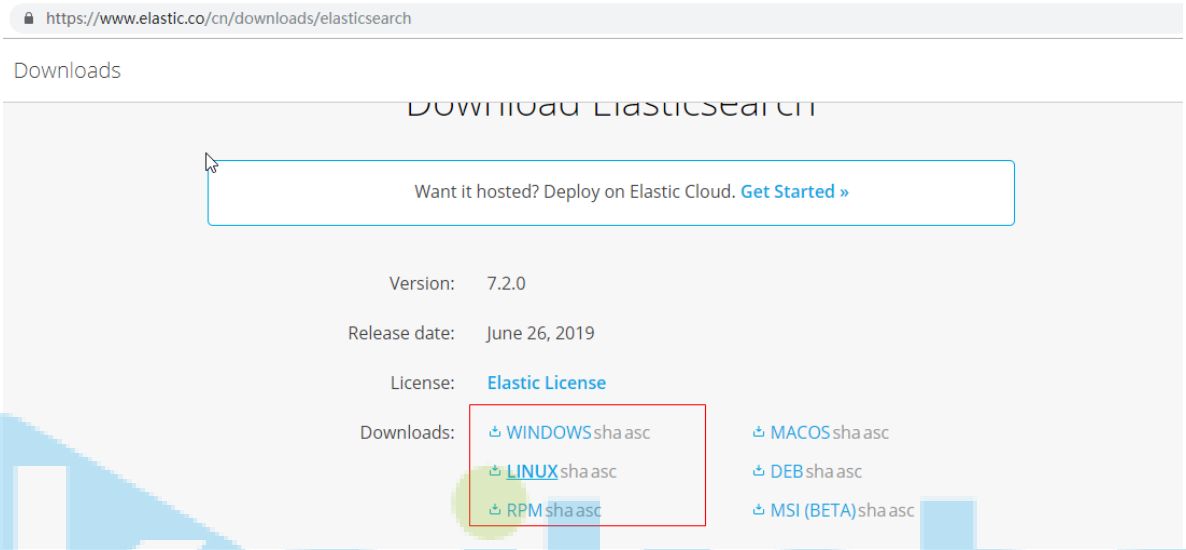
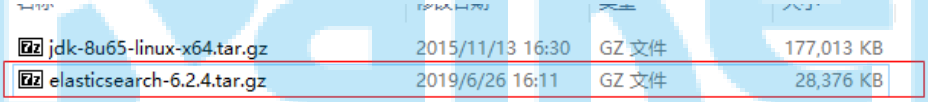
一、ES安装

1、下载ES

下载：（文件比较大，建议手动下载）



下载网址：



2、安装

解压：

```
tar -zxvf elasticsearch-6.2.4.tar.gz
```

注意：把elasticsearch软件必须放入/home/es（es是新建用户）的目录下，并把elasticsearch设置为es用户所属

创建日志、数据存储目录：（留作备用，初次先创建）

```
mkdir -p /data/logs/es
mkdir -p /data/es/{data,work,plugins,scripts}
```

创建用户

```
useradd es -s /bin/bash #es不能在root用户下启动，必须创建新的用户，用来启动es
```

启动：./elasticsearch

```
[root@jackhu bin]# ./elasticsearch
[2019-06-26T16:14:40,309][WARN ][o.e.b.ElasticsearchUncaughtExceptionHandler] [] uncaught exception in thread [main]
org.elasticsearch.bootstrap.StartupException: java.lang.RuntimeException: can not run elasticsearch as root
        at org.elasticsearch.bootstrap.Elasticsearch.init(Elasticsearch.java:125) ~[elasticsearch-6.2.4.jar:6.2.4]
        at org.elasticsearch.bootstrap.Elasticsearch.execute(Elasticsearch.java:112) ~[elasticsearch-6.2.4.jar:6.2.4]
        at org.elasticsearch.cli.EnvironmentAwareCommand.execute(EnvironmentAwareCommand.java:86) ~[elasticsearch-6.2.4
        at org.elasticsearch.cli.Command.mainWithoutErrorHandling(Command.java:124) ~[elasticsearch-cli-6.2.4.jar:6.2.4
        at org.elasticsearch.cli.Command.main(Command.java:90) ~[elasticsearch-cli-6.2.4.jar:6.2.4]
        at org.elasticsearch.bootstrap.Elasticsearch.main(Elasticsearch.java:92) ~[elasticsearch-6.2.4.jar:6.2.4]
        at org.elasticsearch.bootstrap.Elasticsearch.main(Elasticsearch.java:85) ~[elasticsearch-6.2.4.jar:6.2.4]
Caused by: java.lang.RuntimeException: can not run elasticsearch as root
        at org.elasticsearch.bootstrap.Bootstrap.initializeNatives(Bootstrap.java:105) ~[elasticsearch-6.2.4.jar:6.2.4]
        at org.elasticsearch.bootstrap.Bootstrap.setup(Bootstrap.java:172) ~[elasticsearch-6.2.4.jar:6.2.4]
        at org.elasticsearch.bootstrap.Bootstrap.init(Bootstrap.java:323) ~[elasticsearch-6.2.4.jar:6.2.4]
        at org.elasticsearch.bootstrap.Elasticsearch.init(Elasticsearch.java:121) ~[elasticsearch-6.2.4.jar:6.2.4]
```

注意：es不能在root用户下启动，必须创建新的用户，用来启动es

切换用户： su es

再次启动，发现还是报错，原因：当前用户没有执行权限

```
[es@jackhu bin]$ ./elasticsearch
Exception in thread "main" java.nio.file.AccessDeniedException: /home/es/elasticsearch-6.2.4/config/jvm.options
        at sun.nio.fs.UnixException.translateToIOException(UnixException.java:84)
        at sun.nio.fs.UnixException.rethrowAsIOException(UnixException.java:102)
        at sun.nio.fs.UnixException.rethrowAsIOException(UnixException.java:107)
        at sun.nio.fs.UnixFileSystemProvider.newByteChannel(UnixFileSystemProvider.java:214)
        at java.nio.file.Files.newByteChannel(Files.java:361)
        at java.nio.file.Files.newByteChannel(Files.java:407)
        at java.nio.file.spi.FileSystemProvider.newInputStream(FileSystemProvider.java:384)
        at java.nio.file.Files.newInputStream(Files.java:152)
        at org.elasticsearch.tools.launchers.JvmOptionsParser.main(JvmOptionsParser.java:58)
[es@jackhu bin]$ cd ..
```

授权：chown -R es:es elasticsearch-6.2.4

```
[es@jackhu ~]$ ll
总用量 28376
drwxr-xr-x. 8 es   es        143 4月  13 2018 elasticsearch-6.2.4
-rw-r--r--. 1 root root 29056810 6月  26 16:11 elasticsearch-6.2.4.tar.gz
[es@jackhu ~]$ cd elasticsearch-6.2.4
[es@jackhu elasticsearch-6.2.4]$ ll
总用量 224
drwxr-xr-x.  2 es es   4096 6月  26 16:13 bin
```

授权成功，发现elasticsearch已经在es用户下面了，可以启动了，但是启动成功，浏览器不能访问，因此还需要做如下配置：

配置修改：**

```
#
# Set the bind address to a specific IP (IPv4 or IPv6):
#
network.host: 0.0.0.0
#
# Set a custom port for HTTP:
#
```

