**ELK**

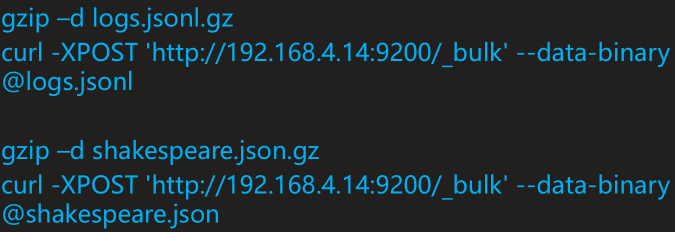
数据批量导入

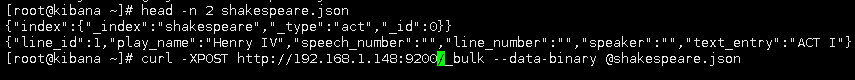
数据批量导入

使用\_bulk批量导入数据

批量导入数据使用POST方式，数据格式为json，url编码使用data-binary

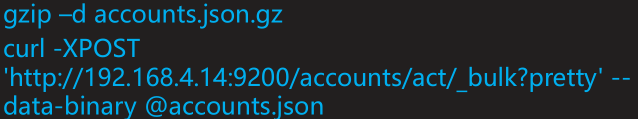
导入含有index配置的json文件

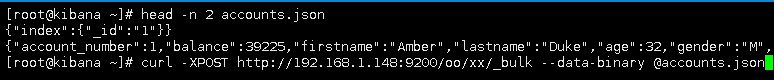




导入没有有index配置的json文件

需要在uri里面制定index和type





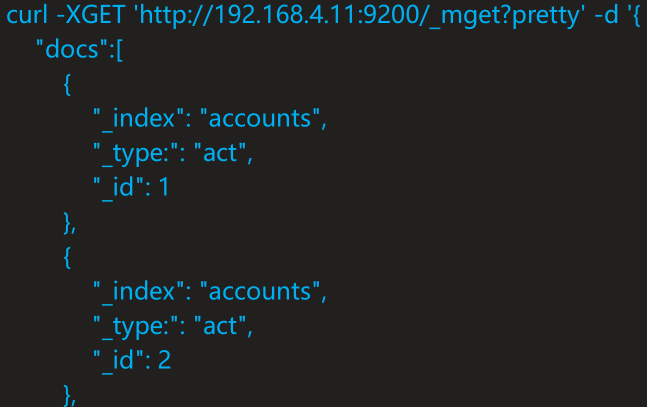


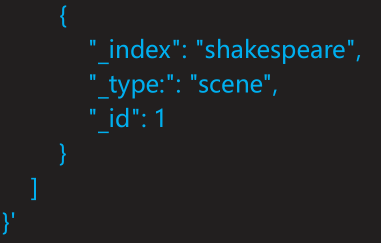


