

ELK 实时日志分析平台环境部署--完整记录

在日常运维工作中，对于系统和业务日志的处理尤为重要。今天，在这里分享一下自己部署的 ELK (+Redis) - 开源实时日志分析平台的记录过程（仅依据本人的实际操作为例说明，如有误述，敬请指出）~

=====概念介绍=====

==

日志主要包括系统日志、应用程序日志和安全日志。系统运维和开发人员可以通过日志了解服务器软硬件信息、检查配置过程中的错误及错误发生的原因。经常分析日志可以了解服务器的负荷，性能安全性，从而及时采取措施纠正错误。

通常，日志被分散在储存不同的设备上。如果你管理数十上百台服务器，你还在使用依次登录每台机器的传统方法查阅日志。这样是不是感觉很繁琐和效率低下。当务之急我们使用集中化的日志管理，例如：开源的 **syslog**，将所有服务器上的日志收集汇总。

集中化管理日志后，日志的统计和检索又成为一件比较麻烦的事情，一般我们使用 **grep**、**awk** 和 **wc** 等 Linux 命令能实现检索和统计，但是对于要求更高的查询、排序和统计等要求和庞大的机器数量依然使用这样的方法难免有点力不从心。

通过我们需要对日志进行集中化管理，将所有机器上的日志信息收集、汇总到一起。**完整的日志数据具有非常重要的作用：**

- 1) **信息查找**。通过检索日志信息，定位相应的 **bug**，找出解决方案。
- 2) **服务诊断**。通过对日志信息进行统计、分析，了解服务器的负荷和服务运行状态，找出耗时请求进行优化等等。
- 3) **数据分析**。如果是格式化的 **log**，可以做进一步的数据分析，统计、聚合出有意义的信息，比如根据请求中的商品 **id**，找出 **TOP10** 用户感兴趣商品。

开源实时日志分析 **ELK** 平台能够完美的解决我们上述的问题，**ELK 由 Elasticsearch、Logstash 和 Kibana 三个开源工具组成：**

1) **ElasticSearch** 是一个基于 **Lucene** 的开源分布式搜索服务器。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。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，是第二流行的企业搜索引擎。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。

在 **elasticsearch** 中，所有节点的数据是均等的。

2) **Logstash** 是一个完全开源的工具，它可以对你的日志进行收集、过滤、分析，支持大量的数据获取方法，并将其存储供以后使用（如搜索）。说到搜索，**logstash** 带有一个 **web** 界面，搜索和展示所有日志。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。

3) **Kibana** 是一个基于浏览器页面的 **Elasticsearch** 前端展示工具，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，可以帮助您汇总、分析和搜索重要数据日志。

为什么要用到 ELK?

一般我们需要进行日志分析场景是：直接在日志文件中 `grep`、`awk` 就可以获得自己想要的信息。但在规模较大的场景中，此方法效率低下，面临问题包括日志量太大如何归档、文本搜索太慢怎么办、如何多维度查询。需要集中化的日志管理，所有服务器上的日志收集汇总。常见解决思路是建立集中式日志收集系统，将所有节点上的日志统一收集，管理，访问。

一般大型系统是一个分布式部署的架构，不同的服务模块部署在不同的服务器上，问题出现时，大部分情况需要根据问题暴露的关键信息，定位到具体的服务器和服务模块，构建一套集中式日志系统，可以提高定位问题的效率。

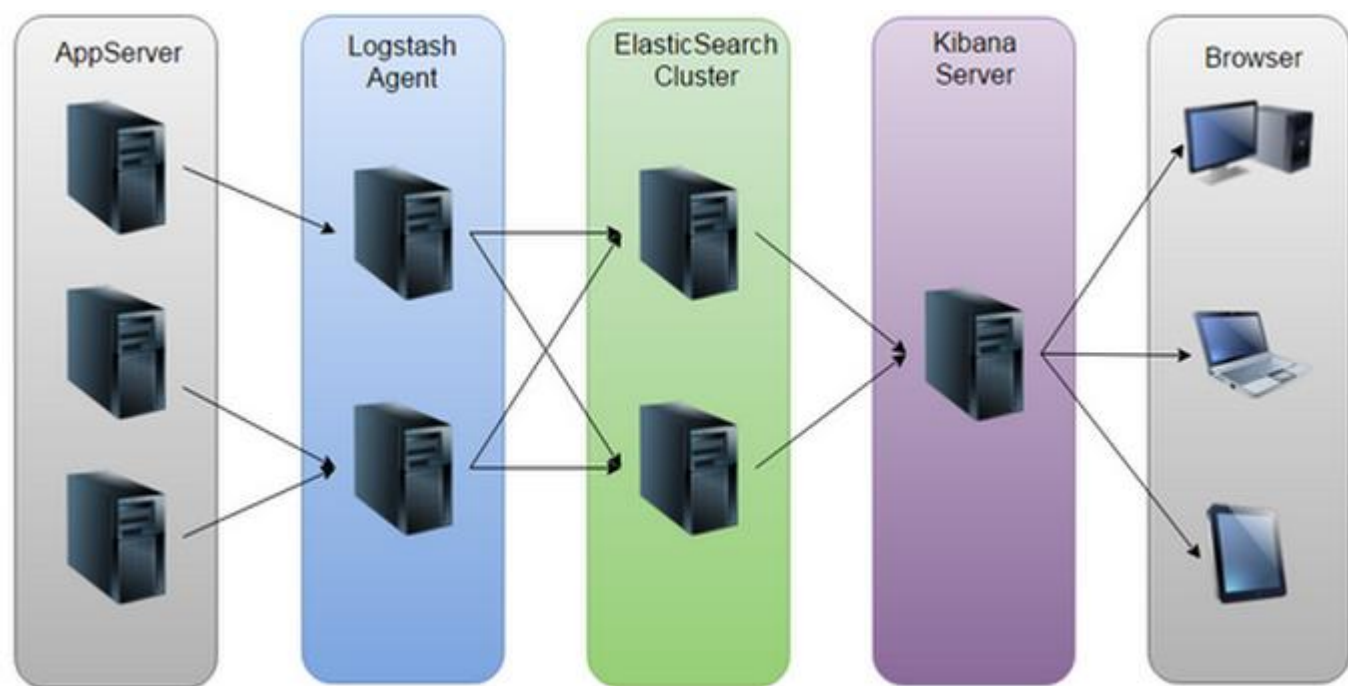
一般大型系统是一个分布式部署的架构，不同的服务模块部署在不同的服务器上，问题出现时，大部分情况需要根据问题暴露的关键信息，定位到具体的服务器和服务模块，构建一套集中式日志系统，可以提高定位问题的效率。

一个完整的集中式日志系统，需要包含以下几个主要特点：

- 1) 收集—能够采集多种来源的日志数据
- 2) 传输—能够稳定的把日志数据传输到中央系统
- 3) 存储—如何存储日志数据
- 4) 分析—可以支持 UI 分析
- 5) 警告—能够提供错误报告，监控机制

ELK 提供了一整套解决方案，并且都是开源软件，之间互相配合使用，完美衔接，高效的满足了很多场合的应用。目前主流的一种日志系统。

ELK 工作原理展示图：

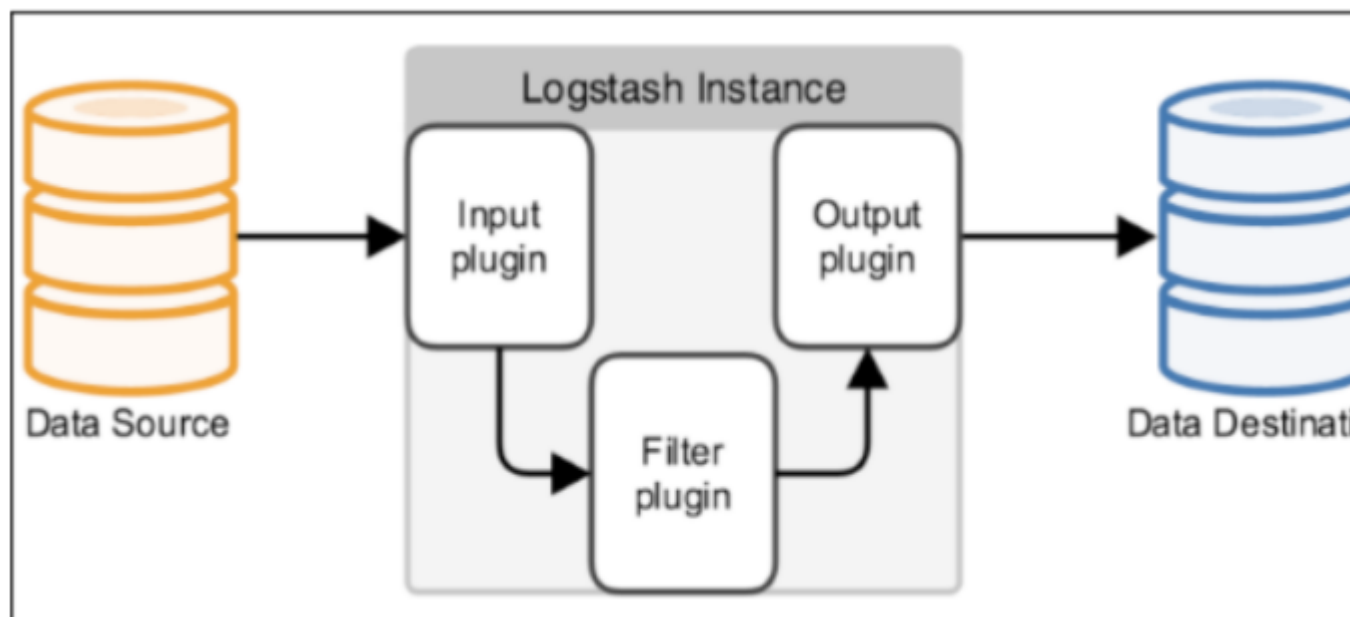


如上图：Logstash 收集 AppServer 产生的 Log，并存放到了 ElasticSearch 集群中，而 Kibana 则从 ES 集群中查询数据生成图表，再返回给 Browser。