再次启动：报如下错误

```
[2019-06-26T16:40:34,368][INFO ][o.e.n.Node               ] [4fZjnig] starting ...
[2019-06-26T16:40:35,232][INFO ][o.e.t.TransportService   ] [4fZjnig] publish_address {192.168.66.66:9300}, b
[2019-06-26T16:40:35,249][INFO ][o.e.b.BootstrapChecks    ] [4fZjnig] bound or publishing to a non-loopback a
ERROR: [3] bootstrap checks failed
[1]: max file descriptors [4096] for elasticsearch process is too low, increase to at least [65536]
[2]: max number of threads [3853] for user [es] is too low, increase to at least [4096]
[3]: max virtual memory areas vm.max_map_count [65530] is too low, increase to at least [262144]
[2019-06-26T16:40:35,273][INFO ][o.e.n.Node               ] [4fZjnig] stopping ...
[2019-06-26T16:40:35,300][INFO ][o.e.n.Node               ] [4fZjnig] stopped
```

1）max file descriptors [4096] for elasticsearch process is too low, increase to at least [65536]

每个进程最大同时打开文件数太小，可通过下面2个命令查看当前数量

```
ulimit -Hn
ulimit -Sn
```

```
[es@jackhu bin]$ ulimit -Hn
4096
[es@jackhu bin]$ ulimit -Hs
unlimited
[es@jackhu bin]$ ulimit -Sn
1024
```

修改/etc/security/limits.conf文件，增加配置，用户退出后重新登录生效

```
*               soft    nofile          65536
*               hard    nofile          65536
```

```
[root@jackhu bin]# ulimit -Hn
65536
[root@jackhu bin]# ulimit -Sn
65536
```

2）max number of threads [3818] for user [es] is too low, increase to at least [4096]

可通过命令查看

```
ulimit -Hu
ulimit -Su
```

```
[root@jackhu bin]# ulimit -Hu
3853
[root@jackhu bin]# ulimit -Su
3853
[root@jackhu bin]#
```

问题同上，最大线程个数太低。修改配置文件/etc/security/limits.conf，增加配置

```
`*              soft    nproc           4096`
`*              hard    nproc           4096`
```

```
[root@jackhu bin]# ulimit -Hu
4096
[root@jackhu bin]# ulimit -Su
4096
```

3）、max virtual memory areas vm.max_map_count [65530] is too low, increase to at least [262144]

修改/etc/sysctl.conf文件

```
vi /etc/sysctl.conf
sysctl -p #执行命令sysctl -p生效
#增加配置vm.max_map_count=262144
```

```
[root@jackhu bin]# vi /etc/sysctl.conf
# sysctl settings are defined through files in
vm.max_map_count=262144                          添加一行配置
# /usr/lib/sysctl.d/, /run/sysctl.d/, and /etc/sysctl.d/.
#
# Vendors settings live in /usr/lib/sysctl.d/.
# To override a whole file, create a new file with the same in
# /etc/sysctl.d/ and put new settings there. To override
# only specific settings, add a file with a lexically later
# name in /etc/sysctl.d/ and put new settings there.
#
# For more information, see sysctl.conf(5) and sysctl.d(5).
"/etc/sysctl.conf" 11L, 473C written
[root@jackhu bin]# sysctl -p          执行此命令生效
vm.max_map_count = 262144
[root@jackhu bin]#
```

错误解决完毕：重新启动

不安全 | 192.168.66.66:9200


```
{
  "name" : "4fZjnig",
  "cluster_name" : "elasticsearch",
  "cluster_uuid" : "amdhilZaSm6tgynYGfLGdQ",
  "version" : {
    "number" : "6.2.4",
    "build_hash" : "ccec39f",
    "build_date" : "2018-04-12T20:37:28.497551Z",
    "build_snapshot" : false,
    "lucene_version" : "7.2.1",
    "minimum_wire_compatibility_version" : "5.6.0",
    "minimum_index_compatibility_version" : "5.0.0"
  },
  "tagline" : "You Know, for Search"
}
```

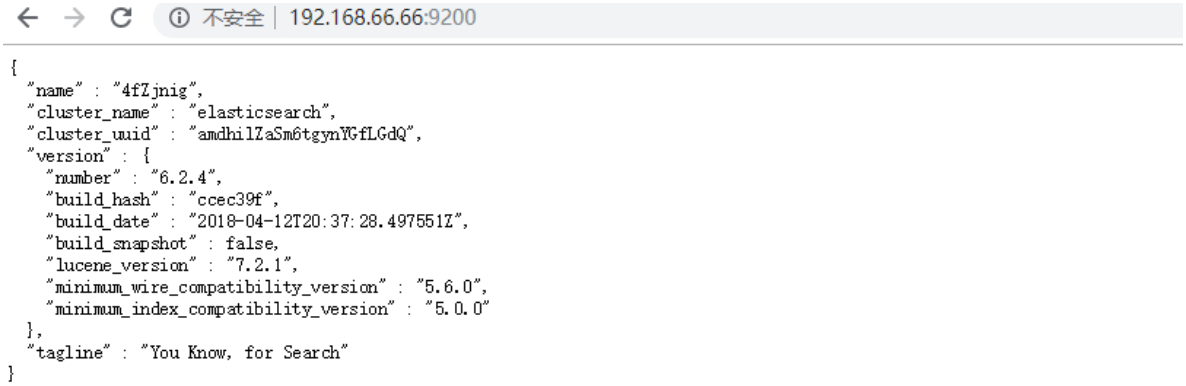

后台启动：

```
./elasticsearch -d
```

3、容器安装

```
#搜索镜像
docker search elasticsearch
#拉取镜像
docker pull elasticsearch:6.2.4
#创建容器
docker create --name elasticsearch --net host -e "discovery.type=single-node" -e
"network.host=192.168.66.66" elasticsearch:6.2.4
#启动
docker start elasticsearch
#查看日志
docker logs elasticsearch
```

访问容器elasticsearch:



后台

二、head插件安装

1、head插件主要用途

elasticsearch-head是一个用来浏览、与Elastic Search簇进行交互的web前端展示插件。elasticsearch-head是一个用来监控Elastic Search状态的客户端插件。

elasticsearch主要有以下三个主要操作——1）簇浏览，显示簇的拓扑并允许你执行索引（index）和节点层面的操作。 2）查询接口，允许你查询簇并以原始json格式或表格的形式显示检索结果。 3）显示簇状态，有许多快速访问的tabs用来显示簇的状态。 4）支持Restful API接口，包含了许多选项产生感兴趣的结果，包括： 第一，请求方式:get,put,post,delete; json请求数据，节点node， 路径path。 第二，JSON验证器。 第三，定时请求的能力。 第四，用javascript表达式传输结果的能力。 第五，统计一段时间的结果或该段时间结果比对的能力。

第六，以简单图标的形式绘制传输结果

2、安装

安装步骤:

```
#下载nodejs,head插件运行依赖node
wget https://nodejs.org/dist/v9.9.0/node-v9.9.0-linux-x64.tar.xz
#解压
tar -xf node-v9.9.0-linux-x64.tar.xz
#重命名
mv node-v9.9.0-linux-x64 nodeJs
#配置文件
vim /etc/profile
#刷新配置
source /etc/profile
#查询node版本，同时查看是否安装成功
```

```
node -v
#下载head插件
wget  https://github.com/mobz/elasticsearch-head/archive/master.zip
#解压
unzip master.zip
#使用淘宝的镜像库进行下载，速度很快
npm install -g cnpm --registry=https://registry.npm.taobao.org
#进入head插件解压目录，执行安装命令
cnpm install
```