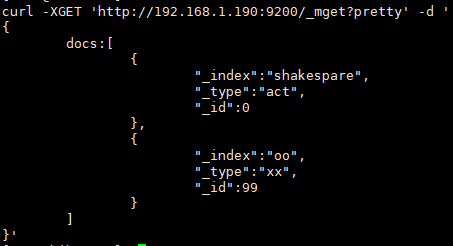
会生成随机的id

数量批量导入

数量批量查询使用GET







map映射

mapping：

映射：创建索引的时候，可以预先定义字段的类型及相关属性

作用：这样会让索引建立得更加的细致和完善

分类：静态映射和动态映射

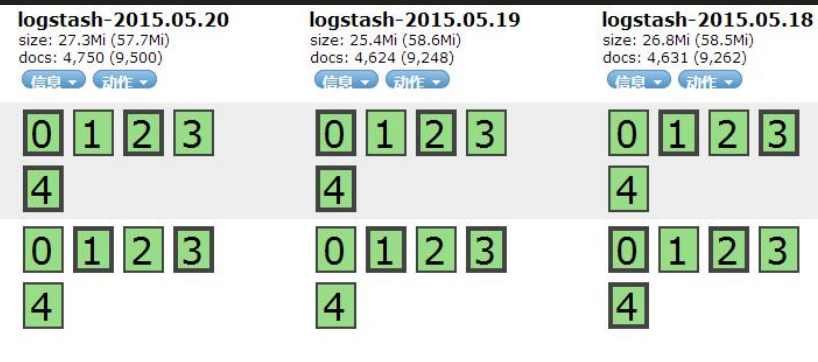
动态映射：自动根据数据进行相应的映射

精通映射：自定义字段映射数据类型

kibana部分

kibana部分

数据导入以后查看logs是否成功导入



修改kibana的配置文件后启动kibana，然后查看ES集群，如果出现.kibana index表示kibana与ES集群连接成功



kibana部分

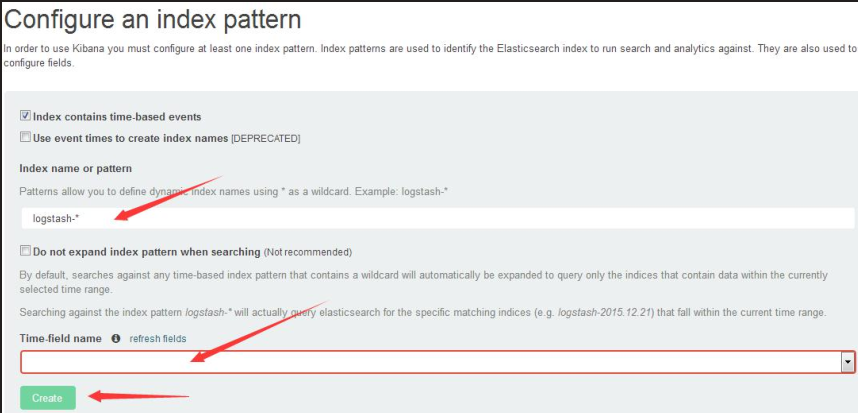
kibana里选择日志

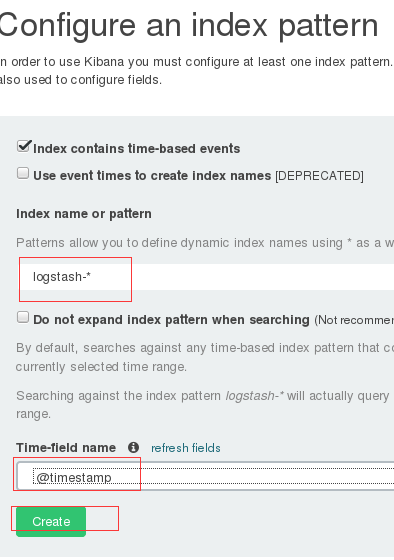