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。

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。**Code cs** 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。

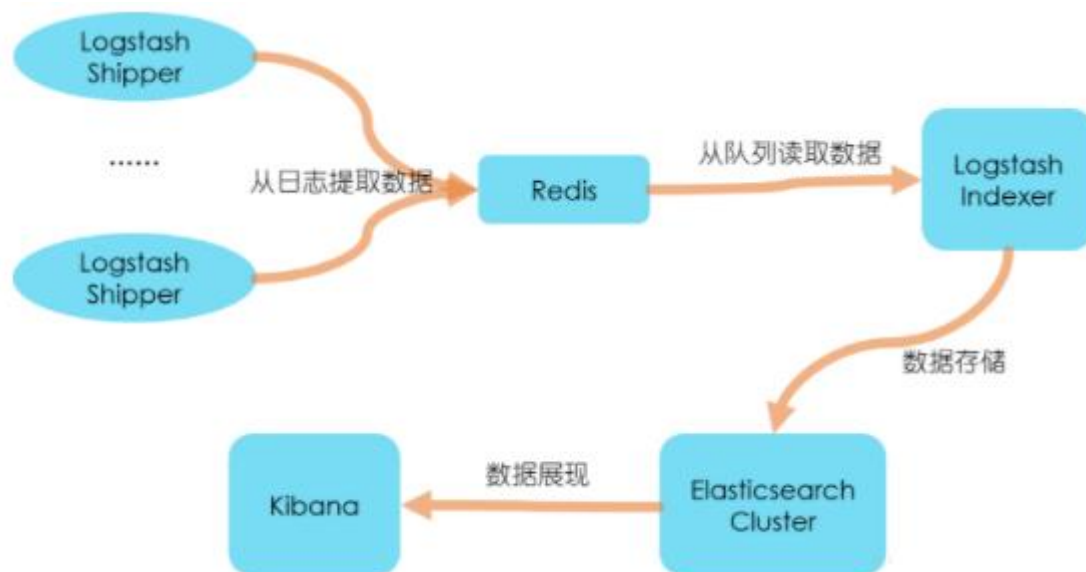
=====ELK 整体方案=====

=====

ELK 中的三个系统分别扮演不同的角色，组成了一个整体的解决方案。Logstash 是一个 ETL 工具，负责从每台机器抓取日志数据，对数据进行格式转换和处理后，输出到 Elasticsearch 中存储。Elasticsearch 是一个分布式搜索引擎和分析引擎，用于数据存储，可提供实时的数据查询。Kibana 是一个数据可视化服务，根据用户的操作从 Elasticsearch 中查询数据，形成相应的分析结果，以图表的形式展现给用户。

ELK 的安装很简单，可以按照"下载->修改配置文件->启动"方法分别部署三个系统，也可以使用 docker 来快速部署。具体的安装方法这里不详细介绍，下面来看一个常见的部署方案，如下图所示，部署思路是：

- 1) 在每台生成日志文件的机器上，部署 Logstash，作为 Shipper 的角色，负责从日志文件中提取数据，但是不做任何处理，直接将数据输出到 Redis 队列(list)中；
- 2) 需要一台机器部署 Logstash，作为 Indexer 的角色，负责从 Redis 中取出数据，对数据进行格式化和相关处理后，输出到 Elasticsearch 中存储；
- 3) 部署 Elasticsearch 集群，当然取决于你的数据量了，数据量小的话可以使用单台服务，如果做集群的话，最好是有 3 个以上节点，同时还需要部署相关的监控插件；
- 4) 部署 Kibana 服务，提供 Web 服务。



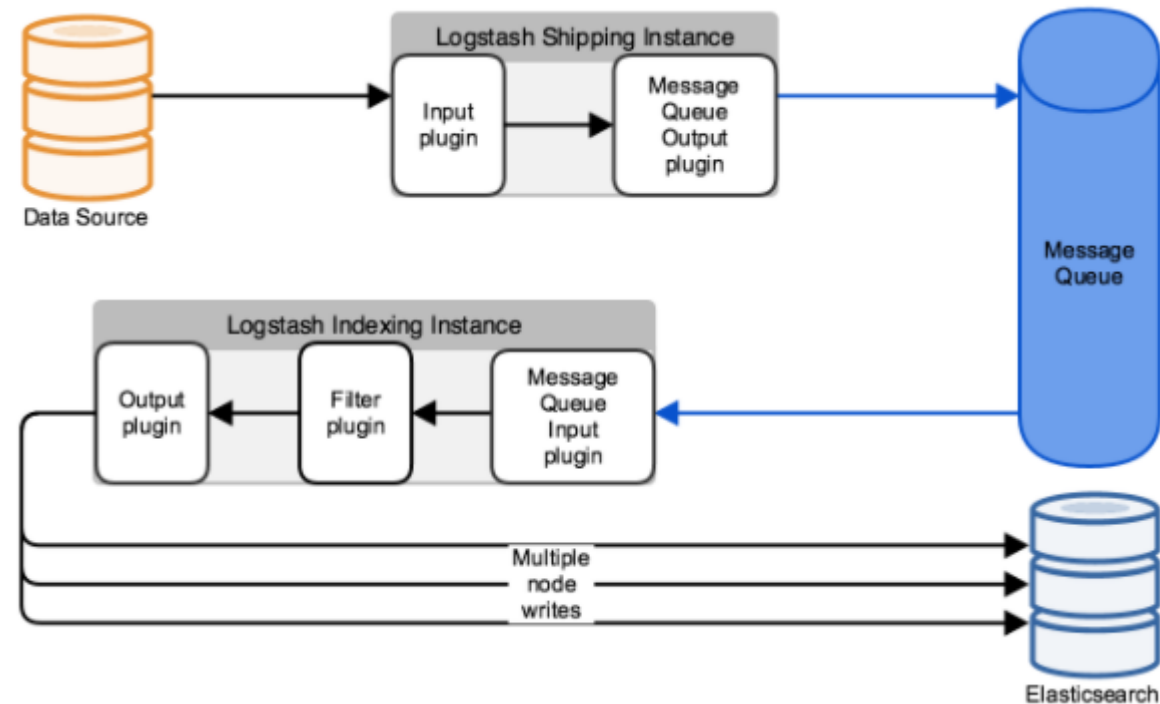
在前期部署阶段，主要工作是 Logstash 节点和 Elasticsearch 集群的部署，而在后期使用阶段，主要工作就是 Elasticsearch 集群的监控和使用 Kibana 来检索、分析日志数据了，当然也可以直接编写程序来消费 Elasticsearch 中的数据。

在上面的部署方案中，我们将 Logstash 分为 Shipper 和 Indexer 两种角色来完成不同的工作，中间通过 Redis 做数据管道，为什么要这样做？为什么不是直接在每台机器上使用 Logstash 提取数据、处理、存入 Elasticsearch？

首先，采用这样的架构部署，有三点优势：**第一**，降低对日志所在机器的影响，这些机器上一般都部署着反向代理或应用服务，本身负载就很重了，所以尽可能的在这些机器上少做事；**第二**，如果有很多台机器需要做日志收集，那么让每台机器都向 Elasticsearch 持续写入数据，必然会

对 Elasticsearch 造成压力，因此需要对数据进行缓冲，同时，这样的缓冲也可以一定程度的保护数据不丢失；**第三**，将日志数据的格式化与处理放到 Indexer 中统一做，可以在一处修改代码、部署，避免需要到多台机器上去修改配置。

其次，我们需要做的是将数据放入一个消息队列中进行缓冲，所以 Redis 只是其中一个选择，也可以是 RabbitMQ、Kafka 等等，在实际生产中，Redis 与 Kafka 用的比较多。由于 Redis 集群一般都是通过 key 来做分片，无法对 list 类型做集群，在数据量大的时候必然不合适了，而 Kafka 天生就是分布式的消息队列系统。



1) 配置 nginx 日志格式

首先需要将 nginx 日志格式规范化，便于做解析处理。在 nginx.conf 文件中设置：

1	log_format main '\$remote_addr "\$time_iso8601" "\$request" \$status \$body_bytes_sent
2	"\$upstream_response_time" "\$http_cookie" "\$http_Authorization" "\$http_token";
	access_log /var/log/nginx/example.access.log main;

