**15Architecture\_04Kibana安装与使用+logstash配置扩展插件**

**一 Kibana**

**1.1 Kibana安装与配置**

**1.1.1 Kibana是什么**

数据可视化平台工具

**1.1.2 Kibana特点**

灵活的分析和可视化平台

实时总结流量和数据的图表

为不同的用户显示直观的界面

即时分享和嵌入的仪表板

**1.1.3 Kibana安装**

room9pc01 ~]# base-vm es56 192.168.1.56

es56 ~]# vim /etc/hosts

192.168.1.51 es51

192.168.1.52 es52

192.168.1.53 es53

192.168.1.54 es54

192.168.1.55 es55

192.168.1.56 es56

es56 ~]# vim /etc/yum.repos.d/local.repo

[elk]

name=elk

baseurl="ftp://192.168.1.254/elk"

enabled=1

gpgcheck=0

es56 ~]# yum -y install kibana

默认安装路径:/opt/kibina/

配置文件路径:/opt/kibana/config/kibana.yml

**1.1.4 修改配置文件/opt/kibana/config/kibana.yml**

es56 ~]# vim /opt/kibana/config/kibana.yml

2 server.port: 5601 #kibana服务监听端口号

5 server.host: "0.0.0.0" #kibana服务监听地址,监听所有地址

15 elasticsearch.url: "http://192.168.1.56:9200" #修改此处

23 kibana.index: ".kibana" #kibana默认数据库

26 kibana.defaultAppId: "discover" #kibana默认首页

53 elasticsearch.pingTimeout: 1500

57 elasticsearch.requestTimeout: 30000

64 elasticsearch.startupTimeout: 5000

**1.1.5 启动\开机自启kibana**

es56 ~]# systemctl restart\enable kibana

**1.1.6 端口检查及网络查看**

es56 ~]# ss -antulp | grep 5601

http://192.168.1.56:5601/

点击status查看状态,为Green

<http://192.168.1.51:9200/_plugin/head/>

查看是否有.kibana的索引信息,正常为有

浏览器访问kibana，如图-1所示：

[student@room9pc01 ~]$ firefox 192.168.1.56:5601

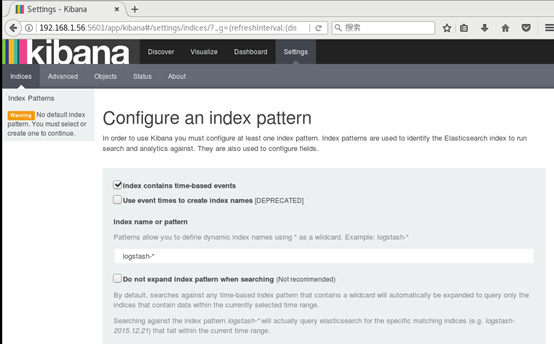


图-1

4）点击Status，查看是否安装成功，全部是绿色的对钩,说明安装成功，如图-2所示：

IMG_257

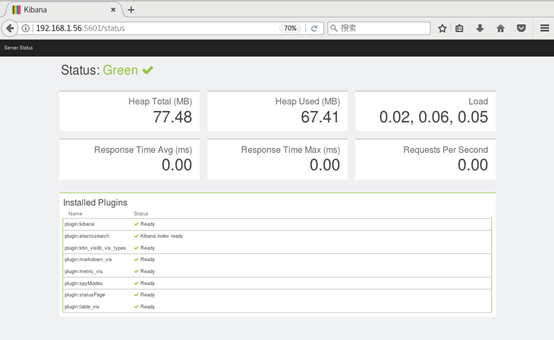


图-2

5）用head插件访问会有.kibana的索引信息，如图-3所示：

[student@room9pc01 ~]$ firefox http://192.168.1.55:9200/\_plugin/head

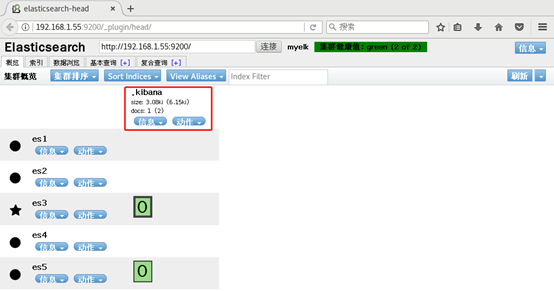


图-3

步骤二：使用kibana查看数据是否导入成功

1）数据导入以后查看logs是否导入成功，如图-4所示：