支持通配符\*

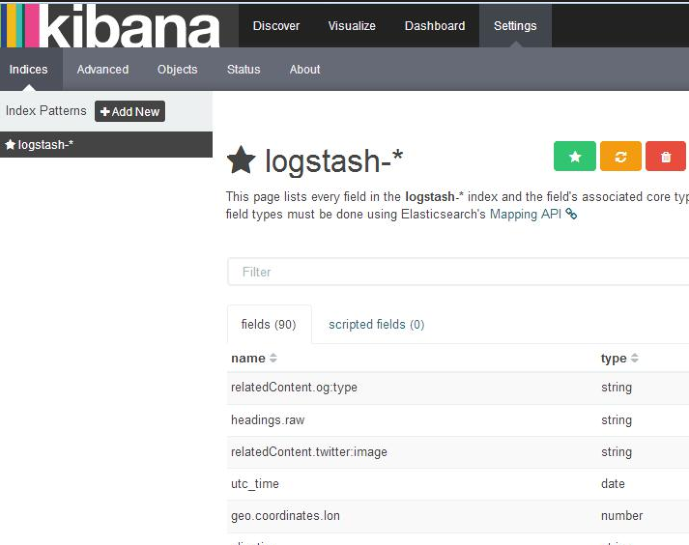
这里选择logstash-\*

在下面的time-field选择@timestramp作为索引

然后点create按钮

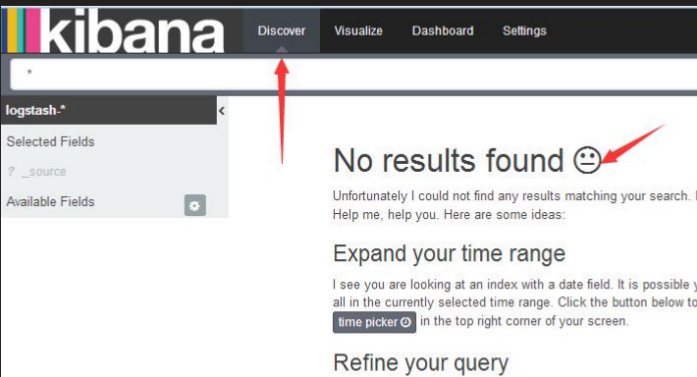






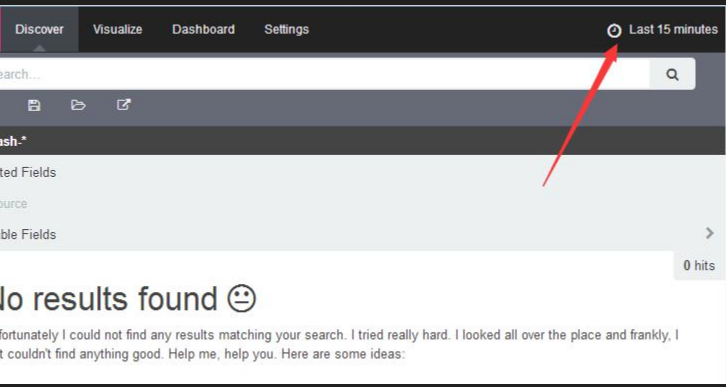


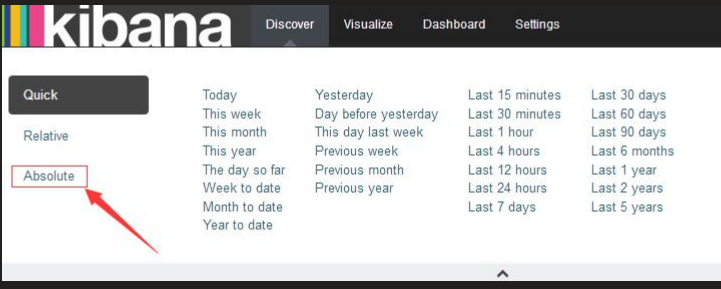
导入成功以后选择discover

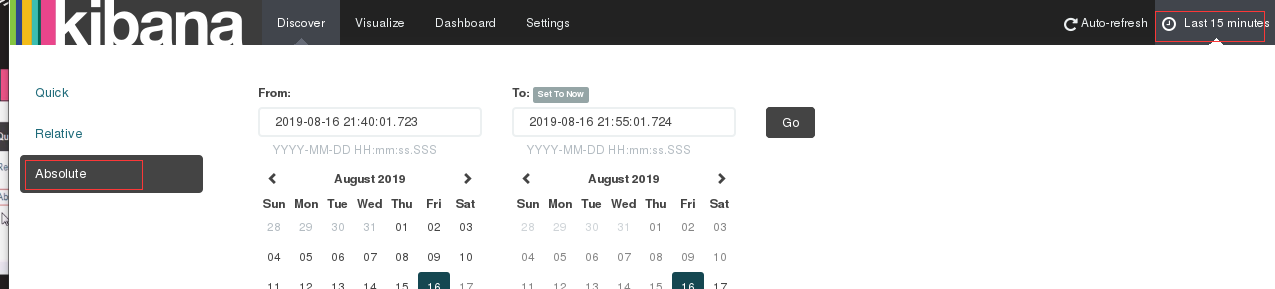


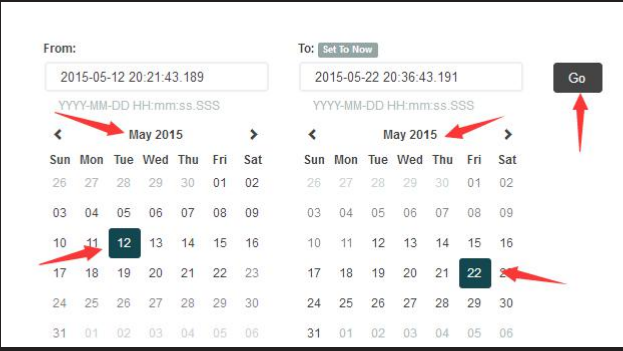
这里没有数据的原因是因为我们导入的日志是2015-05-10至2015-05-20时间段的，默认配置是最近15分钟，这里修改一下时间来显示