2) nginx 日志->>Logstash->>消息队列

这部分是 Logstash Shipper 的工作，涉及 input 和 output 两种插件。input 部分，由于需要提取的是日志文件，一般使用 file 插件，该插件常用的几个参数是：

path: 指定日志文件路径。

type: 指定一个名称，设置 type 后，可以在后面的 filter 和 output 中对不同的 type 做不同的处理，适用于需要消费多个日志文件的场景。

start_position: 指定起始读取位置，“beginning”表示从文件头开始，“end”表示从文件尾开始（类似 tail -f）。

sincedb_path: 与 Logstash 的一个坑有关。通常 Logstash 会记录每个文件已经被读取到的位置，保存在 sincedb 中，如果 Logstash 重启，那么对于同一个文件，会继续从上次记录的位置开始读取。如果想重新从头读取文件，需要删除 sincedb 文件，sincedb_path 则是指定了该文件的路径。为了方便，我们可以根据需要将其设置为“/dev/null”，即不保存位置信息。

1	input {
2	file {
3	type => "example_nginx_access"
4	path => ["/var/log/nginx/example.access.log"]
5	
6	start_position => "beginning"
7	sincedb_path => "/dev/null"
8	}
9	}

output 部分，将数据输出到消息队列，以 **redis** 为例，需要指定 **redis server** 和 **list key** 名称。另外，在测试阶段，可以使用 **stdout** 来查看输出信息。

1	# 输出到 redis
2	output {
3	if [type] == "example_nginx_access" {
4	redis {
5	host => "127.0.0.1"
6	port => "6379"
7	data_type => "list"
8	key => "logstash:example_nginx_access"
9	}
10	# stdout {codec => rubydebug}
11	}
12	}

3) 消息队列->>Logstash->>Elasticsearch

这部分是 **Logstash Indexer** 的工作，涉及 **input**、**filter** 和 **output** 三种插件。在 **input** 部分，我们通过 **redis** 插件将数据从消息队列中取出来。在 **output** 部分，我们通过 **elasticsearch** 插件将数据写入 **Elasticsearch**。

1	# 从 redis 输入数据
2	input {
3	redis {
4	host => "127.0.0.1"
5	port => "6379"
6	data_type => "list"
7	key => "logstash:example_nginx_access"
8	}
9	}
10	
11	output {
12	elasticsearch {
13	index => "logstash-example-nginx-%{+YYYY.MM}"
14	hosts => ["127.0.0.1:9200"]

15	}
16	}

这里，需要重点关注 **filter** 部分，下面列举几个常用的插件，实际使用中根据自身需求从官方文档中查找适合自己业务的插件并使用即可，当然也可以编写自己的插件。

grok: 是 Logstash 最重要的一个插件，用于将非结构化的文本数据转化为结构化的数据。**grok** 内部使用正则语法对文本数据进行匹配，为了降低使用复杂度，其提供了一组 **pattern**，我们可以直接调用 **pattern** 而不需要自己写正则表达式，参考源码 **grok-patterns**。**grok** 解析文本的语法格式是 **%{SYNTAX:SEMANTIC}**，**SYNTAX** 是 **pattern** 名称，**SEMANTIC** 是需要生成的字段名称，使用工具 **Grok Debugger** 可以对解析语法进行调试。例如，在下面的配置中，我们先使用 **grok** 对输入的原始 **nginx** 日志信息（默认以 **message** 作为字段名）进行解析，并添加新的字段 **request_path_with_verb**（该字段的值是 **verb** 和 **request_path** 的组合），然后对 **request_path** 字段做进一步解析。

kv: 用于将某个字段的值进行分解，类似于编程语言中的字符串 **Split**。在下面的配置中，我们将 **request_args** 字段值按照 **"&"** 进行分解，分解后的字段名称以 **"request_args_"** 作为前缀，并且丢弃重复的字段。

geoip: 用于根据 **IP** 信息生成地理位置信息，默认使用自带的一份 **GeoLiteCity database**，也可以自己更换为最新的数据库，但是需要数据格式需要遵循 **Maxmind** 的格式（参考 **GeoLite**），似乎目前只能支持 **legacy database**，数据类型必须是 **.dat**。下载 **GeoLiteCity.dat.gz** 后解压，并将文件路径配置到 **source** 中即可。

translate: 用于检测某字段的值是否符合条件，如果符合条件则将其翻译成新的值，写入一个新的字段，匹配 **pattern** 可以通过 **YAML** 文件来配置。例如，在下面的配置中，我们对 **request_api** 字段翻译成更加易懂的文字描述。

1	filter {
2	grok {
3	match => {"message" => "%{IPORHOST:client_ip} \" %{TIMESTAMP_ISO8601:request_time} HTTP/%{NUMBER:httpversion}\" %{NUMBER:response_status:int} %{NUMBER:response_body_size:float} \"%{NOTSPACE:http_x_forwarder_for}\" \"%{NUMBER:request_time:float}\" \"%{DATA:request_path}\" \"%{DATA:http_token}\"\""}}
4	add_field => {"request_path_with_verb" => "%{verb} %{request_path}"}
5	}
6	grok {
7	match => {"request_path" => "%{URIPATH:request_api} (?:\%{NOTSPACE}+)"}
8	add_field => {"request_annotation" => "%{request_api}"}
9	}
10	kv {
11	prefix => "request_args_"
12	field_split => "&"
13	source => "request_args"
14	allow_duplicate_values => false
15	}
16	}
17	
18	
19	
20	
21	

22	geoip {
23	source => "client_ip"
24	database => "/home/elktest/geoip_data/GeoLiteCity.dat"
25	}
26	
27	translate {
28	field => request_path
29	destination => request_annotation
30	regex => true
31	exact => true
32	dictionary_path => "/home/elktest/api_annotation.yaml"
	override => true
	}
	}

Elasticsearch

Elasticsearch 承载了数据存储和查询的功能，其基础概念和使用方法可以参考另一篇博文 [Elasticsearch 使用总结](#)，这里主要介绍些实际生产中的问题和方法：

1) 关于集群配置，重点关注三个参数：第一，`discovery.zen.ping.unicast.hosts`，Elasticsearch 默认使用 Zen Discovery 来做节点发现机制，推荐使用 unicast 来做通信方式，在该配置项中列举出 Master 节点。第二，`discovery.zen.minimum_master_nodes`，该参数表示集群中可工作的具有 Master 节点资格的最小数量，默认值是 1。为了提高集群的可用性，避免脑裂现象（所谓脑裂，就是同一个集群中的不同节点，对集群的状态有不一致的理解。），官方推荐设置为 $(N/2)+1$ ，其中 N 是具有 Master 资格的节点的数量。第三，`discovery.zen.ping_timeout`，表示节点在发现过程中的等待时间，默认值是 3 秒，可以根据自身网络环境进行调整，一定程度上提供可用性。

1	<code>discovery.zen.ping.unicast.hosts: ["master1", "master2", "master3"]</code>
2	<code>discovery.zen.minimum_master_nodes: 2</code>
3	<code>discovery.zen.ping_timeout: 10</code>

2) 关于集群节点，第一，节点类型包括：候选 Master 节点、数据节点和 Client 节点。通过设置两个配置项 `node.master` 和 `node.data` 为 true 或 false，来决定将一个节点分配为什么类型的节点。第二，尽量将候选 Master 节点和 Data 节点分离开，通常 Data 节点负载较重，需要考虑单独部署。

3) 关于内存，Elasticsearch 默认设置的内存是 1GB，对于任何一个业务部署来说，这个都太小了。通过指定 `ES_HEAP_SIZE` 环境变量，可以修改其堆内存大小，服务进程在启动时候会读取这个变量，并相应的设置堆的大小。建议设置系统内存的一半给 Elasticsearch，但是不要超过 32GB。参考官方文档。

4) 关于硬盘空间，Elasticsearch 默认将数据存储在 `/var/lib/elasticsearch` 路径下，随着数据的增长，一定会出现硬盘空间不够用的情形，此时就需要给机器挂载新的硬盘，并将 Elasticsearch 的路径配置到新硬盘的路径下。通过 `path.data` 配置项来进行设置，比如 `path.data: /data1,/var/lib/elasticsearch,/data`。需要注意的是，同一分片下的数据只能写入到一个路径下，因此还是需要合理的规划和监控硬盘的使用。

5) 关于 Index 的划分和分片的个数，这个需要根据数据量来做权衡了，Index 可以按时间划分，比如每月一个或者每天一个，在 Logstash 输出时进行配置，shard 的数量也需要做好控制。

6) 关于监控，笔者使用过 head 和 marvel 两个监控插件，head 免费，功能相对有限，marvel 现在需要收费了。另外，不要在数据节点开启监控插件。

Kibana

Kibana 提供的是数据查询和显示的 Web 服务，有丰富的图表样板，能满足大部分的数据可视化需求，这也是很多人选择 ELK 的主要原因之一。UI 的操作没有什么特别需要介绍的，经常使用就会熟练，这里主要介绍经常遇到的三个问题。

a) 查询语法

在 Kibana 的 Discover 页面中，可以输入一个查询条件来查询所需的数据。查询条件的写法使用的是 Elasticsearch 的 Query String 语法，而不是 Query DSL，参考官方文档 [query-string-syntax](#)，这里列举其中部分常用的：

.单字段的全文检索，比如搜索 args 字段中包含 first 的文档，写作 `args:first`;

.单字段的精确检索，比如搜索 args 字段值为 first 的文档，写作 `args: "first"`;

.多个检索条件的组合，使用 NOT, AND 和 OR 来组合，注意必须是大写，比如 `args:("first" OR "second") AND NOT agent: "third"`;

.字段是否存在，`_exists_:agent` 表示要求 agent 字段存在，`_missing_:agent` 表示要求 agent 字段不存在;