[student@room9pc01 ~]$ firefox http://192.168.1.55:9200/\_plugin/head

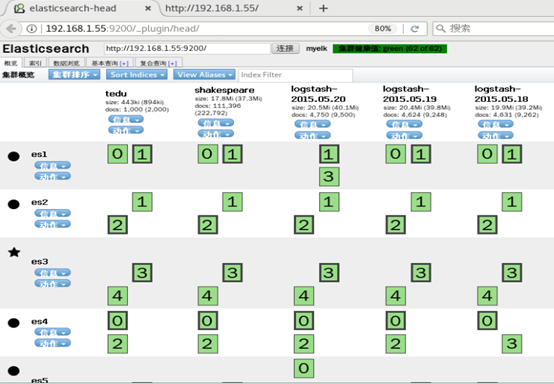


图-4

2）kibana导入数据，如图-5所示：

[student@room9pc01 ~]$ firefox http://192.168.1.56:5601

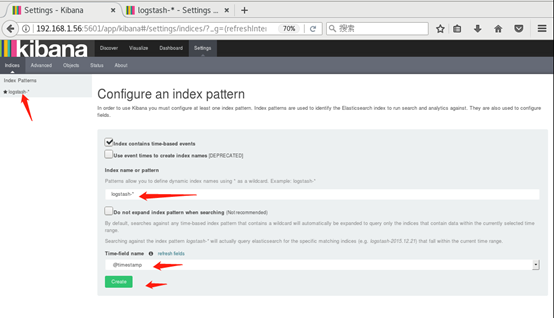


图-5

3）成功创建会有logstash-\*，如图-6所示：



图-6

4）导入成功之后选择Discover，如图-7所示：

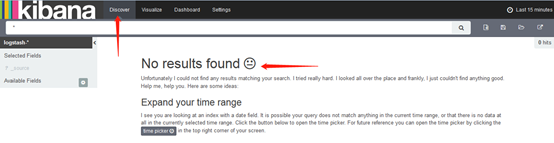


图-7

注意： 这里没有数据的原因是导入日志的时间段不对，默认配置是最近15分钟，在这可以修改一下时间来显示

5）kibana修改时间，选择Lsat 15 miuntes，如图-8所示：

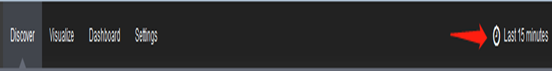


图-8

6）选择Absolute，如图-9所示：

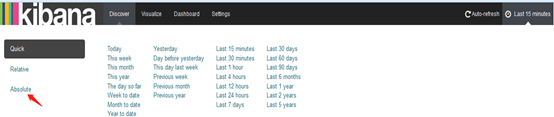


图-9

7）选择时间2015-5-15到2015-5-22，如图-10所示：

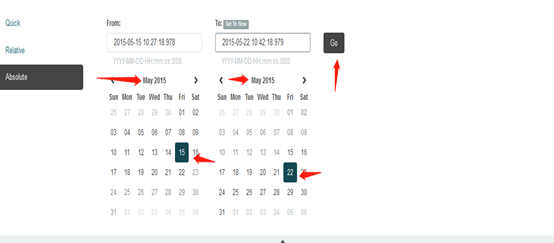


图-10

8）查看结果，如图-11所示：



图-11

9）除了柱状图，Kibana还支持很多种展示方式 ，如图-12所示：

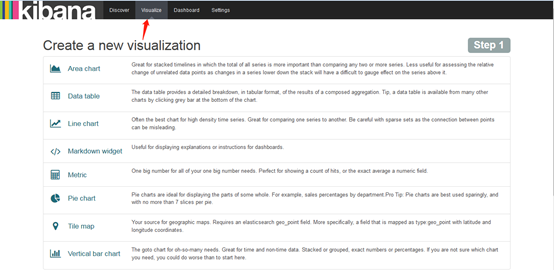


图-12

10）做一个饼图，选择Pie chart，如图-13所示：

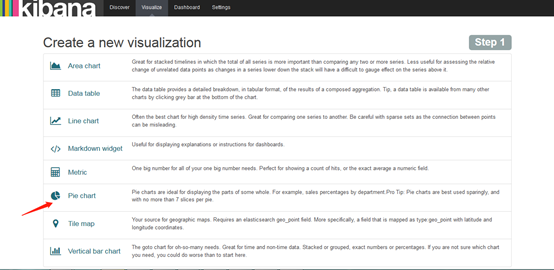


图-14

11）选择from a new serach，如图-11所示：

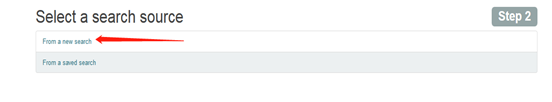


图-15

12）选择Spilt Slices，如图-16所示：

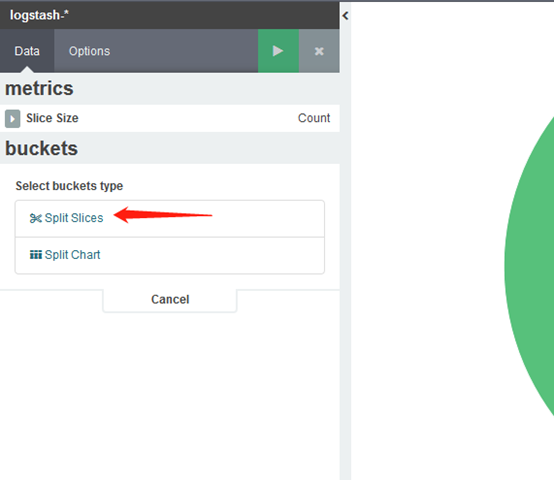


图-16

13）选择Trems,Memary(也可以选择其他的，这个不固定)，如图-17所示：

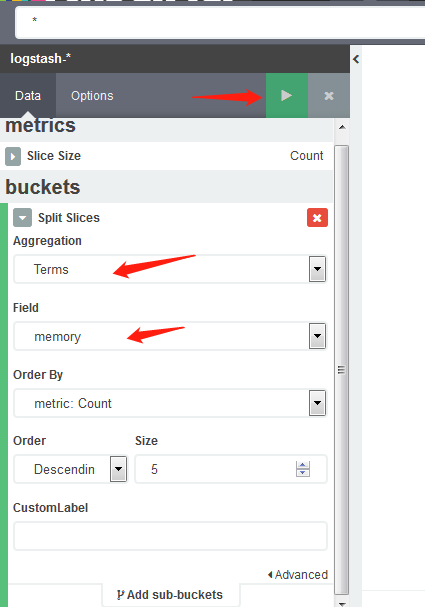


图-17

14）结果，如图-18所示：

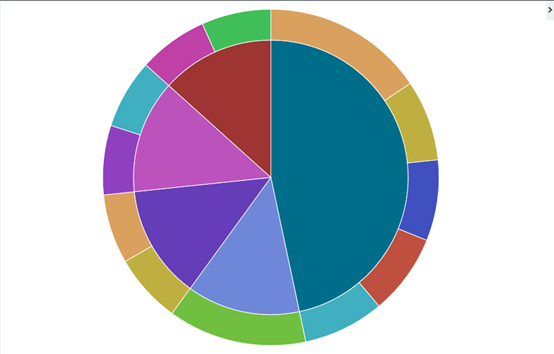


图-18

15）保存后可以在Dashboard查看，如图-19所示：

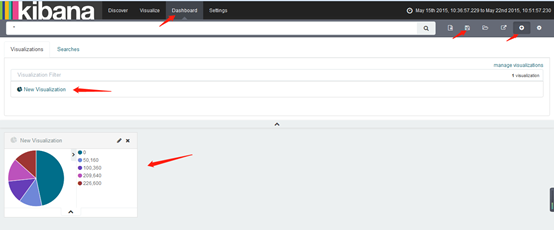


图-19

**二 logstash**

**2.1 logstash是什么**

是一个数据采集\加工处理及传输的工具

所有类型的书记集中处理

不同模式的格式数据的正常化

自定义日志格式的迅速扩展

为自定义数据源轻松添加插件

**2.2 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

room9pc01 ~]# base-vm es57 192.168.1.57

es51 ~]# scp /etc/yum.repos.d/local.repo

root@192.168.1.57:/etc/yum.repos.d/local.repo

es57 ~]# vim /etc/hosts

192.168.1.51 es51

192.168.1.52 es52

192.168.1.53 es53

192.168.1.54 es54

192.168.1.55 es55

192.168.1.56 es57