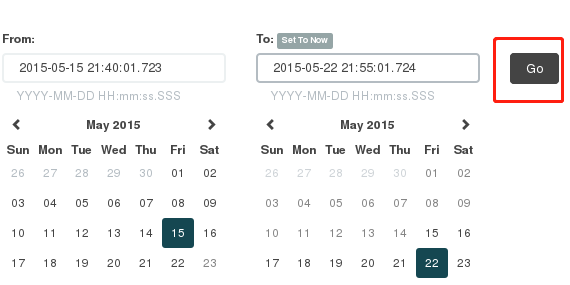
修改时间



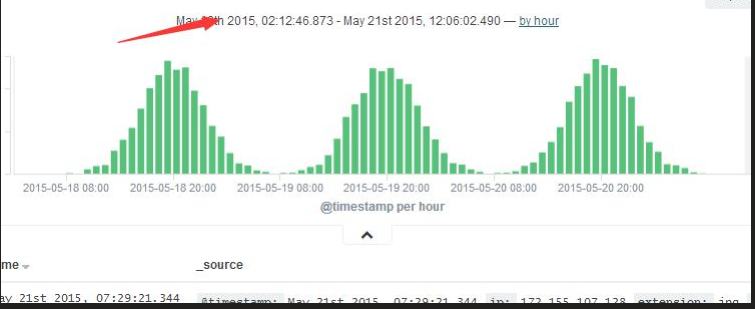


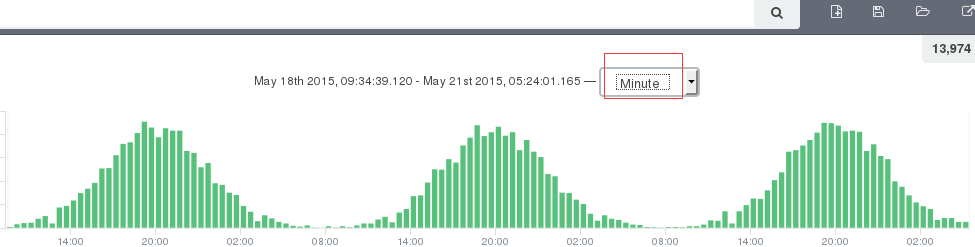




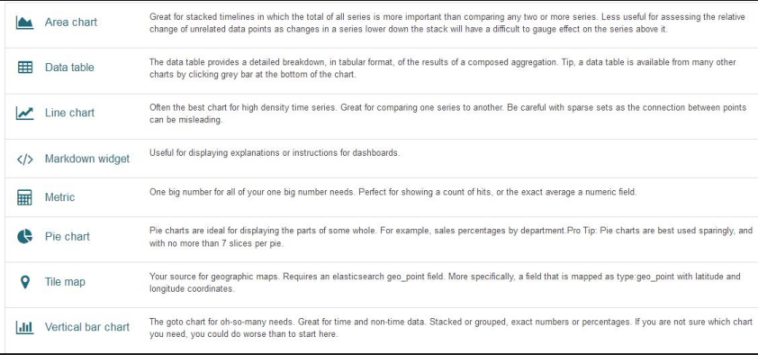