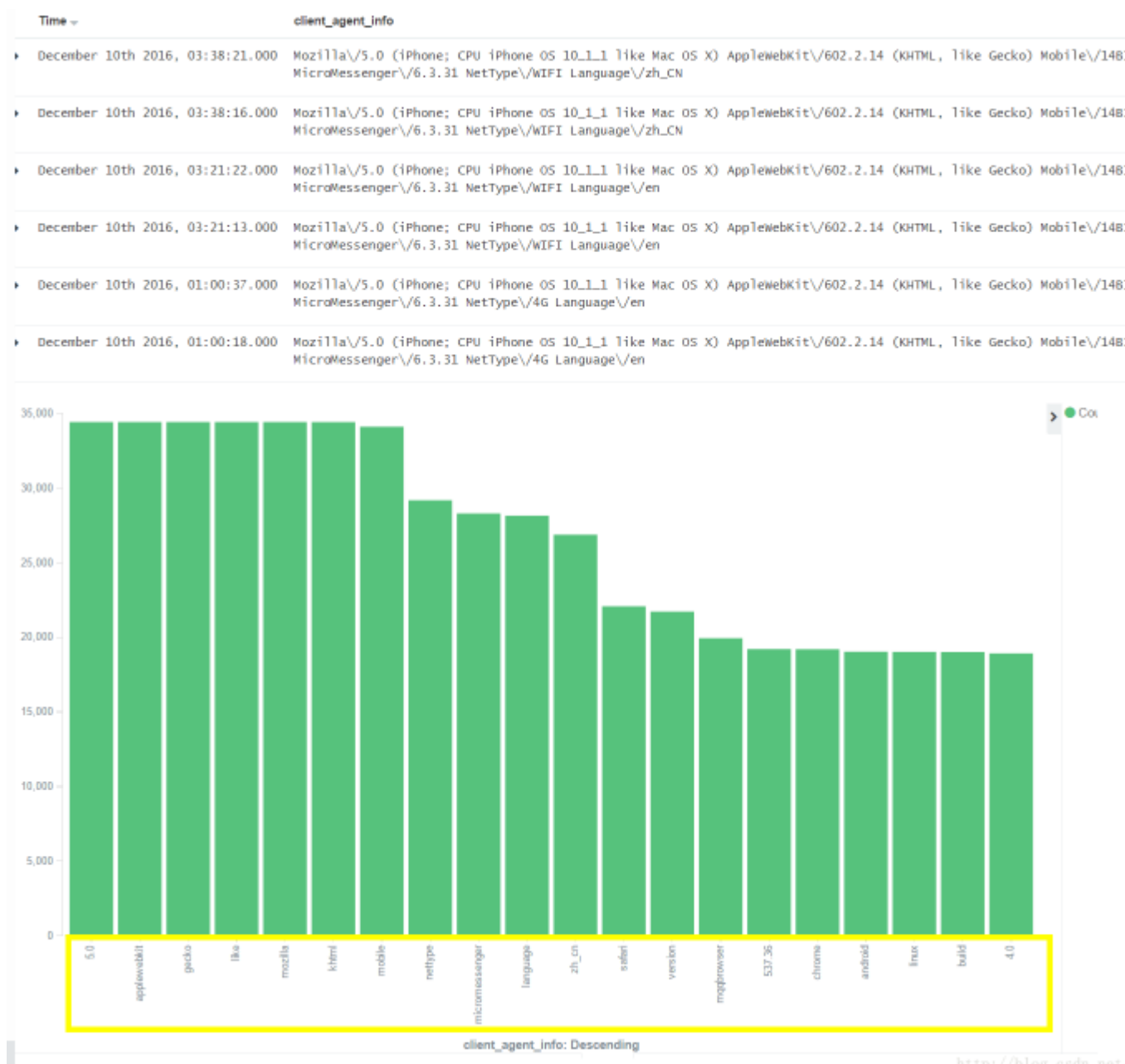
.通配符：用 ? 表示单字母，* 表示任意个字母。

b) 错误“Discover: Request Timeout after 30000ms”

这个错误经常发生在要查询的数据量比较大的情况下，此时 Elasticsearch 需要较长时间才能返回，导致 Kibana 发生 Timeout 报错。解决问题的方法，就是在 Kibana 的配置文件中修改 `elasticsearch.requestTimeout` 一项的值，然后重启 Kibana 服务即可，注意单位是 ms。

c) 疑惑“字符串被分解了”

经常碰到这样一个问题：为什么查询结果的字段值是正确的，可是做图表时却发现字段值被分解了，不是想要的结果？如下图所示的 `client_agent_info` 字段。



得到这样一个不正确结果的原因是使用了 **Analyzed** 字段来做图表分析，默认情况下 **Elasticsearch** 会对字符串数据进行分析，建立倒排索引，所以如果对这么一个字段进行 **terms** 聚合，必然会得到上面所示的错误结果了。那么应该怎么做才对？默认情况下，**Elasticsearch** 还会创建一个相对应的没有被 **Analyzed** 的字段，即带 **“.raw”** 后缀的字段，在这样的字段上做聚合分析即可。

又会有很多人问这样的问题：为什么我的 **Elasticsearch** 没有自动创建带 **“.raw”** 后缀的字段？然而在 **Logstash** 中输出数据时，设置 **index** 名称前缀为 **“logstash-”** 就有了这个字段。这个问题的根源是 **Elasticsearch** 的 **dynamic template** 在捣鬼，**dynamic template** 用于指导 **Elasticsearch** 如何为插入的数据自动建立 **Schema** 映射关系，默认情况下，**Logstash** 会在 **Elasticsearch** 中建立一个名为 **“logstash”** 的模板，所有前缀为 **“logstash-”** 的 **index** 都会参照这个模板来建立映射关系，在该模板中申明了要为每个字符串数据建立一个额外的带 **“.raw”** 后缀的字段。可以向 **Elasticsearch** 来查询你的模板，使用 **API**：GET http://localhost:9200/_template。

以上便是对 ELK 日志系统的总结介绍，还有一个重要的功能没有提到，就是如何将日志数据与自身产品业务的数据融合起来。举个例子，在 nginx 日志中，通常会包含 API 请求访问时携带的用户 Token 信息，由于 Token 是有时效性的，我们需要及时将这些 Token 转换成真实的用户信息存储下来。这样的需求通常有两种实现方式，一种是自己写一个 Logstash filter，然后在 Logstash 处理数据时调用；另一种是将 Logstash Indexer 产生的数据再次输出到消息队列中，由我们自己的脚本程序从消息队列中取出数据，做相应的业务处理后，输出到 Elasticsearch 中。

=====ELK 环境部署=====

=====

(0) 基础环境介绍

系统： Centos7.1

防火墙： 关闭

Sellinux： 关闭

机器环境： 两台

elk-node1: 192.168.1.160 **#master 机器**

elk-node2: 192.168.1.161 **#slave 机器**

注明：

master-slave 模式：

master 收集到日志后，会把一部分数据碎片到 slave 上（随机的一部分数据）；同时，master 和 slave 又都会各自做副本，并把副本放到对方机器上，这样就保证了数据不会丢失。

如果 master 宕机了，那么客户端在日志采集配置中将 elasticsearch 主机指向改为 slave，就可以保证 ELK 日志的正常采集和 web 展示。

=====

由于 elk-node1 和 elk-node2 两台是虚拟机，没有外网 ip，所以访问需要通过宿主机进行代理转发实现。

有以下两种转发设置：（任选其一）

通过访问宿主机的 19200,19201 端口分别转发到 elk-node1,elk-node2 的 9200 端口

通过访问宿主机的 15601 端口转发到 elk-node1 的 5601 端口

宿主机： 112.110.115.10(内网 ip 为 192.168.1.7) （为了不让线上的真实 ip 暴露，这里任意给了一个 ip 做记录）

a) 通过宿主机的 haproxy 服务进行代理转发，如下是宿主机上的代理配置：

```
[root@kvm-server conf]# pwd
```

```
/usr/local/haproxy/conf
```

```
[root@kvm-server conf]# cat haproxy.cfg
```

```
.....
```

```
.....
```

```
listen node1-9200 0.0.0.0:19200
```

```
mode tcp
```

```
option tcplog
```

```
balance roundrobin
```

```
server 192.168.1.160 192.168.1.160:9200 weight 1 check inter 1s rise 2 fall 2
```

```

listen node2-9200 0.0.0.0:19201
mode tcp
option tcplog
balance roundrobin
server 192.168.1.161 192.168.1.161:9200 weight 1 check inter 1s rise 2 fall 2
listen node1-5601 0.0.0.0:15601
mode tcp
option tcplog
balance roundrobin
server 192.168.1.160 192.168.1.160:5601 weight 1 check inter 1s rise 2 fall 2
重启 haproxy 服务
[root@kvm-server conf]# /etc/init.d/haproxy restart
设置宿主机防火墙
[root@kvm-server conf]# cat /etc/sysconfig/iptables
.....
-A INPUT -p tcp -m state --state NEW -m tcp --dport 19200 -j ACCEPT
-A INPUT -p tcp -m state --state NEW -m tcp --dport 19201 -j ACCEPT
-A INPUT -p tcp -m state --state NEW -m tcp --dport 15601 -j ACCEPT

[root@kvm-server conf]# /etc/init.d/iptables restart

