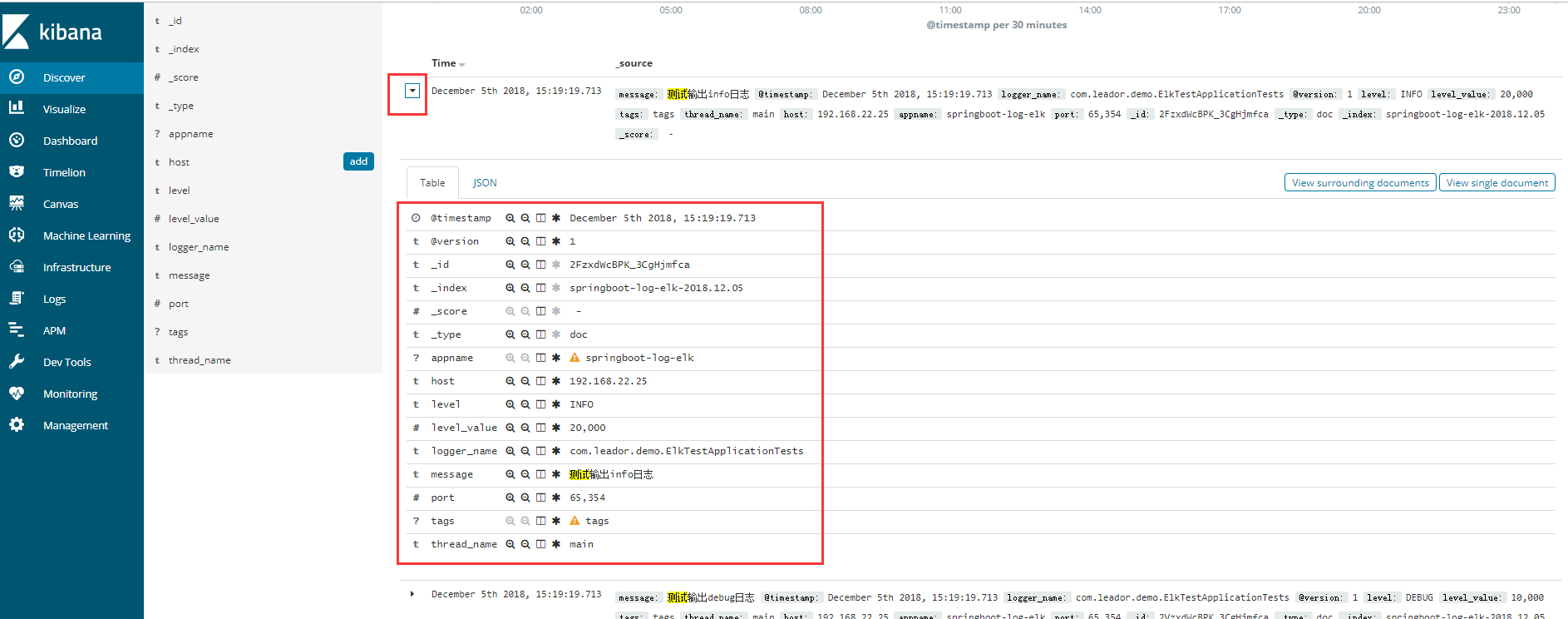
## 在Kibana上看日志



## 快速查找自己的日志



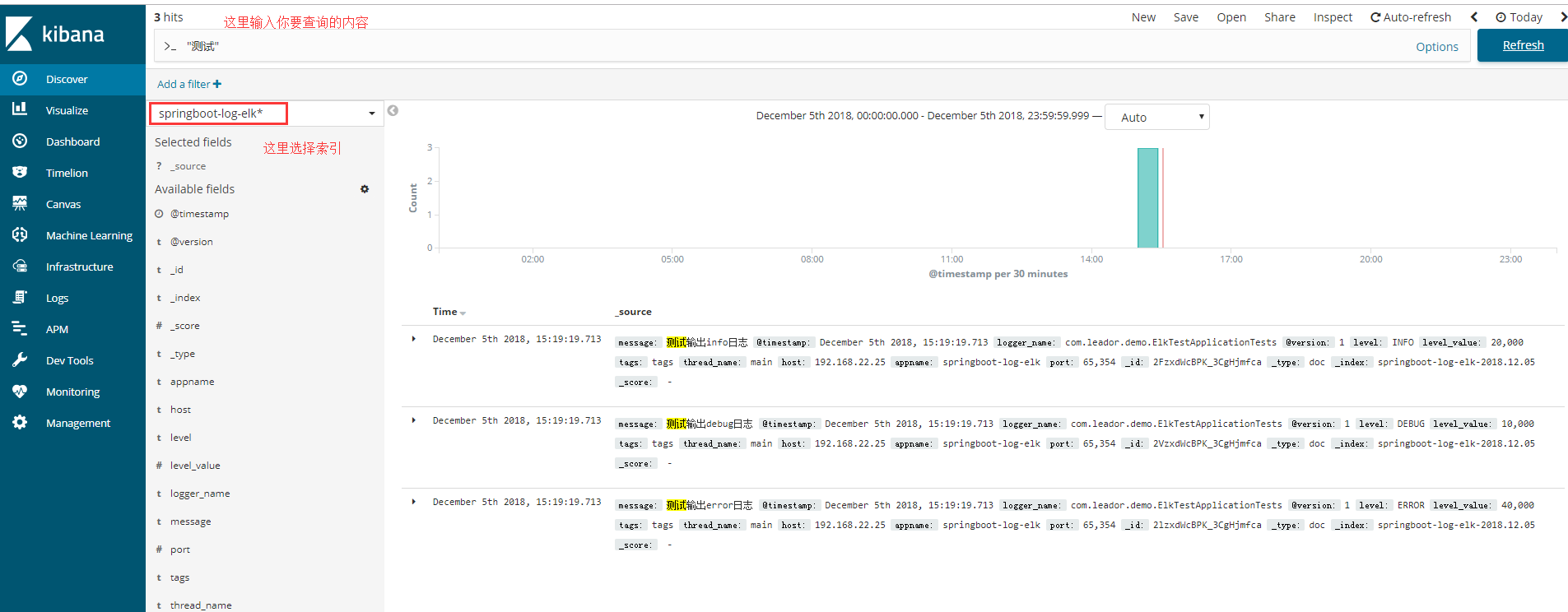
## 查看日志全部信息



# Kibana查询语法

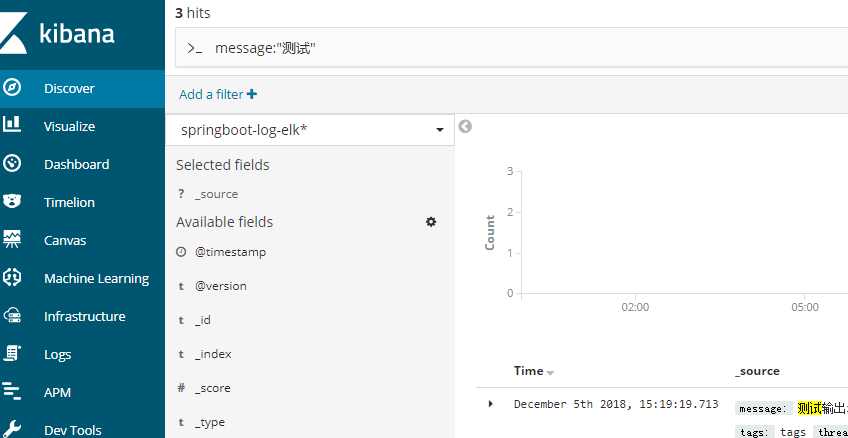
## 全文搜索

使用双引号包起来作为一个短语搜索：”测试”



## 字段搜索

field:value



## 通配符搜索

? 匹配单个字符  
\* 匹配0到多个字符

kiba?a,  el\*search

但是? \* 不能用第一个字符，如?text

## 模糊搜索

~:在一个单词后面加上~启用模糊搜索，可以搜到一些拼写错误的单词

quikc~ brwn~ foks~

## 近似搜索

在短语后面加上~，可以搜到被隔开或顺序不同的单词

"where select"~5 表示 select 和 where 中间可以隔着5个单词，可以搜到 select password from users where id=1