数据展示

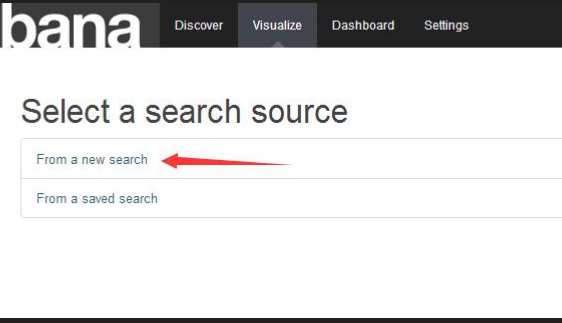


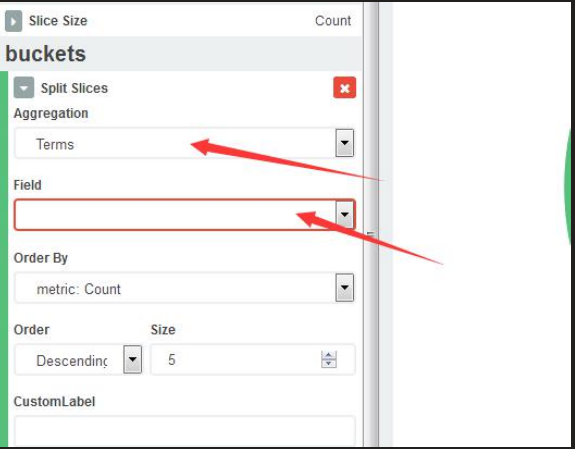
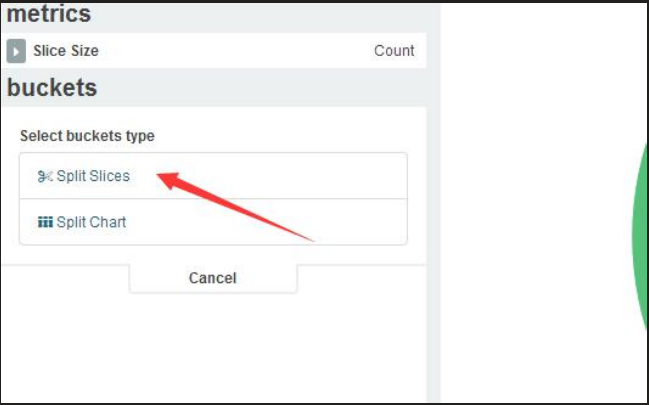
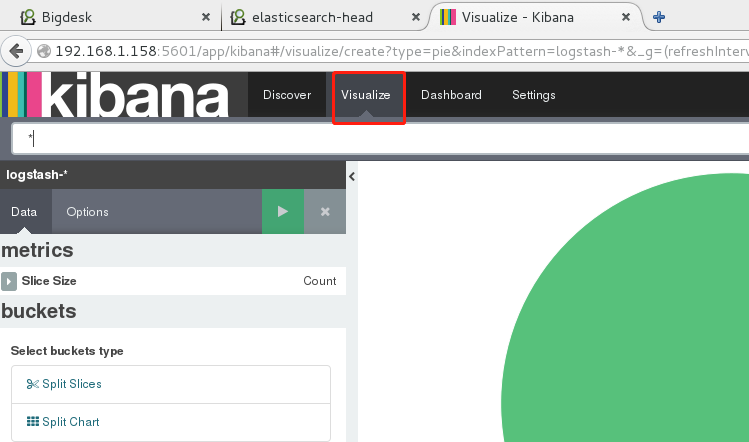