192.168.1.57 es58

es56 ~]# yum -y install java-1.8.0-openjdk

es57 ~]# yum -y install logstash

**logstash工作结构**

{数据源} ==>

input{ } ==>

filter{ } ==>

output{ } ==>

{ES}

logstash的第一个配置文件

es57 ~]# vim /etc/logstash/logstash.conf

input{ #标准输入,然后标准输出

stdin{}

}

filter{}

output{

stdout{}

}

es57 ~]# /opt/logstash/bin/logstash -h #查看帮助文档

es57 ~]# /opt/logstash/bin/logstash -f

/etc/logstash/logstash.conf #比较吃内存

Settings: Default pipeline workers: 2

Pipeline main started #主程序启动成功

abcdefghijklmn #输入

2019-10-16T03:04:16.951Z es57 abcdefghijklmn #输出

**logstash插件**

es57 ~]# /opt/logstash/bin/logstash-plugin #查看插件命令用法

es57 ~]# /opt/logstash/bin/logstash-plugin list #列出插件

logstash-插件名称-xxx

插件及文档地址: <http://github.com/logstash-plugins>

官方手册:

<https://www.elastic.co/guide/en/logstash/current/index.html>

es57 ~]# vim /etc/logstash/logstash.conf

input{

stdin{codec => "json"}

}

filter{}

output{

stdout{codec => "rubydebug"}

}

#json格式输入,rubydebug输出

es57 ~]# /opt/logstash/bin/logstash -f

/etc/logstash/logstash.conf

Settings: Default pipeline workers: 2

Pipeline main started

{"a":1,"b":2,"c":3} #输入

{ #输出

"a" => 1,

"b" => 2,

"c" => 3,

"@version" => "1",

"@timestamp" => "2019-10-16T03:48:18.078Z",

"host" => "es57"

}

**input插件**

es57 ~]# vim /etc/logstash/logstash.conf

input {

file {

path => ["/tmp/a.log", "/var/tmp/b.log"]

}

}

filter {

}

output {

stdout {

codec => "rubydebug"

}

}

es57 ~]# /opt/logstash/bin/logstash -f

/etc/logstash/logstash.conf #运行

另开终端

es57 ~]# echo A\_${RANDOM} >> /tmp/a.log

原终端查看输出结果

{

"message" => "A\_1280",

"@version" => "1",

"@timestamp" => "2019-10-16T06:22:26.631Z",