## 范围搜索

数值/时间/IP/字符串 类型的字段可以对某一范围进行查询

length:[100 TO 200]

sip:["172.24.20.110" TO "172.24.20.140"]

date:{"now-6h" TO "now"}

tag:{b TO e} 搜索b到e中间的字符

count:[10 TO \*] \* 表示一端不限制范围

count:[1 TO 5} [ ] 表示端点数值包含在范围内，{ } 表示端点数值不包含在范围内，可以混合使用，此语句为1到5，包括1，不包括5

可以简化成以下写法：

age:>10

age:<=10

age:(>=10 AND <20)

## 优先级

quick^2 fox  
使用^使一个词语比另一个搜索优先级更高，默认为1，可以为0~1之间的浮点数，来降低优先级

## 逻辑操作

AND  
OR

+：搜索结果中必须包含此项  
-：不能含有此项  
+apache -jakarta test aaa bbb：结果中必须存在apache，

不能有jakarta，剩余部分尽量都匹配到

## 分组

(jakarta OR apache) AND jakarta

## 字段分组

title:(+return +"pink panther")  
host:(baidu OR qq OR google) AND host:(com OR cn)

# ELK集群配置

## 集群安装

单台安装，上面已经描述，不再重复

假设已经部署了3台，IP分别如下：

192.168.22.126 (主节点 –> master-node)

192.168.22.127 (数据节点 –> data-node1)

192.168.22.128 (数据节点 –> data-node2)

yum list telnet\* 列出telnet相关的安装包

yum install telnet-server 安装telnet服务

yum install telnet.\* 安装telnet客户端

## 配置3台机器hosts

编辑：vi /etc/hosts

192.168.22.126 master-node

192.168.22.127 data-node1

192.168.22.128 data-node2

3台都关闭防火墙

## 集群配置

编辑：vi ./elasticsearch-6.5.1/config/elasticsearch.yml

**配置说明：**

elasticsearch的config文件夹里面有两个配置文件：elasticsearch.yml和logging.yml，第一个是es的基本配置文件，第二个是日志配置文件，es也是使用log4j来记录日志的，所以logging.yml里的设置按普通log4j配置文件来设置就行了。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](http://lib.csdn.net/base/linux" \t "_blank" \o "Linux知识库)下可以通过`ulimit-l unlimited`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](http://lib.csdn.net/base/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

**192.168.22.126机器配置：**

cluster.name: master-node # 集群中的名称

node.name: master # 该节点名称

node.master: true # 意思是该节点为主节点

node.data: false # 表示这不是数据节点

network.host: 0.0.0.0 # 监听全部ip，在实际环境中应设置为一个安全的ip

http.port: 9200 # es服务的端口号

discovery.zen.ping\_timeout: 120s

client.transport.ping\_timeout: 60s

discovery.zen.ping.unicast.hosts: ["192.168.22.126"] # 配置可以成为master的ip自动发现

discovery.zen.minimum\_master\_nodes: 1

**192.168.22.127机器配置：**

cluster.name: master-node # 集群中的名称

node.name: master # 该节点名称

node.master: false # 意思是该节点不为主节点

node.data: true # 表示这是数据节点

network.host: 0.0.0.0 # 监听全部ip，在实际环境中应设置为一个安全的ip

http.port: 9200 # es服务的端口号

discovery.zen.ping\_timeout: 120s

client.transport.ping\_timeout: 60s

discovery.zen.ping.unicast.hosts: ["192.168.22.126"] # 配置可以成为master的ip自动发现

discovery.zen.minimum\_master\_nodes: 1

**192.168.22.128机器配置：**

cluster.name: master-node # 集群中的名称

node.name: master # 该节点名称

node.master: false # 意思是该节点不为主节点

node.data: true # 表示这是数据节点

network.host: 0.0.0.0 # 监听全部ip，在实际环境中应设置为一个安全的ip

http.port: 9200 # es服务的端口号

discovery.zen.ping\_timeout: 120s

client.transport.ping\_timeout: 60s

discovery.zen.ping.unicast.hosts: ["192.168.22.126"] # 配置可以成为master的ip自动发现

discovery.zen.minimum\_master\_nodes: 1

**完成以上的配置之后，删除data目录下的文件，到主节点上，启动es服务**

netstat -lntp |grep java # es服务会监听两个端口



9300端口是集群通信用的，9200则是数据传输时用的。

主节点启动成功后，依次启动其他节点即可，我这里其他节点都是启动正常的。

## curl查看es集群情况

**集群的健康检查:**

curl '192.168.22.126:9200/\_cluster/health?pretty'