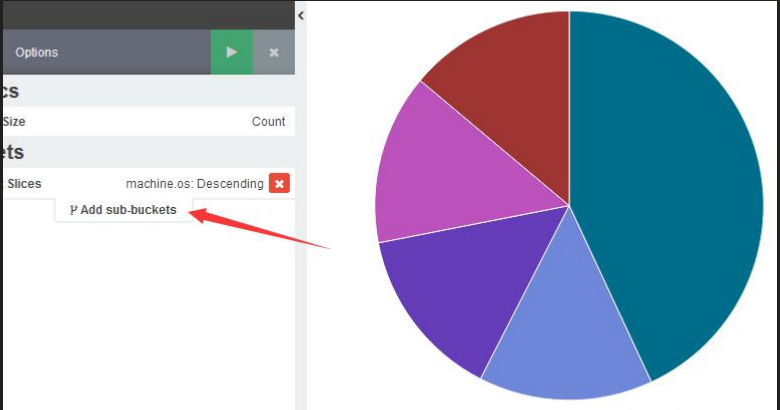
除了柱状图，kibana还支持很多种展示方式

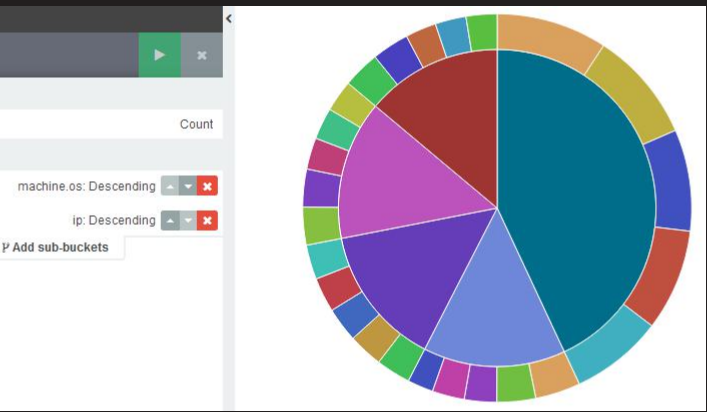
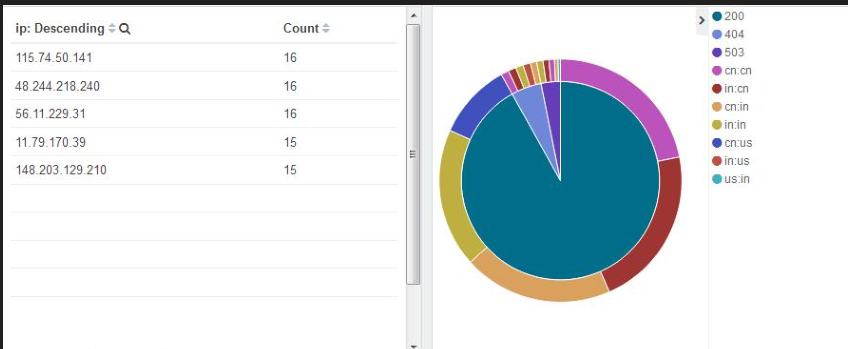


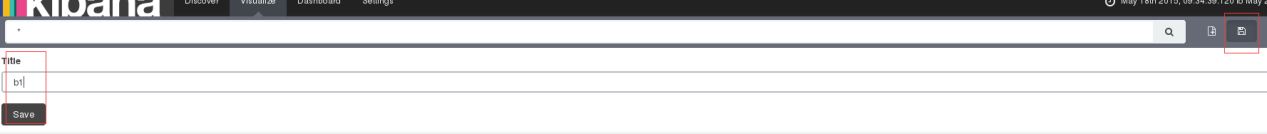
饼图与列表，多种维度自定义统计分析



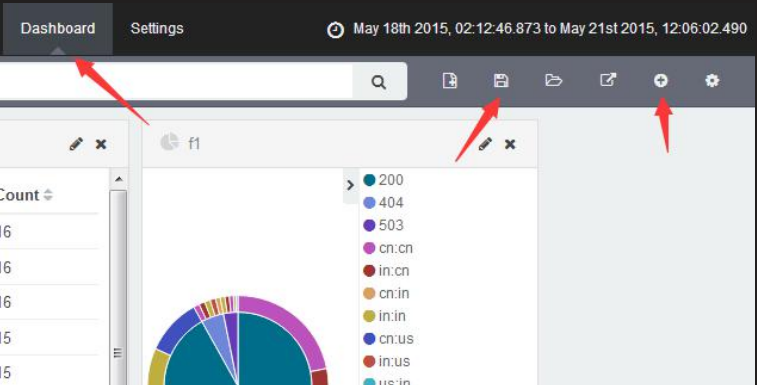






保存后，可以在dashboard查看





logstash部分

logstash部分

logstash是什么

logstash是一个数据采集、加工处理一级传输的工具

logstash特点：

所有类型的数据集中处理

不同模式和格式数据的正常化

自定义日志格式的迅速扩展

为定义数据源轻松添加插件

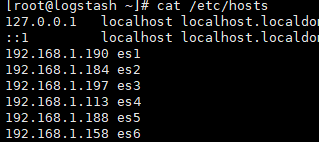
logstash安装

logstash依赖java环境，需要安装java-1.8.0-openjdk

logstash没有默认的配置文件，需要手动配置

logstash安装在/opt/logstash目录下

rpm -ivh logstash-2.3.4-1.noarch.rpm



因为要与elasticsearch相连，所以要配置hosts，可以不写kibana，因为不与kibana相连



logstash工作结构

{数据源}==>

input{}==>

filter{}==>

output{}==>

{ES}

logstash里面的类型

布尔值类型：ssl\_enable=>true

字节类型：bytes=>”1Mib”