3、运行

```
npm start #启动head插件
```

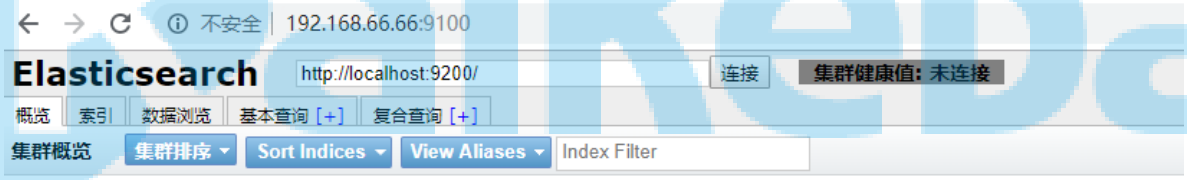
启动运行端口为：9100

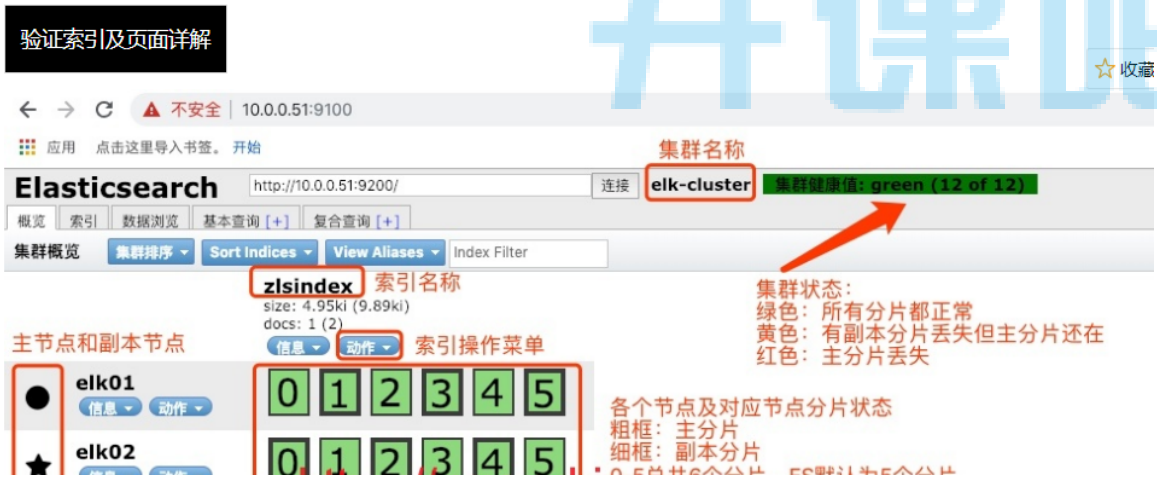
```
[root@jackhu elasticsearch-head-master]# npm start

> elasticsearch-head@0.0.0 start /opt/elasticsearch-head-master
> grunt server

(node:3946) ExperimentalWarning: The http2 module is an experimental API.
Running "connect:server" (connect) task
Waiting forever...
Started connect web server on http://localhost:9100
```

访问：





此时未连接，需要配置才能连接：

修改 Gruntfile.js文件：

```
[root@jackhu elasticsearch-head-master]# ll
总用量 228
-rw-r--r--.   1 root root    248 6月   25 21:18 Dockerfile
-rw-r--r--.   1 root root    221 6月   25 21:18 Dockerfile-alpine
-rw-r--r--.   1 root root    104 6月   25 21:18 elasticsearch-head.sublime-project
-rw-r--r--.   1 root root   2171 6月   25 21:18 Gruntfile.js
-rw-r--r--.   1 root root   3482 6月   25 21:18 grunt_fileSets.js
-rw-r--r--.   1 root root   1100 6月   25 21:18 index.html
-rw-r--r--.   1 root root    559 6月   25 21:18 LICENCE
drwxr-xr-x. 376 root root  12288 6月   26 18:43 node_modules
-rw-r--r--.   1 root root    886 6月   25 21:18 package.json
-rw-r--r--.   1 root root 169953 6月   26 18:40 package-lock.json
-rw-r--r--.   1 root root    100 6月   25 21:18 plugin-descriptor.properties
drwxr-xr-x.   4 root root     53 6月   25 21:18 proxy
-rw-r--r--.   1 root root   7034 6月   25 21:18 README.textile
drwxr-xr-x.   5 root root    140 6月   25 21:18 _site
drwxr-xr-x.   4 root root     31 6月   25 21:18 src
drwxr-xr-x.   4 root root     70 6月   25 21:18 test
```

修改如下：

```
            connect: {
                    server: {
                            options: {
                                    port: 9100,
                                    base: '.',
                                    keepalive: true,
                                    hostname:'*'
                            }
                    }
```

修改_site/app.js

修改IP地址，连接elasticsearch

```
app.App = ui.AbstractWidget.extend({
        defaults: {
                base_uri: null
        },
        init: function(parent) {
                this._super();
                this.prefs = services.Preferences.instance();
                this.base_uri = this.config.base_uri || this.prefs.get("app-base_uri") || "http://192.168.66.66:9200";
                if( this.base_uri.charAt( this.base_uri.length - 1 ) !== "/" ) {
                        // XHR request fails if the URL is not ending with a "/"
                        this.base_uri += "/";
                }
```

启用CORS:

当head插件访问es时，您必须在elasticsearch中启用CORS，否则您的浏览器将拒绝跨域。

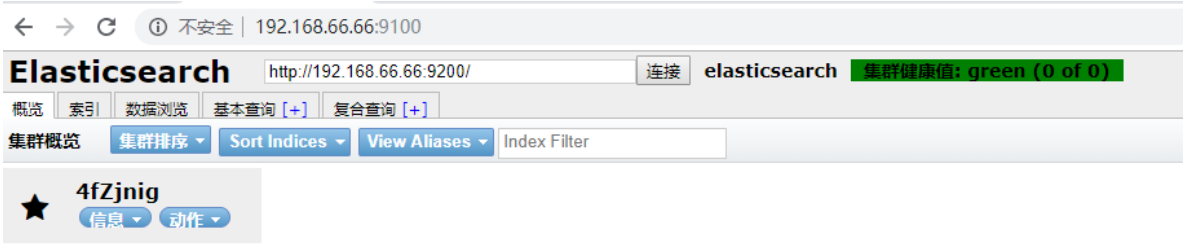
在elasticsearch配置中：

```
http.cors.enabled: true
```

您还必须设置，http.cors.allow-origin因为默认情况下不允许跨域。http.cors.allow-origin: "*" 是允许配置的，但由于这样配置的任何地方都可以访问，所以有安全风险。 我在集群安装的时候已经配好了、如果你刚配置、需要重启ElasticSearch服务

```
http.cors.enabled: true
http.cors.allow-origin: "*"
```

访问head**插件**



三、LogStash安装

1、LogStash插件介绍

Logstash是一个具有实时管道的开源数据收集引擎。可以动态地统一不同来源的数据，并将数据归到不同目的地。也是一个管理事件和日志工具。你可以用它来收集日志，分析它们，并将它们储存起来以供以后使用。

Logstash 通常都是和 Kibana 以及 Elasticsearch 一起使用。

2、logStash安装

```
#注意版本和elasticsearch,kibana 必须保持一致，es,kibana都是6.2.4版本
wget https://artifacts.elastic.co/downloads/logstash/logstash-6.2.4.tar.gz
#解压
tar -zxvf logstash-6.2.4.tar.gz
#启动 基本的 intput output
#stdin stdout 输入输出插件
./logstash -e 'input{ stdin{} } output{ stdout{} }'

# codec
./logstash -e 'input{ stdin{} } output{ stdout{ codec => json } }'

#日志内容写入elasticsearch
./logstash -e 'input{ stdin{} } output{ elasticsearch{hosts =>
["192.168.66.66:9200"]} }'

#日志内容写入elasticsearch，同时输出
#注意elasticsearch插件的语法格式：hosts 对应数组
./logstash -e 'input{ stdin{} } output{ elasticsearch{hosts =>
["192.168.66.66:9200"]} stdout{} }'
```

3、logStash插件

3.1、input插件

Input plugins

edit

An input plugin enables a specific source of events to be read by Logstash.