```

b) 通过宿主机的 NAT 端口转发实现

```

[root@kvm-server conf]# iptables -t nat -A PREROUTING -p tcp -m tcp --dport
19200 -j DNAT --to-destination 192.168.1.160:9200
[root@kvm-server conf]# iptables -t nat -A POSTROUTING -d 192.168.1.160/3
2 -p tcp -m tcp --sport 9200 -j SNAT --to-source 192.168.1.7
[root@kvm-server conf]# iptables -t filter -A INPUT -p tcp -m state --state NE
W -m tcp --dport 19200 -j ACCEPT
[root@kvm-server conf]# iptables -t nat -A PREROUTING -p tcp -m tcp --dport
19201 -j DNAT --to-destination 192.168.1.161:9200
[root@kvm-server conf]# iptables -t nat -A POSTROUTING -d 192.168.1.161/3
2 -p tcp -m tcp --sport 9200 -j SNAT --to-source 192.168.1.7
[root@kvm-server conf]# iptables -t filter -A INPUT -p tcp -m state --state NE
W -m tcp --dport 19201 -j ACCEPT
[root@kvm-server conf]# iptables -t nat -A PREROUTING -p tcp -m tcp --dport
15601 -j DNAT --to-destination 192.168.1.160:5601
[root@kvm-server conf]# iptables -t nat -A POSTROUTING -d 192.168.1.160/3
2 -p tcp -m tcp --sport 5601 -j SNAT --to-source 192.168.1.7
[root@kvm-server conf]# iptables -t filter -A INPUT -p tcp -m state --state NE
W -m tcp --dport 15601 -j ACCEPT
[root@kvm-server conf]# service iptables save
[root@kvm-server conf]# service iptables restart

```

提醒一点：

nat 端口转发设置成功后，/etc/sysconfig/iptables 文件里要注释掉下面两行！不然 nat 转发会有问题！一般如上面在 nat 转发规则设置好并 save 和 restart 防火墙之后就会自动在/etc/s

ysconfig/iptables 文件里删除掉下面两行内容了。

```
[root@kvm-server conf]# vim /etc/sysconfig/iptables
```

.....

```
#-A INPUT -j REJECT --reject-with icmp-host-prohibited
```

```
#-A FORWARD -j REJECT --reject-with icmp-host-prohibited
```

```
[root@linux-node1 ~]# service iptables restart
```

```
=====
```

(1) Elasticsearch 安装配置

基础环境安装 (elk-node1 和 elk-node2 同时操作)

1) 下载并安装 GPG Key

```
[root@elk-node1 ~]# rpm --import https://packages.elastic.co/GPG-KEY-elastic
search
```

2) 添加 yum 仓库

```
[root@elk-node1 ~]# vim /etc/yum.repos.d/elasticsearch.repo
```

```
[elasticsearch-2.x]
```

```
name=Elasticsearch repository for 2.x packages
```

```
baseurl=http://packages.elastic.co/elasticsearch/2.x/centos
```

```
gpgcheck=1
```

```
gpgkey=http://packages.elastic.co/GPG-KEY-elasticsearch
```

```
enabled=1
```

3) 安装 elasticsearch

```
[root@elk-node1 ~]# yum install -y elasticsearch
```

4) 安装相关测试软件

#提前先下载安装 epel 源: epel-release-latest-7.noarch.rpm, 否则 yum 会报错:No Package.....

```
[root@elk-node1 ~]# wget http://dl.fedoraproject.org/pub/epel/epel-release-lat
est-7.noarch.rpm
```

```
[root@elk-node1 ~]# rpm -ivh epel-release-latest-7.noarch.rpm
```

#安装 Redis

```
[root@elk-node1 ~]# yum install -y redis
```

#安装 Nginx

```
[root@elk-node1 ~]# yum install -y nginx
```

#安装 java

```
[root@elk-node1 ~]# yum install -y java
```

安装完 java 后, 检测

```
[root@elk-node1 ~]# java -version
```

```
openjdk version "1.8.0_102"
```

```
OpenJDK Runtime Environment (build 1.8.0_102-b14)
```

```
OpenJDK 64-Bit Server VM (build 25.102-b14, mixed mode)
```

配置部署 (下面先进行 elk-node1 的配置)

1) 配置修改配置文件

```
[root@elk-node1 ~]# mkdir -p /data/es-data
```

```
[root@elk-node1 ~]# vim /etc/elasticsearch/elasticsearch.yml
```

【将里面内容情况，配置下面内容】

```
cluster.name: huanqiu          # 组名（同一个组，组名必须一致）
node.name: elk-node1          # 节点名称，建议和主机名一致
path.data: /data/es-data      # 数据存放的路径
path.logs: /var/log/elasticsearch/ # 日志存放的路径
bootstrap.mlockall: true      # 锁住内存，不被使用到交换分区去
network.host: 0.0.0.0         # 网络设置
http.port: 9200               # 端口
```

2) 启动并查看

```
[root@elk-node1 ~]# chown -R elasticsearch.elasticsearch /data/
[root@elk-node1 ~]# systemctl start elasticsearch
[root@elk-node1 ~]# systemctl status elasticsearch
CGroup: /system.slice/elasticsearch.service
└─3005 /bin/java -Xms256m -Xmx1g -Djava.awt.headless=true -XX:+UseParNe
wGC -XX:+UseConcMarkSweepGC -XX:CMSI...
```

注意：上面可以看出 **elasticsearch** 设置的内存最小 **256m**，最大 **1g**

====温馨提示：Elasticsearch 启动出现"could not find java"=====

1	yum 方法安装 elasticsearch，使用"systemctl start elasticsearch"启动服务失败.
2	"systemctl status elasticsearch"查看，发现报错说 could not find java
3	但是"java -version" 查看发现 java 已经安装了
4	
5	这是因为 elasticsearch 在启动过程中，引用的 java 路径找不到
6	
7	解决办法：在 elasticsearch 配置文件中定义 java 全路径
8	
9	[root@elk-node01 ~]# java -version
10	java version "1.8.0_131"
11	Java(TM) SE Runtime Environment (build 1.8.0_131-b11)
12	Java HotSpot(TM) 64-Bit Server VM (build 25.131-b11, mixed mode
13	
14	[root@elk-node01 ~]# find / -name java
15	/var/lib/alternatives/java
16	/usr/share/swig/2.0.10/java
17	/usr/java
18	/usr/java/jdk1.8.0_131/bin/java
19	/usr/java/jdk1.8.0_131/jre/bin/java
20	/usr/bin/java
21	/etc/pki/java
22	/etc/pki/ca-trust/extracted/java
23	/etc/alternatives/java
24	
25	[root@elk-node01 ~]# vim /etc/sysconfig/elasticsearch

26	添加 JAVA_HOME 环境变量的配置
27	JAVA_HOME=/usr/java/jdk1.8.0_131

```
[root@linux-node1 src]# netstat -antlp |egrep "9200|9300"
```

```
tcp6 0 0 :::9200 :::* LISTEN 3005/java
```

```
tcp6 0 0 :::9300 :::* LISTEN 3005/java
```

然后通过 web 访问（访问的浏览器最好用 [google 浏览器](#)）

<http://112.110.115.10:19200/>



```
{
  "name" : "elk-node1",
  "cluster_name" : "huanqiu",
  "cluster_uuid" : "UZ5PQ9NwSdi15p62PY3QYA",
  "version" : {
    "number" : "2.4.1",
    "build_hash" : "c67dc32e24162035d18d6fe1e952c4c4c4cbe79d16",
    "build_timestamp" : "2016-09-27T18:57:55Z",
    "build_snapshot" : false,
    "lucene_version" : "5.5.2"
  },
  "tagline" : "You Know, for Search"
}
```

3) 通过命令的方式查看数据（在 112.110.115.10 宿主机或其他外网服务器上查看，如下）

```
[root@kvm-server src]# curl -i -XGET 'http://192.168.1.160:9200/_count?pretty' -d '{"query":{"match_all":{"}}}'
```

```
HTTP/1.1 200 OK
```

```
Content-Type: application/json; charset=UTF-8
```

```
Content-Length: 95
```

```
{
  "count" : 0,
  "_shards" : {
    "total" : 0,
    "successful" : 0,
    "failed" : 0
  }
}
```

这样感觉用命令来查看，特别的不爽。

4) 接下来安装插件，使用插件进行查看~ （下面两个插件要在 elk-node1 和 elk-node2 上都要安装）

4.1) 安装 head 插件

```
=====
```

=====

a) 插件安装方法一

```
[root@elk-node1 src]# /usr/share/elasticsearch/bin/plugin install mobz/elasticsearch-head
```

b) 插件安装方法二

首先下载 head 插件，下载到/usr/local/src 目录下

下载地址: <https://github.com/mobz/elasticsearch-head>

=====

=====

head 插件包百度云盘下载: <https://pan.baidu.com/s/1boBE0qj>

提取密码: ifj7

=====

=====

```
[root@elk-node1 src]# unzip elasticsearch-head-master.zip
```

```
[root@elk-node1 src]# ls
```

```
elasticsearch-head-master elasticsearch-head-master.zip
```

在/usr/share/elasticsearch/plugins 目录下创建 head 目录

然后将上面下载的 elasticsearch-head-master.zip 解压后的文件都移到/usr/share/elasticsearch/plugins/head 下