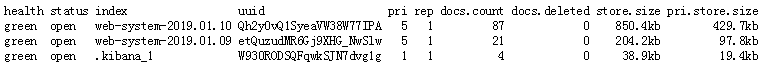
**查看集群的详细信息：**

curl '192.168.22.126:9200/\_cluster/state?pretty'

信息比较多，这里不截图了。

**查看集群的索引数：**

http://192.168.22.132:9200/\_cat/indices?v



status：状态，表明索引是否打开。

index：索引名称

uuid：索引内部随机分配的名称，表示唯一标识这个索引。

Pri：主分片，这个就是集群的主分片数。

docs.count：文档数，web-system在之前的演示添加了两条记录。

docs.deleted：已删除文档数，这里统计了被删除文档的数量。

store.size：索引存储的总容量。

pri.store.size：主分片的总容量。

**查看集群所在磁盘的分配状况：**

http://192.168.22.132:9200/\_cat/allocation?v



shards：分片数，集群中各节点的分片数相同，集群的总分片数为22。

disk.indices：索引所占空间，该节点中所有索引在该磁盘所点的空间。

disk.used：磁盘使用容量，已经使用空间548.4kb

disk.avail：磁盘可用容量，可用空间12.5gb

disk.total：磁盘总容量，总共容量17.1gb

disk.percent：磁盘便用率，磁盘使用率26%。

**查看集群的节点：**

http://192.168.22.132:9200/\_cat/nodes?v



heap.percent：堆内存使用情况

ram.percent：运行内存使用情况

cpu：cpu使用情况

检查没有问题后，我们的es集群就搭建完成了

## kibana配置

**编辑：vi /etc/kibana/kibana.yml # 增加以下内容**

server.port: 5601 # 配置kibana的端口

server.host: 192.168.22.126 # 配置监听ip

elasticsearch.url: "http://192.168.22.126:9200" # 配置es服务器的ip，如果是集群则配置该集群中主节点的ip

logging.dest: /var/log/kibana.log # 配置kibana的日志文件路径，不然默认是messages里记录日志

**创建日志文件**

touch /var/log/kibana.log;

chmod 777 /var/log/kibana.log

**启动kibana服务，并检查进程和监听端口：**

查看进程：ps aux |grep kibana

查看端口：netstat -lntp |grep 5601

## 日志收集配置

**架构一：**

无序在做，按之前搭建单台Logstash服务器，在通过SpringBoot集成，在通过tcp远程与Logstash通信传输日志

**架构二：**

在每台应用服务器都搭建一台Logstash

假如应用的日志写在 /home/elkhome/logs/elklog/\*.log

在每台Logstash的配置文件，如：Logstash.conf文件

只需修改：

#　输入

input {

file {

path => ["/home/elkhome/logs/elklog/\*.log"]

type => "springboot-log-elk"

start\_position => "beginning"

sincedb\_path => "/home/elkhome/logs/voip\_feedback.access"

codec => json {

charset => "UTF-8"

}

}

}

output {

elasticsearch {

hosts => "192.168.22.126:9200"

index => "%{[appname]}-%{+YYYY.MM.dd}"

}

stdout {

codec => json {

charset => "UTF-8"

}

}

}

**架构三：**

在架构二的基础上修改

1. Logstash收集的日志不直接给es，先放入kafka或redis
2. 搭建Logstash服务器，专门从kafka或redis去消费日志
3. 然后再上传到es里。

input {

kafka {

bootstrap\_servers => "IP:9092"

topics => "systemlog" #日志的topic

group\_id => "systemlog-g1" #kafka组id

}

}

output {

elasticsearch {

hosts => "192.168.22.126:9200"

index => "%{[appname]}-%{+YYYY.MM.dd}"

}

stdout {

codec => json {

charset => "UTF-8"

}

}

}