The following input plugins are available below. For a list of Elastic supported plugins, please consult the Support Matrix.

stdin	Reads events from standard input	logstash-input-stdin
file	Streams events from files	logstash-input-file
http	Receives events over HTTP or HTTPS	logstash-input-http
tcp	Reads events from a TCP socket	logstash-input-tcp

输入比较常见的几个插件：stdin、file、http、tcp

3.2、output插件

Output plugins

edit

An output plugin sends event data to a particular destination. Outputs are the final stage in the event pipeline.

The following output plugins are available below. For a list of Elastic supported plugins, please consult the Support Matrix.

stdout	Prints events to the standard output	logstash-output-stdout
file	Writes events to files on disk	logstash-output-file
http	Sends events to a generic HTTP or HTTPS endpoint	logstash-output-http
tcp	Writes events over a TCP socket	logstash-output-tcp

把日志内容输出到elasticsearch插件：

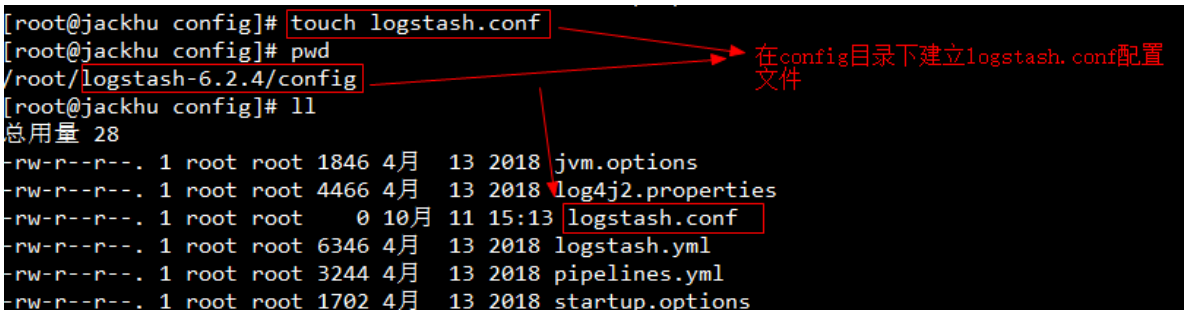
elasticsearch	Stores logs in Elasticsearch	logstash-output-elasticsearch

3.3、codec插件

Codec（Code Decode）Plugin作用于input和output plugin，负责将数据在原始与Logstash之间转换，常见的codec有： **plain** 读取原始内容 **dots** 将内容简化为点进行输出 **rubydebug** 将内容按照ruby格式输出，方便调试 **line** 处理带有换行符的内容 **json** 处理json格式的内容 **multiline** 处理多行数据的内容

4、logStash配置

4.1、创建配置



4.2、配置语法

Configuring Logstash

edit

To configure Logstash, you create a config file that specifies which plugins you want to use and settings for each plugin. You can reference event fields in a configuration and use conditionals to process events when they meet certain criteria. When you run logstash, you use the `-f` to specify your config file.

Let's step through creating a simple config file and using it to run Logstash. Create a file named "logstash-simple.conf" and save it in the same directory as Logstash.

```
input { stdin { } }
output {
  elasticsearch { hosts => ["localhost:9200"] }
  stdout { codec => rubydebug }
}
```

开始配置:

```
input { stdin { } }
output {
  elasticsearch { hosts => ["192.168.66.66:9200"] }
  stdout { codec => rubydebug }
}

#启动命令
bin/logstash -f config/logstash.conf
```

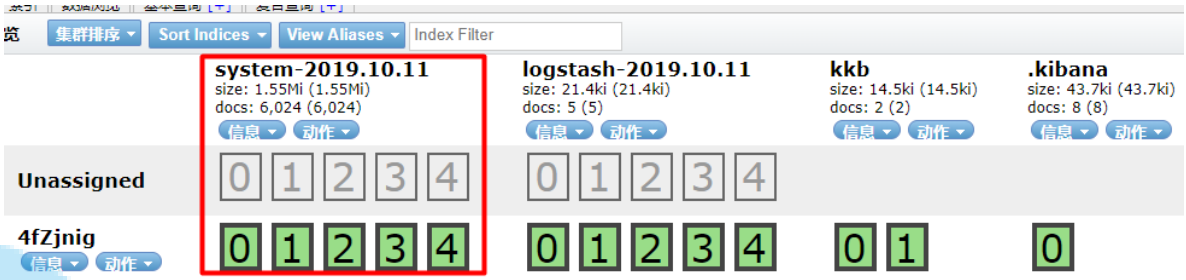
5、file日志收集

```
#建立新的配置文件
mv logstash.conf file.conf
#详细配置如下
input {
        file{
           path => "/var/log/messages" #收集messages文件日志
           type => "system"
           start_position => "beginning" #记录上次收集的位置
```

```
        }
 }
output {
  elasticsearch {
        hosts => ["192.168.66.66:9200"] #写入elasticsearch的地址
        index => "system-%{+YYYY.MM.dd}" #定义索引的名称
         }
  stdout { codec => rubydebug }
}

#启动logstash,配置文件名字叫什么无所谓
bin/logstash -f config/file.conf
```

收集效果如下所示：



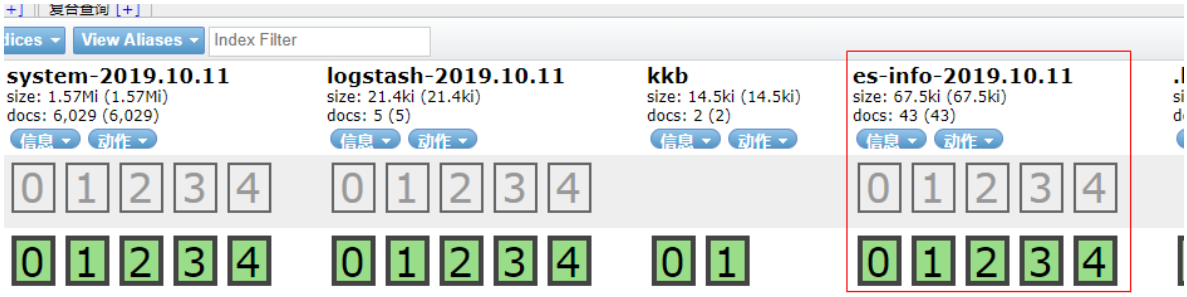
6、Java日志收集

```
#在原来file文件的基础上进行编辑
input {
        file{
           path => "/var/log/messages"
           type => "system"
           start_position => "beginning"
        }
        #加一个file文件收集日志插件，收集elasticsearch日志、es就是java语言开发的。
        file{
           path => "/home/es/elasticsearch-6.2.4/logs/elasticsearch.log"
           type => "es-info"
           start_position => "beginning"
        }
 }
output {

  if [type] == "system"{
  elasticsearch {
        hosts => ["192.168.66.66:9200"]
        index => "system-%{+YYYY.MM.dd}"
         }
  }
  #判断，导入到不同的索引库，否则会放入同一个索引库中
  if [type] == "es-info"{
  elasticsearch {
        hosts => ["192.168.66.66:9200"]
        index => "es-info-%{+YYYY.MM.dd}"
         }
  }
```

```
  stdout { codec => rubydebug }
}
```