接着重启 elasticsearch 服务即可!

```
[root@elk-node1 src]# cd /usr/share/elasticsearch/plugins/
```

```
[root@elk-node1 plugins]# mkdir head
```

```
[root@elk-node1 plugins]# ls
```

```
head
```

```
[root@elk-node1 plugins]# cd head
```

```
[root@elk-node1 head]# cp -r /usr/local/src/elasticsearch-head-master/* ./
```

```
[root@elk-node1 head]# pwd
```

```
/usr/share/elasticsearch/plugins/head
```

```
[root@elk-node1 head]# chown -R elasticsearch:elasticsearch /usr/share/elasticsearch/plugins
```

```
[root@elk-node1 head]# ll
```

```
total 40
```

```
-rw-r--r--. 1 elasticsearch elasticsearch 104 Sep 28 01:57 elasticsearch-head.s  
ublime-project
```

```
-rw-r--r--. 1 elasticsearch elasticsearch 2171 Sep 28 01:57 Gruntfile.js
```

```
-rw-r--r--. 1 elasticsearch elasticsearch 3482 Sep 28 01:57 grunt_fileSets.js
```

```
-rw-r--r--. 1 elasticsearch elasticsearch 1085 Sep 28 01:57 index.html
```

```
-rw-r--r--. 1 elasticsearch elasticsearch 559 Sep 28 01:57 LICENCE
```

```
-rw-r--r--. 1 elasticsearch elasticsearch 795 Sep 28 01:57 package.json
```

```
-rw-r--r--. 1 elasticsearch elasticsearch 100 Sep 28 01:57 plugin-descriptor.pro  
perties
```

```
-rw-r--r--. 1 elasticsearch elasticsearch 5211 Sep 28 01:57 README.textile
```

```
drwxr-xr-x. 5 elasticsearch elasticsearch 4096 Sep 28 01:57 _site
```

```
drwxr-xr-x. 4 elasticsearch elasticsearch 29 Sep 28 01:57 src
drwxr-xr-x. 4 elasticsearch elasticsearch 66 Sep 28 01:57 test
[root@elk-node1 _site]# systemctl restart elasticsearch
```

=====

=====

插件访问（最好提前将 **elk-node2** 节点的配置和插件都安装后，再进行访问和数据插入测试）

http://112.110.115.10:19200/_plugin/head/



先插入数据实例，测试下

如下：打开“复合查询”，在 POST 选项下，任意输入如 `/index-demo/test`，然后在下面输入数据（注意内容之间换行的逗号不要漏掉）；

数据输入好之后（如下输入 `wangshibo; hello world` 内容），下面点击“验证 JSON”->“提交请求”，提交成功后，观察右栏里出现的信息：有 `index`，`type`，`version` 等信息，`failed:0`（成功消息）

112.110.115.10:19200/_plugin/head/

应用 网址导航 百度 爱淘宝 天猫 游戏大全 网址大全 百度Hao123网址导航 360

Elasticsearch

http://112.110.115.10:19200/ 连接 huanqiu 集群健康

概览 索引 数据浏览 基本查询 [+] 复合查询 [+]

历史记录

▼ 查询

http://112.110.115.10:19200/

/index-demo/test POST

```
{
  "user": "wangshibo",
  "mesg": "hello world"
}
```

提交请求 验证 JSON 易读

结果转换器 ?

重复请求

显示选项 ?

```
{
  "_index": "index-demo",
  "_type": "test",
  "_id": "AVg-K__Os1YzW2y3IzKM",
  "_version": 1,
  "_shards": {
    "total": 2,
    "successful": 1,
    "failed": 0
  },
  "created": true
}
```

再查看测试实例，如下：

"复合查询"下，选择 GET 选项，在/index-demo/test/后面输入上面 POST 结果中的 id 号，不输入内容，即{}括号里为空！

然后点击"验证 JSON"-"提交请求"，观察右栏内就有了上面插入的数据了（即 wangshibo, hello world）

← → ↻ 112.110.115.10:19200/_plugin/head/

应用 网址导航 百度 爱淘宝 天猫 游戏大全 网址大全 hao 百度Hao123网址导航 + 36

Elasticsearch

http://112.110.115.10:19200/ 连接 huanqiu 集群健康

概览 索引 数据浏览 基本查询 [+]
复合查询 [+]

▶ 历史记录
▼ 查询

http://112.110.115.10:19200/
/index-demo/test/AVg-K__Os1YzW2y3IzKM GET

{}

提交请求 验证 JSON 易读

▶ 结果转换器 ?
▶ 重复请求 ?
▶ 显示选项 ?

```
{
  "_index": "index-demo",
  "_type": "test",
  "_id": "AVg-K__Os1YzW2y3IzKM",
  "_version": 1,
  "found": true,
  "_source": {
    "user": "wangshibo",
    "mesg": "hello world"
  }
}
```

打开"基本查询", 查看下数据, 如下, 即可查询到上面插入的数据:

← → ↻ 112.110.115.10:19200/_plugin/head/

应用 网址导航 百度 爱淘宝 天猫 游戏大全 网址大全 hao 百度Hao123网址导航 + 36

Elasticsearch

http://112.110.115.10:19200/ 连接 huanqiu 集群健康

概览 索引 数据浏览 基本查询 [+]
复合查询 [+]

搜索 index-demo (1 个文档) 的文档, 查询条件:

must match_all + -

搜索 返回格式: Table 显示数量: 10 显示查询语句

查询 5 个分片中用的 5 个. 1 命中. 耗时 0.008 秒

_index	_type	_id	_score	user	mesg
index-demo	test	AVg-K__Os1YzW2y3IzKM	1	wangshibo	hello world

打开“数据浏览”，也能查看到插入的数据：

112.110.115.10:19200/_plugin/head/

Elasticsearch http://112.110.115.10:19200/ 连接 huanqiu 集群健康

概览 索引 数据浏览 基本查询 [+] 复合查询 [+]

数据浏览

所有索引

索引

index-demo

类型

test

字段

▶ msg

▶ user

查询 5 个分片中用的 5 个. 1 命中. 耗时 0.007 秒

_index	_type	_id
index-demo	test	AVg-K__Os1YzW2y3lzKM

如下：一定要提前在 **elk-node2** 节点上也完成配置（配置内容在下面提到），否则上面插入数据后，集群状态会呈现黄色 **yellow** 状态，**elk-node2** 完成配置加入到集群里后就会恢复到正常的绿色状态。

112.110.115.10:19200/_plugin/head/

Elasticsearch http://112.110.115.10:19200/ 连接 huanqiu 集群健康

概览 索引 数据浏览 基本查询 [+] 复合查询 [+]

集群概览 集群排序 Sort Indices View Aliases Index Filter

index-demo
size: 3.79ki (7.59ki)
docs: 1 (2)

信息 动作

★ elk-node1
信息 动作

● elk-node2
信息 动作

0 1 2 3 4

0 1 2 3 4

4.2) 安装 Kopf 监控插件

```
=====
```

a) 监控插件安装方法一

```
[root@elk-node1 src]# /usr/share/elasticsearch/bin/plugin install lmenezes/elasticsearch-kopf
```

b) 监控插件安装方法二

首先下载监控插件 Kopf，下载到/usr/local/src 目录下

下载地址：<https://github.com/lmenezes/elasticsearch-kopf>

```
=====
```

Kopf 插件包百度云盘下载：<https://pan.baidu.com/s/1qYixSL2>

提取密码：ya4t

```
=====
```

```
[root@elk-node1 src]# unzip elasticsearch-kopf-master.zip
```

```
[root@elk-node1 src]# ls
```

```
elasticsearch-kopf-master elasticsearch-kopf-master.zip
```

在/usr/share/elasticsearch/plugins 目录下创建 Kopf 目录

然后将上面下载的 elasticsearch-kopf-master.zip 解压后的文件都移到/usr/share/elasticsearch/plugins/kopf 下

接着重启 elasticsearch 服务即可！

```
[root@elk-node1 src]# cd /usr/share/elasticsearch/plugins/
```

```
[root@elk-node1 plugins]# mkdir Kopf
```

```
[root@elk-node1 plugins]# cd Kopf
```

```
[root@elk-node1 Kopf]# cp -r /usr/local/src/elasticsearch-kopf-master/* ./
```

```
[root@elk-node1 Kopf]# pwd
```

```
/usr/share/elasticsearch/plugins/kopf
```

```
[root@elk-node1 Kopf]# chown -R elasticsearch:elasticsearch /usr/share/elasticsearch/plugins
```

```
[root@elk-node1 Kopf]# ll
```

```
total 40
```

```
-rw-r--r--. 1 elasticsearch elasticsearch 237 Sep 28 16:28 CHANGELOG.md
```

```
drwxr-xr-x. 2 elasticsearch elasticsearch 22 Sep 28 16:28 dataset
```

```
drwxr-xr-x. 2 elasticsearch elasticsearch 73 Sep 28 16:28 docker
```

```
-rw-r--r--. 1 elasticsearch elasticsearch 4315 Sep 28 16:28 Gruntfile.js
```

```
drwxr-xr-x. 2 elasticsearch elasticsearch 4096 Sep 28 16:28 imgs
```

```
-rw-r--r--. 1 elasticsearch elasticsearch 1083 Sep 28 16:28 LICENSE
```

```
-rw-r--r--. 1 elasticsearch elasticsearch 1276 Sep 28 16:28 package.json
```

```
-rw-r--r--. 1 elasticsearch elasticsearch 102 Sep 28 16:28 plugin-descriptor.properties
```

```
-rw-r--r--. 1 elasticsearch elasticsearch 3165 Sep 28 16:28 README.md
```