字符串类型：name=>”xkops”

数值类型：port=>22

数组：match=>[“datetime”,”UNIX”]

哈希：options=>[k=>“v”,k2=>”v2”]

编码解码：codec=>”json”

路径：file\_path=>”/tmp/filename”

注释：#

logstash条件判断

等于：==

不等于：!=

小于：<

大于：>

小于等于：<=

大于等于：>=

匹配正则：=~

不匹配正则：!~

包含：in

不包含：not in

与：and

或：or

非与：nand

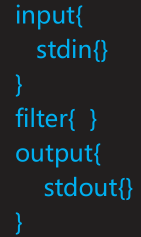
非或：xor

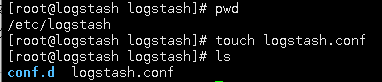
符合表达式：()

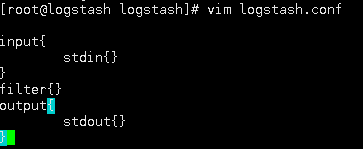
取反符合：!()

logstash的第一个配置文件

/etc/logstash/logstash.conf

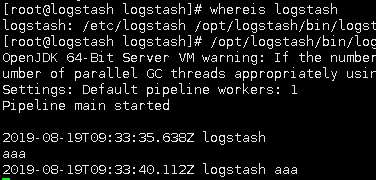






启动并验证

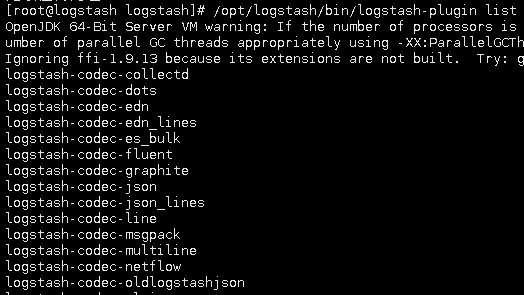
logstash -f logstash.conf



logstash插件

在配置文件中使用了logstash-input-stdin和logstash-output-stdout两个插件，logstash还有filter和codec类插件，查看插件的方式是

/opt/logstash/bin/logstash-plugin list



[https://github.com/logstash-plugins](https://github.com/logstash-plugins插件使用详情地址)

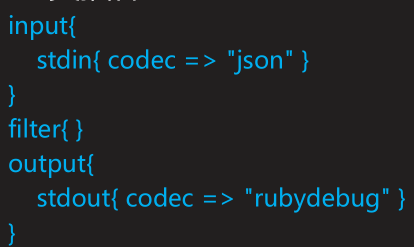
[插件使用详情地址](https://github.com/logstash-plugins插件使用详情地址)