导入效果：



问题：目前导入日志都是按照行导入的、但是有些日志多行是一句话，如果分开的话，就不太容查看日志的完整的意思了。

```
#下面日志是一个日志，如果按照行收集就不符合整个日志的日志，会把整体的日志给分开
Exception in thread "main"
org.springframework.jmx.access.InvalidInvocationException: bean:name=boy
at
org.springframework.jmx.access.MBeanClientInterceptor.invoke(MBeanClientIntercep
tor.java:358)
at
org.springframework.aop.framework.ReflectiveMethodInvocation.proceed(ReflectiveM
ethodInvocation.java:171)
at
org.springframework.aop.framework.JdkDynamicAopProxy.invoke(JdkDynamicAopProxy.j
ava:204)
at $Proxy0.sayHello(Unknown Source)
at com.ebupt.jmxTest.Client.main(Client.java:13)
```

解决方案： 可以使用codec来进行解决，codec把多行日志

```
#在原来file文件的基础上进行编辑
input {
        file{
           path => "/var/log/messages"
           type => "system"
           start_position => "beginning"
        }
        #加一个file文件收集日志插件，收集elasticsearch日志、es就是java语言开发的。
        file{
           path => "/home/es/elasticsearch-6.2.4/logs/elasticsearch.log"
           type => "es-info"
           start_position => "beginning"
           #使用正则表达式，合并多行日志
           codec => multiline {
                   pattern => "^\[" #发现中括号，就合并日志
                   negate => true
                   what => "previous"
                 }
        }
 }
output {
```

```
  if [type] == "system"{
  elasticsearch {
        hosts => ["192.168.66.66:9200"]
        index => "system-%{+YYYY.MM.dd}"
         }
  }
  #判断，导入到不同的索引库，否则会放入同一个索引库中
  if [type] == "es-info"{
  elasticsearch {
        hosts => ["192.168.66.66:9200"]
        index => "es-info-%{+YYYY.MM.dd}"
         }
  }

  stdout { codec => rubydebug }
}
```

使用codec插件处理多行信息。把多行日志合并为一行，导入到es

四、Kibana安装

1、kibana 插件介绍

kibana 插件提供了Marvel监控的UI界面。 kibana是一个与elasticsearch一起工作的开源的分析和可视化的平台。使用kibana可以查询、查看并与存储在elasticsearch索引的数据进行交互操作。使用kibana能执行高级的数据分析，并能以图表、表格和地图的形式查看数据。 kibana使得理解大容量的数据变得非常容易。它非常简单，基于浏览器的接口使我们能够快速的创建和分享显示elasticsearch查询结果实时变化的仪表盘。

2、kibana下载

下载命令：

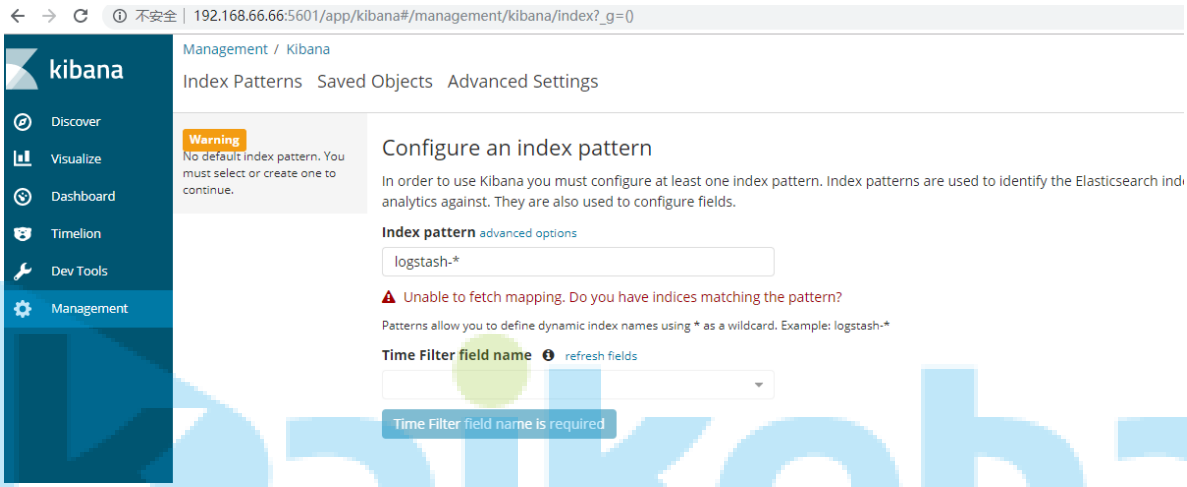
```
#kibana版本必须和es版本一致，否则监控不到es
wget https://artifacts.elastic.co/downloads/kibana/kibana-6.2.4-linux-
x86_64.tar.gz
#将shasum生产的SHA与已发布的SHA进行比较。
shasum -a 512 kibana-6.4.2-linux-x86_64.tar.gz
tar -xzf kibana-6.4.2-linux-x86_64.tar.gz
# 归档包解压的目录为$KIBANA_HOME
cd kibana-6.4.2-linux-x86_64/
```

3、环境配置

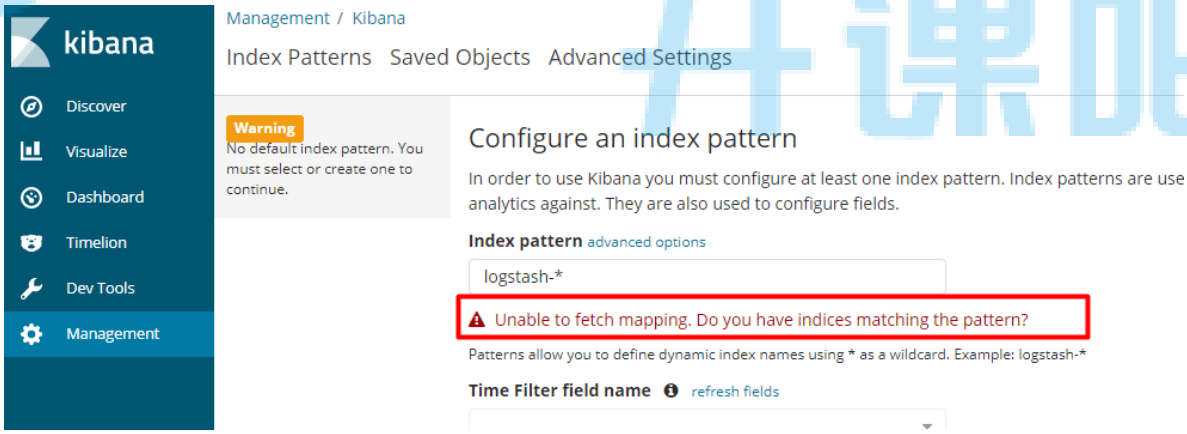
详细配置地址：https://www.elastic.co/guide/en/kibana/5.6/settings.html

```
# 将默认配置改成如下：
server.port: 5601
server.host: "192.168.66.66"
#修改成自己集群的端口号及IP
elasticsearch.url: "http://192.168.66.66:9200"
kibana.index: ".kibana"
```