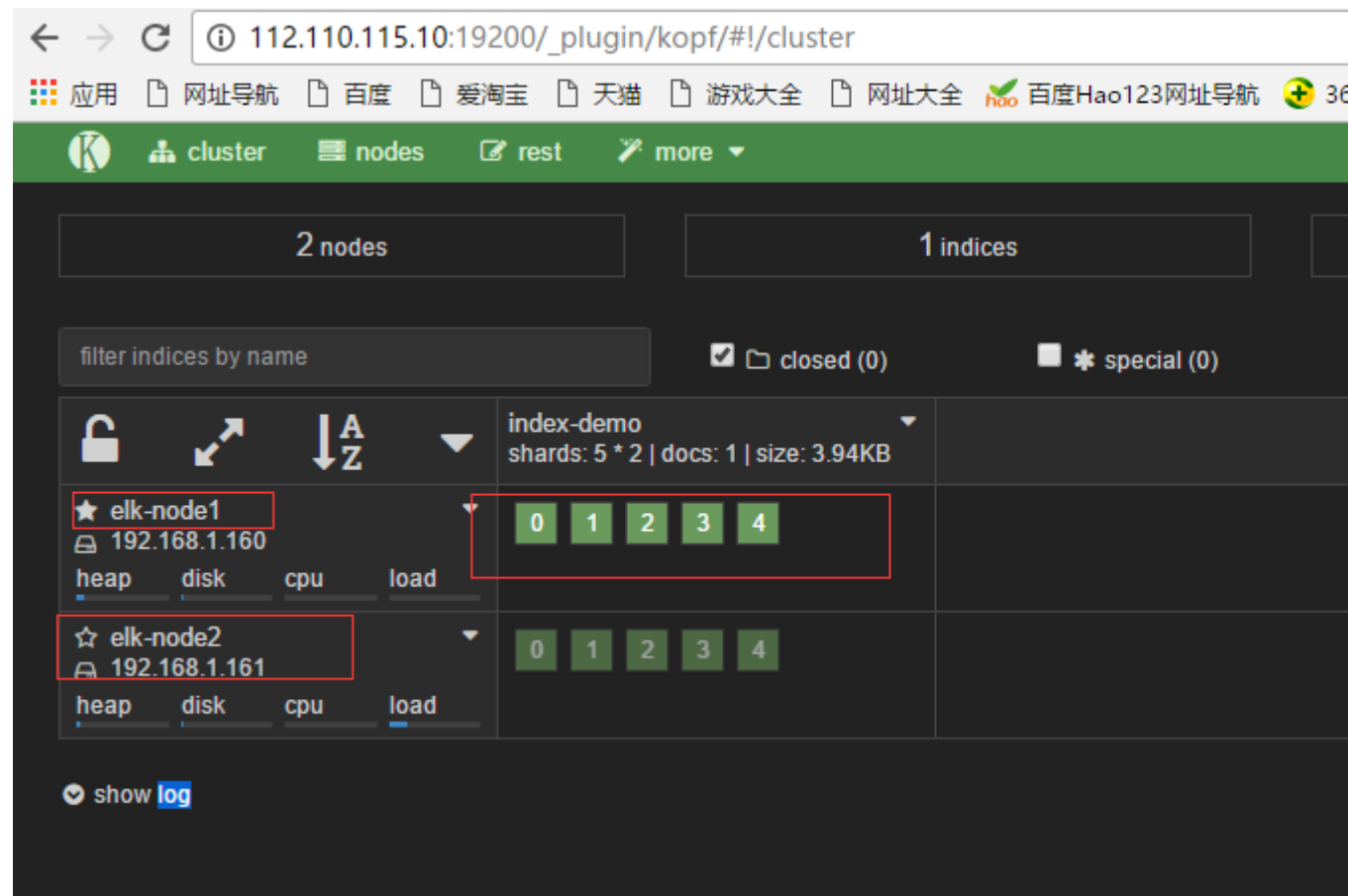
```
drwxr-xr-x. 6 elasticsearch elasticsearch 4096 Sep 28 16:28 _site
```

```
drwxr-xr-x. 4 elasticsearch elasticsearch 27 Sep 28 16:28 src
drwxr-xr-x. 4 elasticsearch elasticsearch 4096 Sep 28 16:28 tests
```

```
[root@elk-node1 _site]# systemctl restart elasticsearch
```

访问插件：（如下，同样要提前安装好 **elk-node2** 节点上的插件，否则访问时会出现集群节点为黄色的 **yellow** 告警状态）

http://112.110.115.10:19200/_plugin/kopf/#!/cluster



```
*****
*****
```

下面进行节点 **elk-node2** 的配置 （如上的两个插件也在 **elk-node2** 上同样安装）

注释：其实两个的安装配置基本上是一样的。

```
[root@elk-node2 src]# mkdir -p /data/es-data
[root@elk-node2 ~]# cat /etc/elasticsearch/elasticsearch.yml
cluster.name: huanqiu
node.name: elk-node2
path.data: /data/es-data
path.logs: /var/log/elasticsearch/
bootstrap.mlockall: true
```

```

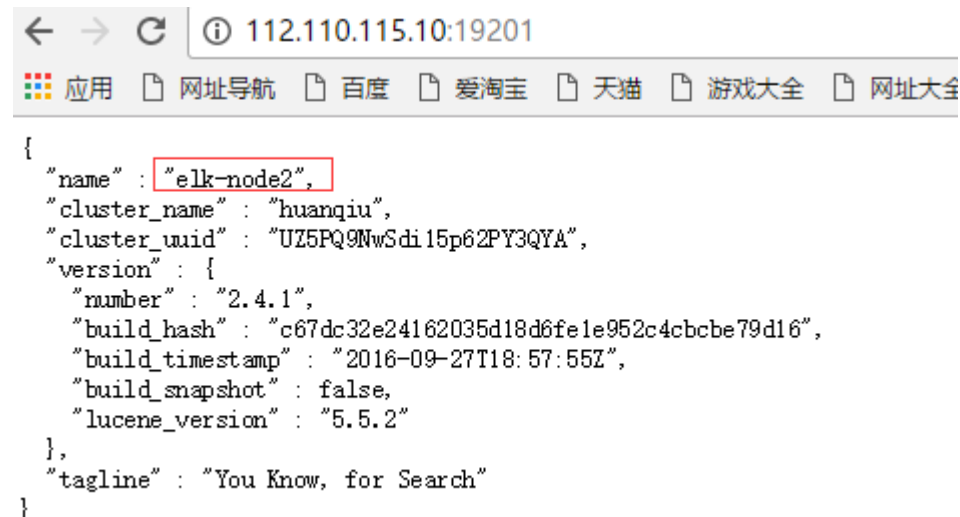
network.host: 0.0.0.0
http.port: 9200
discovery.zen.ping.multicast.enabled: false
discovery.zen.ping.unicast.hosts: ["192.168.1.160", "192.168.1.161"]
# 修改权限配置
[root@elk-node2 src]# chown -R elasticsearch.elasticsearch /data/
# 启动服务
[root@elk-node2 src]# systemctl start elasticsearch
[root@elk-node2 src]# systemctl status elasticsearch
• elasticsearch.service - Elasticsearch
Loaded: loaded (/usr/lib/systemd/system/elasticsearch.service; enabled; vendor
       preset: disabled)
Active: active (running) since Wed 2016-09-28 16:49:41 CST; 1 weeks 3 days
       ago
Docs: http://www.elastic.co
Process: 17798 ExecStartPre=/usr/share/elasticsearch/bin/elasticsearch-systemd
       -pre-exec (code=exited, status=0/SUCCESS)
Main PID: 17800 (java)
CGroup: /system.slice/elasticsearch.service
└─17800 /bin/java -Xms256m -Xmx1g -Djava.awt.headless=true -XX:+UsePar
       NewGC -XX:+UseConcMarkSweepGC -XX:CMSInitiatingOccupancyFra...
Oct 09 13:42:22 elk-node2 elasticsearch[17800]: [2016-10-09 13:42:22,295]
[WARN ][transport ] [elk-node2] Transport res...943817]
Oct 09 13:42:23 elk-node2 elasticsearch[17800]: [2016-10-09 13:42:23,111]
[WARN ][transport ] [elk-node2] Transport res...943846]
.....
.....
# 查看端口
[root@elk-node2 src]# netstat -antlp|egrep "9200|9300"
tcp6 0 0 :::9200 :::* LISTEN 2928/java
tcp6 0 0 :::9300 :::* LISTEN 2928/java
tcp6 0 0 127.0.0.1:48200 127.0.0.1:9300 TIME_WAIT -
tcp6 0 0 ::1:41892 ::1:9300 TIME_WAIT -
*****
*****
通过命令的方式查看 elk-node2 数据（在 112.110.115.10 宿主机或其他外网服务器上查看，
如下）
[root@kvm-server ~]# curl -i -XGET 'http://192.168.1.161:9200/_count?pretty'
-d '{"query":{"match_all":{"}}}'
HTTP/1.1 200 OK
Content-Type: application/json; charset=UTF-8
Content-Length: 95
{
"count" : 1,