"path" => "/tmp/a.log",

"host" => "es57"

}

es57 ~]# echo A\_${RANDOM} >> /var/tmp/b.log

{

"message" => "A\_21973",

"@version" => "1",

"@timestamp" => "2019-10-16T06:22:34.643Z",

"path" => "/var/tmp/b.log",

"host" => "es57"

}

当原终端的logstash停止后,新终端新写入的数据在启动logstash后仍然可以被读出来,但上次已读出并显示的数据不再显示出来

logstash内有指针,指向最后一次读出的日志数据

input{

file{

path => ["/tmp/a.log", "/var/tmp/b.log"]

sincedb\_path => “/var/lib/sincedb” #指针文件地址

start\_position => “beginning” #没有指针文件时从日志开头读取

}

}

filter{

}

output{

stdout{

codec => "rubydebug"

}

}

input{

file{

path => ["/tmp/a.log"]

sincedb\_path => "/var/lib/sincedb"

start\_position => "beginning"

type => "nsd1906"

}

file{

path => ["/var/tmp/b.log"]

sincedb\_path => "/var/lib/sincedb"

start\_position => "beginning"

type => "tsd1906"

}

}

filter{

}

output{

stdout{

codec => "rubydebug"

}

}

**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,但使用性极广

几乎可以应用于各类型数据

room9pc01 ~]# base-vm es58 192.168.1.58

正则示例

/opt/logstash/vendor/bundle/jruby/1.9/gems/logstash-patterns-core-2.0.5/patterns/grok-patterns

input{

file{

path => ["/tmp/a.log"]

sincedb\_path => "/dev/null"

start\_position => "beginning"

type => "nsd1906"

}

}

filter{

grok{

match => {"message" => "%{COMBINEDAPACHELOG}"}

}

}

output{

stdout{

codec => "rubydebug"

}

}

input{

file{

path => ["/tmp/a.log"]

sincedb\_path => "/dev/null"

start\_position => "beginning"

type => "nsd1906"

}

}

filter{

grok{

match => {"message" => "%{COMBINEDAPACHELOG}"}

}

}

output{

stdout{codec => "rubydebug"}

elasticsearch{

hosts => ["es51:9200", "es52:9200", "es53:9200"]

index => "apache\_log"

flush\_size => 2000

idle\_flush\_time => 10 }

}