4、安装成功

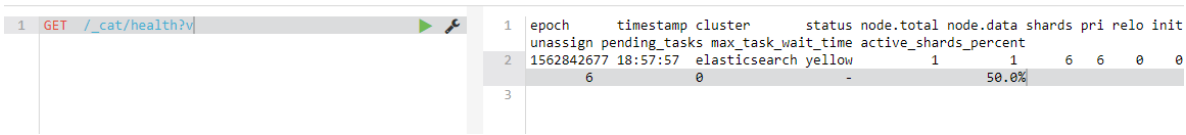


##5、kibana使用



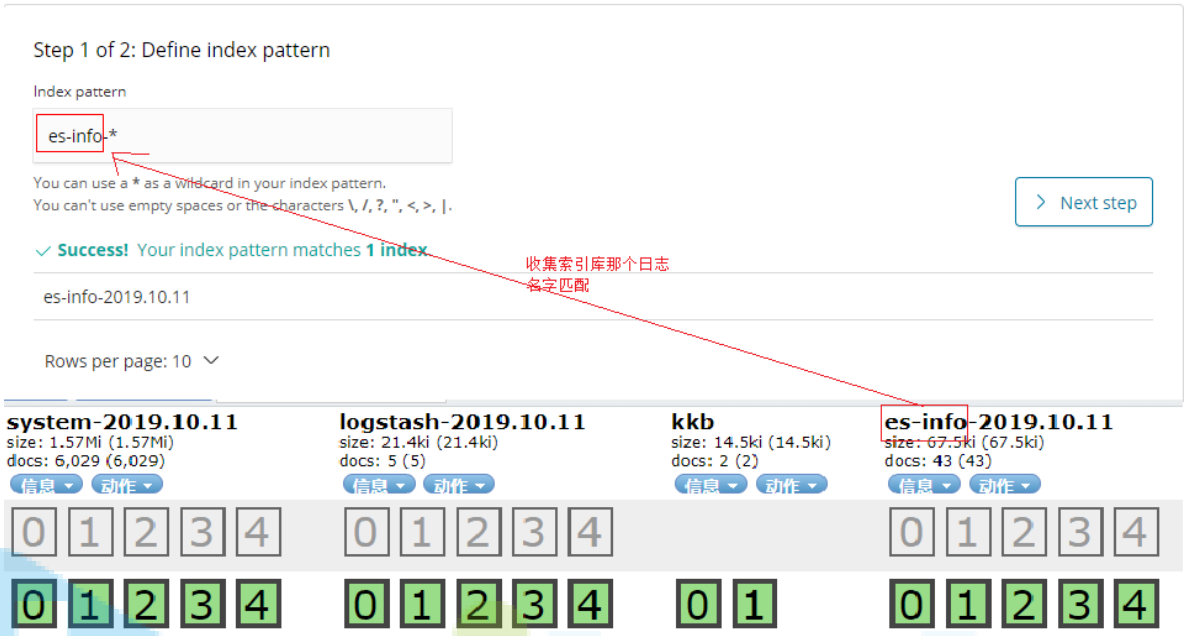
可以看到这里有个红色的错误：“Unable to fetch mapping. Do you have indices matching the pattern?”，这个是因为在elasticsearch中还没有任何有关logstash-*格式相关的数据，所以这里才提示报错，可以暂时忽略。

初次使用我们可以进行一些简单的测试，比如查看elasticsearch集群的状态，在Kibana控制台中运行命令GET /_cat/health?v，得到如下所示：