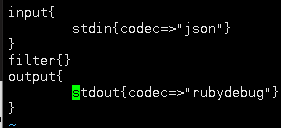
codec类插件

常用的插件：plain、json、json\_lines、rubydebug、multiline等

还使用刚刚的例子，不过这次输入json数据

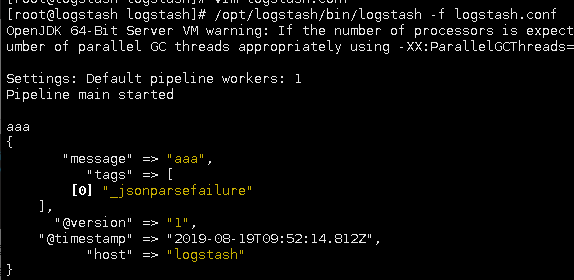
设置输入源的codec是json，在输入的时候选择rubydebug





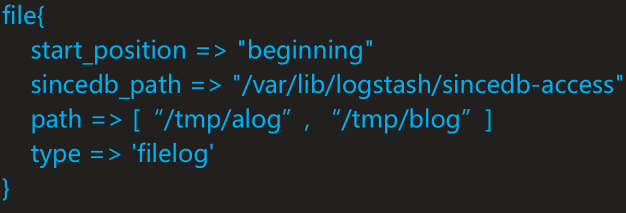
输入普通数据和json对比

{“a”:1,”c”:3,”b”:2}





练习input file插件



sincedb\_path记录读取文件的位置

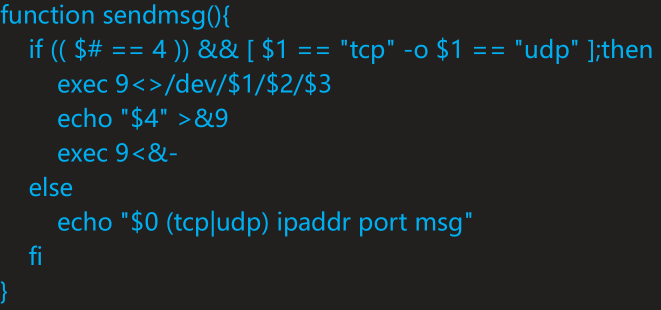
start\_position配置第一次读取文件从什么地方开始

练习input tcp和udp插件



tcp&udp练习

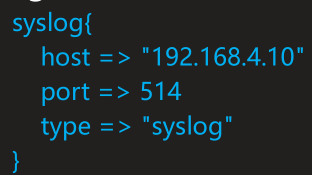
使用shell脚本，对tcp指定端口发送数据



发送tcp数据：sendmsg tcp 192.168.4.10 8888 ‘tcp msg’

发送udp数据：sendmsg udp 192.168.4.10 9999 ‘udp msg’

syslog插件练习



rsyslog.conf配置向远程发送数据：local0.info @@192.168.4.10:514

写syslog，查看状态：logger -p local0.info -t test\_logstash 'test message'

filter grok插件

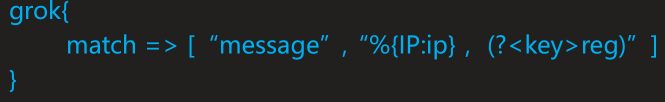
解析各种非结构化的日志数据插件

grok使用正则表达式把非结构化的数据结构化

在分组匹配，正则表达式需要根据具体数据结构编写

虽然编写困难，但适用性极广

几乎可以应用于各类数据



grok正则分组匹配

匹配ip时间戳和请求方法

"(?<ip>(\d+\.){3}\d+) \S+ \S+

(?<time>.\*\])\s+\"(?<method>[A-Z]+)"]

使用正则宏

%{IPORHOST:clientip} %{HTTPDUSER:ident} %{USER:auth}

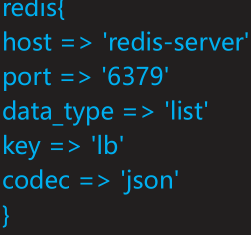
\[%{HTTPDATE:timestamp}\] \"%{WORD:verb}

最终版本

%{COMMONAPACHELOG} \"(?<referer>[^\"]+)\"

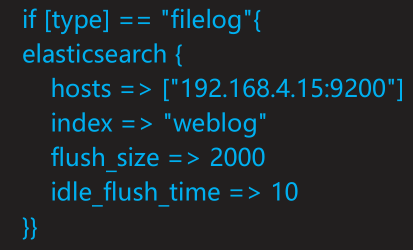
\"(?<UA>[^\"]+)\"

input redis插件



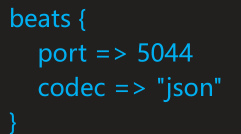
生产环境往往由redis来做缓冲，这里给出配置

output ES插件



调用成功后，把数据写入ES集群

input filebeats插件



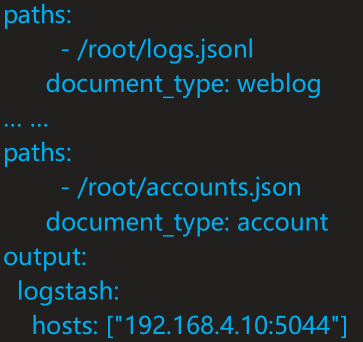
这个插件主要用来接收beats类软件发送过来的数据，有雨logstash依赖java环境，而且占用资源非常大，往往不希望所有集群的机器都部署Java环境安装logstash，而是用更轻量的filebeat代替

filebeat安装与配置

是用rpm安装filebeate

rpm -ivh filebeat-1.2.3-x86\_64.rpm

修改配置文件 /etc/filebeat/filebeat.yml



设置开机运行

systemctl enable filebeat

开启服务

systemctl start filebeat