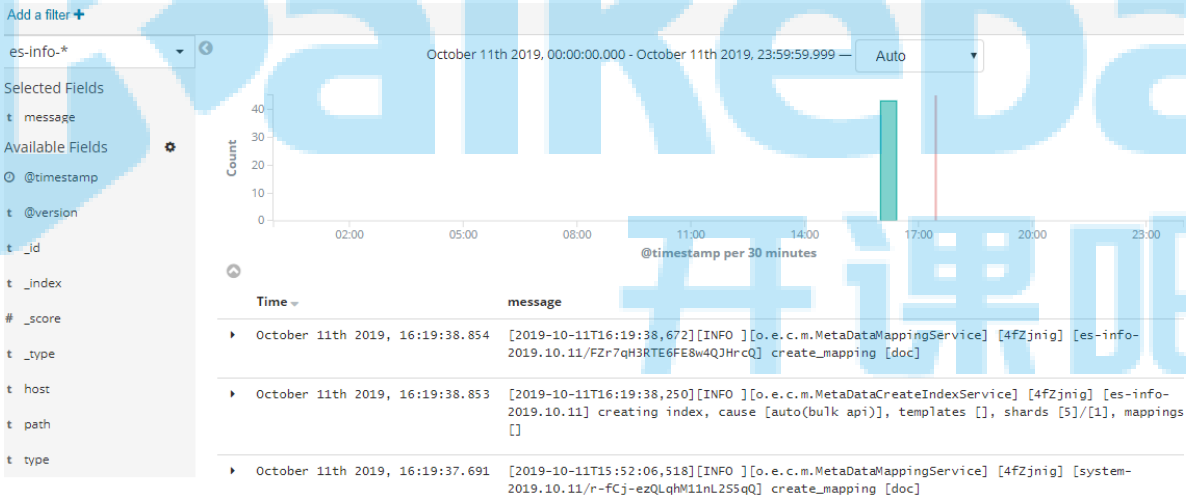


6、日志视图

1）创建日志



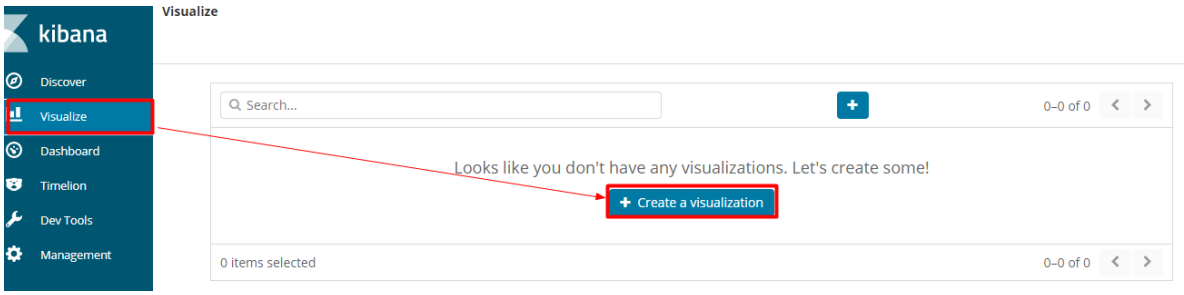
可以发现，日志可视化展示以及成功：



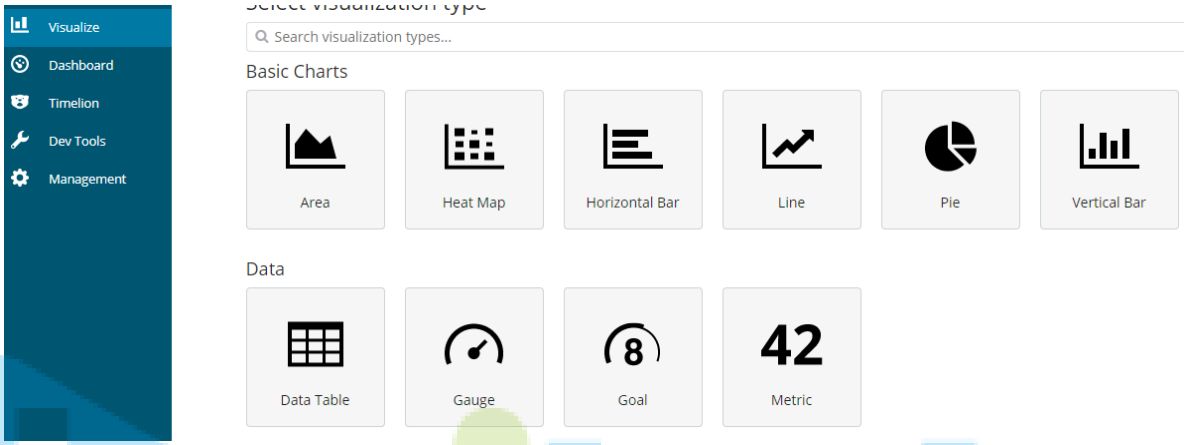
7、可视化数据

在Visualize应用程序中，你可以使用各种图表、表格和地图等来塑造数据，你将创建四个可视化效果：饼图、柱状图、坐标图和Markdown小部件。

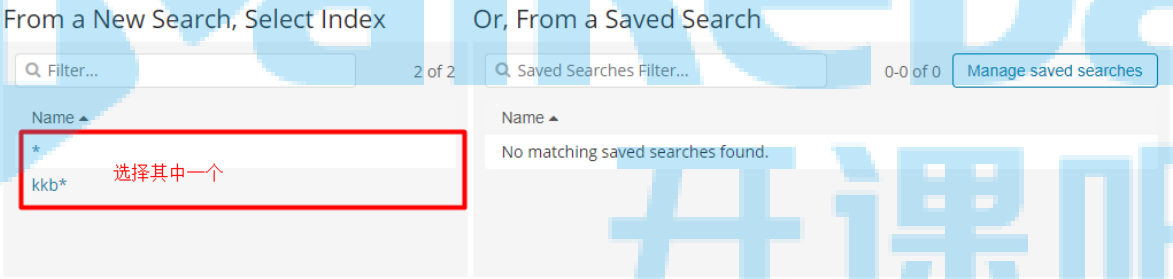
1）点击**Visualize**

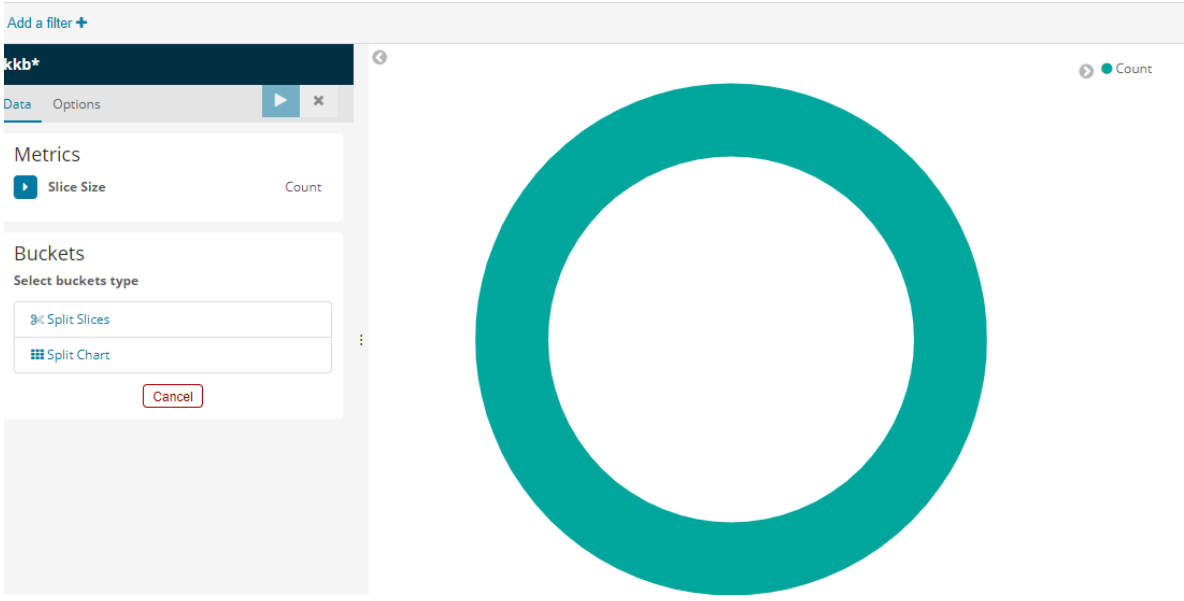


2）点击相应图标

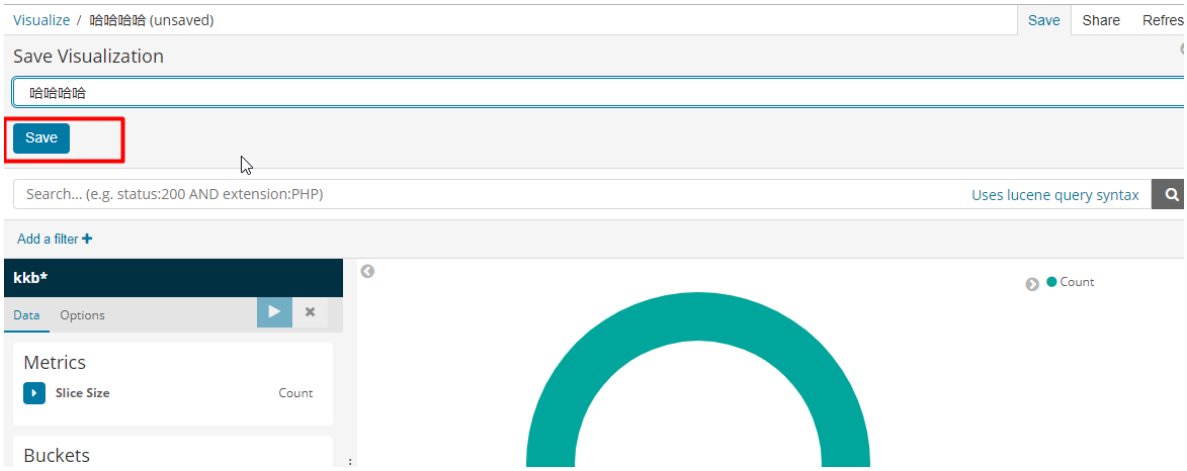


3）选择new search

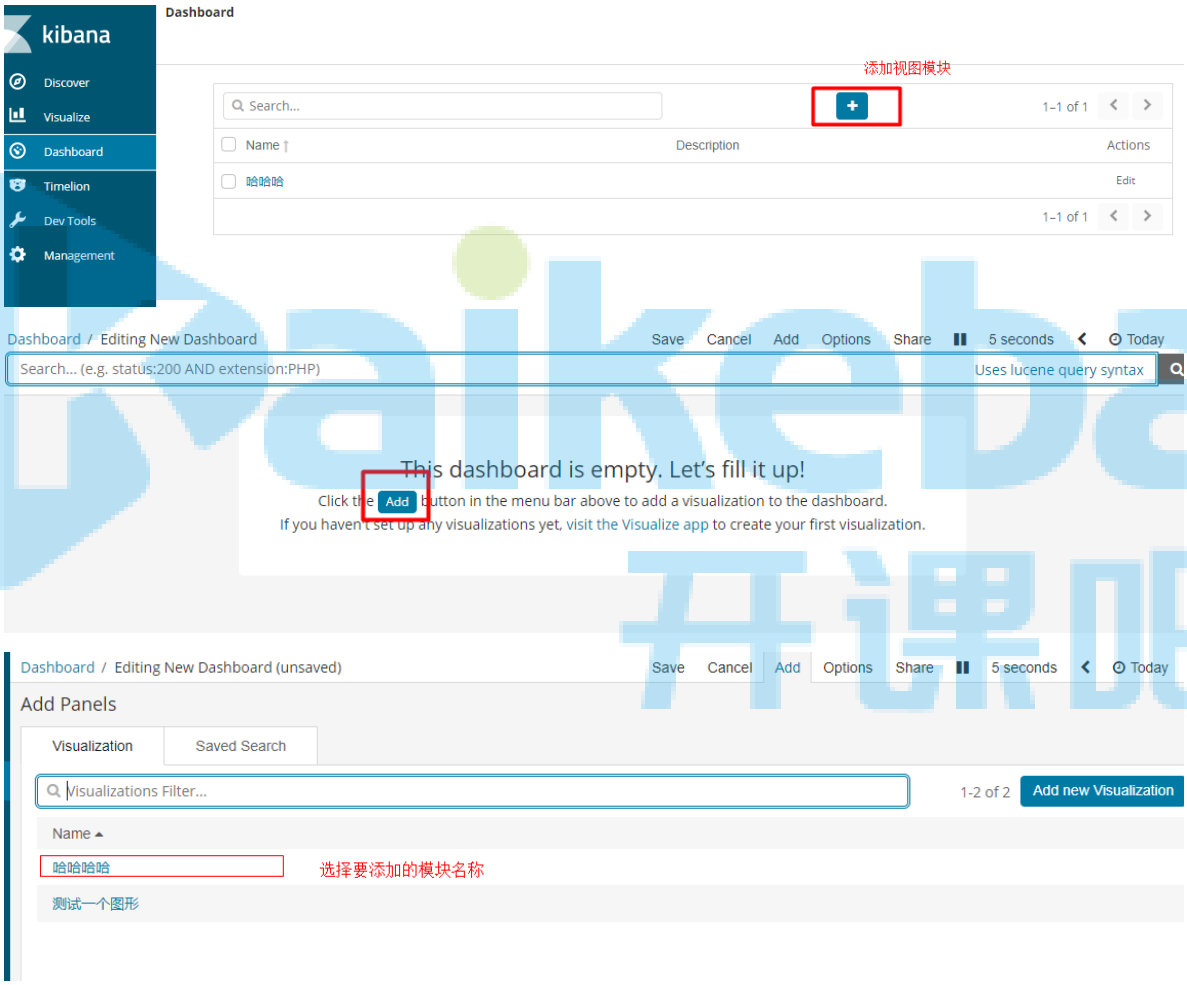




4）保存



5）添加视图模块



五、nginx日志

1、安装nginx（参考安装资料）

2、日志格式JSON

2.1、文本格式（默认格式）

开启日志：把日志写入到日志文件

```
server {
    listen       80;
    server_name  localhost;

    #charset koi8-r;

    access_log  logs/host.access.log ;

    location / {
```

开启普通文本日志，前提条件是logs文件夹必须存在

日志内容如下所示：

```
-rw-r--r--. 1 root root 2810 10月  12 15:15 host.access.log
[root@jackhu logs]# cat host.access.log
192.168.66.1 - - [12/Oct/2019:15:12:07 +0800] "GET / HTTP/1.1" 304 0 "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, lik
Gecko) Chrome/77.0.3865.90 Safari/537.36"
192.168.66.1 - - [12/Oct/2019:15:12:07 +0800] "GET / HTTP/1.1" 304 0 "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, lik
Gecko) Chrome/77.0.3865.90 Safari/537.36"
192.168.66.1 - - [12/Oct/2019:15:12:07 +0800] "GET / HTTP/1.1" 304 0 "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, lik
Gecko) Chrome/77.0.3865.90 Safari/537.36"
192.168.66.1 - - [12/Oct/2019:15:12:08 +0800] "GET / HTTP/1.1" 304 0 "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, lik
Gecko) Chrome/77.0.3865.90 Safari/537.36"
192.168.66.1 - - [12/Oct/2019:15:12:08 +0800] "GET / HTTP/1.1" 304 0 "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, lik
Gecko) Chrome/77.0.3865.90 Safari/537.36"
```

发现日志都是一行普通文本格式的日志。这样不利于logstash收集整理，是否能把这些日志直接变成json格式呢。当然是可以的。

2.2、JSON格式

日志格式转换配置如下所示：

```
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"'
' }';
```

定义一个名称为log_json的json格式输出格式化类

输出日志引用此格式输出日志：

```
server {
    listen       80;
    server_name  localhost;

    #charset koi8-r;

    access_log  logs/host.access.log log_json;

    location / {
        root   html;
```

此名称和json格式日志格式化名称相同

日志输出效果如下所示：

```
{ "@timestamp": "12/Oct/2019:15:15:34 +0800", "remote_addr": "192.168.66.1", "referer": "-", "request": "GET / HTTP/1.1", "status": 304, "bytes"
 "agent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/77.0.3865.90 Safari/537.36", "x_forwarded": "
"up_addr": "-","up_host": "-","up_resp_time": "-","request_time": "0.000" }
{ "@timestamp": "12/Oct/2019:15:15:35 +0800", "remote_addr": "192.168.66.1", "referer": "-", "request": "GET / HTTP/1.1", "status": 304, "bytes"
 "agent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/77.0.3865.90 Safari/537.36", "x_forwarded": "
"up_addr": "-","up_host": "-","up_resp_time": "-","request_time": "0.000" }
{ "@timestamp": "12/Oct/2019:15:15:35 +0800", "remote_addr": "192.168.66.1", "referer": "-", "request": "GET / HTTP/1.1", "status": 304, "bytes"
 "agent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/77.0.3865.90 Safari/537.36", "x_forwarded": "
"up_addr": "-","up_host": "-","up_resp_time": "-","request_time": "0.000" }
{ "@timestamp": "12/Oct/2019:15:15:35 +0800", "remote_addr": "192.168.66.1", "referer": "-", "request": "GET / HTTP/1.1", "status": 304, "bytes"
 "agent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/77.0.3865.90 Safari/537.36", "x_forwarded": "
"up_addr": "-","up_host": "-","up_resp_time": "-","request_time": "0.000" }
{ "@timestamp": "12/Oct/2019:15:15:35 +0800", "remote_addr": "192.168.66.1", "referer": "-", "request": "GET / HTTP/1.1", "status": 304, "bytes"
 "agent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/77.0.3865.90 Safari/537.36", "x_forwarded": "
"up_addr": "-","up_host": "-","up_resp_time": "-","request_time": "0.000" }
```


发现日志已经完成变成json格式了。此处有掌声..............

3、logstash收集

3.1、配置文件

```
#把日志文件日志以json格式输入
input {
        file{
              path => "/usr/local/src/logs/host.access.log"
              codec => json
            }


      }
#在logstash 控制台看一下输出日志的格式
output {
  stdout { codec => rubydebug }
}
```

3.2、导入到es

```
#输入插件
input {
        file{
           path => "/var/log/messages"
           type => "system"
           start_position => "beginning"
        }

        file{
           path => "/home/es/elasticsearch-6.2.4/logs/elasticsearch.log"
           type => "es-info"
           start_position => "beginning"
        }

        #收集nginx日志
        file{
           path => "/usr/local/src/logs/host.access.log"
           type => "nginx-log"
           start_position => "beginning"
           codec => json
        }
 }
```

```
#输出插件
output {

  if [type] == "system"{
  elasticsearch {
        hosts => ["192.168.66.66:9200"]
        index => "system-%{+YYYY.MM.dd}"
         }
  }

 if [type] == "es-info"{
  elasticsearch {
        hosts => ["192.168.66.66:9200"]
        index => "es-info-%{+YYYY.MM.dd}"
         }
  }

 #添加nginx日志
 if [type] == "nginx-log"{
  elasticsearch {
        hosts => ["192.168.66.66:9200"]
        index => "nginx-log-%{+YYYY.MM.dd}"
         }
  }

  #同时在控制台输出
  stdout { codec => rubydebug }
}
```

六、项目日志