```

```
"_shards" : {  
  "total" : 5,  
  "successful" : 5,  
  "failed" : 0  
}
```

然后通过 web 访问 elk-node2

<http://112.110.115.10:19201/>

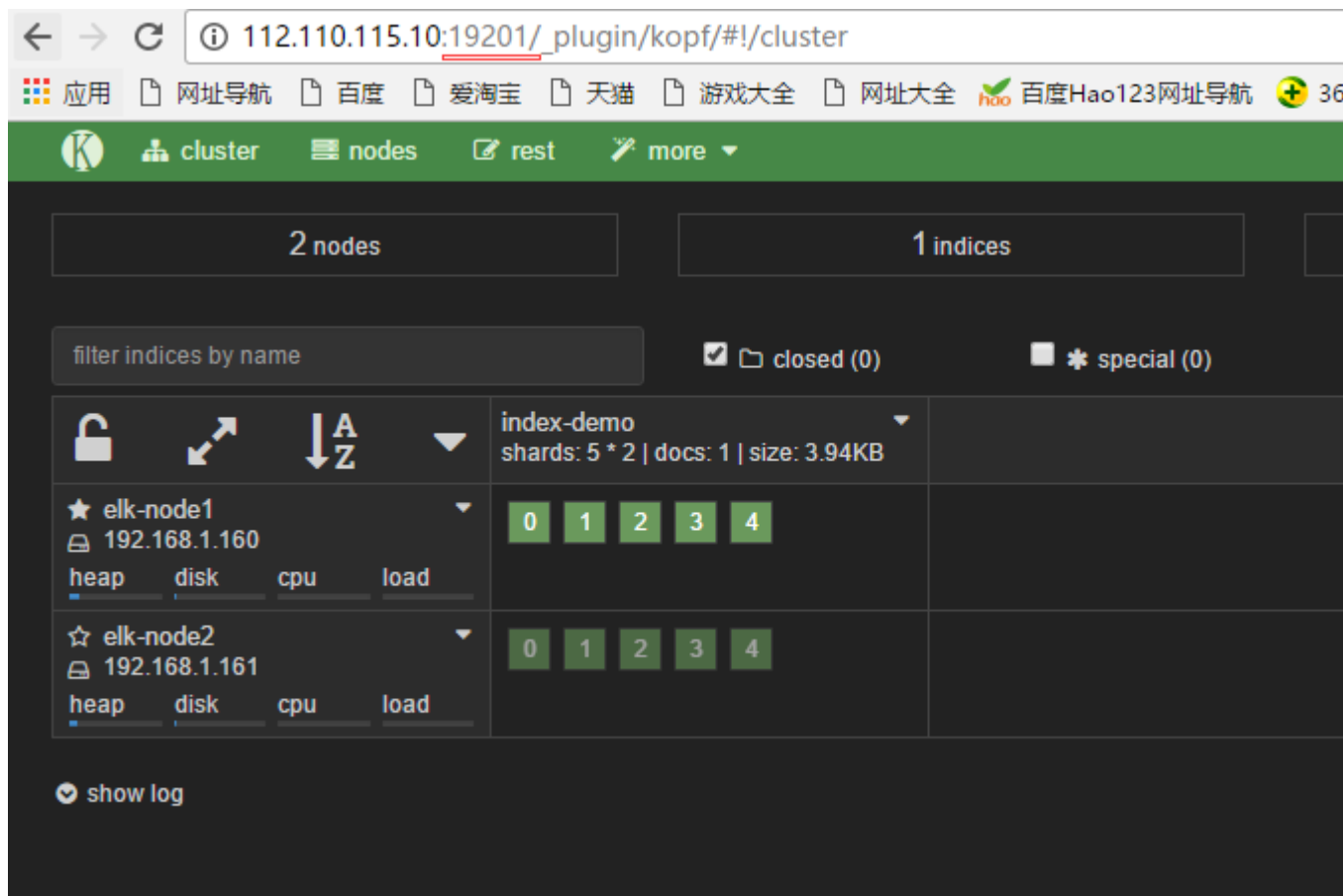


访问两个插件:

http://112.110.115.10:19201/_plugin/head/

http://112.110.115.10:19201/_plugin/kopf#!/cluster





(2) Logstash 安装配置 (这个在客户机上是要安装的。elk-node1 和 elk-node2 都安装)
基础环境安装 (客户端安装 logstash, 收集到的数据写入到 **elasticsearch** 里, 就可以登陆 **logstash** 界面查看到了)

1) 下载并安装 GPG Key

```
[root@elk-node1 ~]# rpm --import https://packages.elastic.co/GPG-KEY-elastic
search
```

2) 添加 yum 仓库

```
[root@hadoop-node1 ~]# vim /etc/yum.repos.d/logstash.repo
[logstash-2.1]
```

```
name=Logstash repository for 2.1.x packages
```

```
baseurl=http://packages.elastic.co/logstash/2.1/centos
```

```
gpgcheck=1
```

```
gpgkey=http://packages.elastic.co/GPG-KEY-elasticsearch
```

```
enabled=1
```

3) 安装 logstash

```
[root@elk-node1 ~]# yum install -y logstash
```

4) logstash 启动

```
[root@elk-node1 ~]# systemctl start elasticsearch
```

```
[root@elk-node1 ~]# systemctl status elasticsearch
```

```
• elasticsearch.service - Elasticsearch
```

```
Loaded: loaded (/usr/lib/systemd/system/elasticsearch.service; disabled; vendo
```

r preset: disabled)
Active: active (running) since Mon 2016-11-07 18:33:28 CST; 3 days ago
Docs: <http://www.elastic.co>
Main PID: 8275 (java)
CGroup: /system.slice/elasticsearch.service
└─8275 /bin/java -Xms256m -Xmx1g -Djava.awt.headless=true -XX:+UseParNewGC -XX:+UseConcMarkSweepGC -XX:CMSInitiatingOccupancyFrac...

.....

.....

数据的测试

1) 基本的输入输出

```
[root@elk-node1 ~]# /opt/logstash/bin/logstash -e 'input { stdin{} } output { stdout{} }'
```

Settings: Default filter workers: 1

Logstash startup completed

hello #输入

这个

2016-11-11T06:41:07.690Z elk-node1 hello #输出这个

wangshibo #输入这

个

2016-11-11T06:41:10.608Z elk-node1 wangshibo #输出这个

2) 使用 rubydebug 详细输出

```
[root@elk-node1 ~]# /opt/logstash/bin/logstash -e 'input { stdin{} } output { stdout{ codec => rubydebug } }'
```

Settings: Default filter workers: 1

Logstash startup completed

hello #输入

这个

{ #输出

下面信息

```
    "message" => "hello",
    "@version" => "1",
    "@timestamp" => "2016-11-11T06:44:06.711Z",
    "host" => "elk-node1"
```

}

wangshibo #输入这个

{ #输出下

面信息

```
    "message" => "wangshibo",
    "@version" => "1",
    "@timestamp" => "2016-11-11T06:44:11.270Z",
    "host" => "elk-node1"
```

}

3) 把内容写到 elasticsearch 中

```
[root@elk-node1 ~]# /opt/logstash/bin/logstash -e 'input { stdin{} } output { elasticsearch { hosts => ["192.168.1.160:9200"]} }'
```

Settings: Default filter workers: 1

Logstash startup completed

#输入下面的测试数据

123456

wangshibo

huanqiu

hahaha

使用 `rubydebug` 和写到 `elasticsearch` 中的区别：其实就在于后面标准输出的区别，前者使用 `codec`；后者使用 `elasticsearch`

写到 `elasticsearch` 中在 `logstash` 中查看，如下图：

注意：

master 收集到日志后，会把一部分数据碎片到 **salve** 上（随机的一部分数据），**master** 和 **slave** 又都会各自做副本，并把副本放到对方机器上，这样就保证了数据不会丢失。

如下，**master** 收集到的数据放到了自己的第 1,3 分片上，其他的放到了 **slave** 的第 0,2,4 分片上。



← → ↻ http://112.110.115.10:19200/_plugin/head/

应用 网址导航 百度 爱淘宝 天猫 游戏大全 网址大全 hao 百度Hao123网址导航 36

Elasticsearch

<http://103.10.86.7:19200/> [连接](#) huanqiu 集群健康

概览 索引 **数据浏览** 基本查询 [\[+\]](#) 复合查询 [\[+\]](#)

数据浏览

所有索引 ▼

索引

- index-demo
- logstash-2016.11.11**

类型

- _default_
- logs
- test

查询 5 个分片中用的 5 个, 3 命中, 耗时 0.012 秒

_index	_type	_id
logstash-2016.11.11	logs	AVhSMhv85S5U0
logstash-2016.11.11	logs	AVhSMjOs5S5U0
logstash-2016.11.11	logs	AVhSMjhS5S5U0

← → ↻ 112.110.115.10:19200/_plugin/kopf/#!/cluster

应用 网址导航 百度 爱淘宝 天猫 游戏大全 网址大全 hao 百度Hao123网址导航 36

cluster nodes rest more ▼

2 nodes 2 indices

filter indices by name ☐ closed (0) ☐ special (0)

	index-demo shards: 5 * 2 docs: 1 size: 3.94KB	logstash-2016.11.11 shards: 5 * 2 docs: 3 size: 12.76KB
★ elk-node1 192.168.1.160 heap disk cpu load	0 1 2 3 4	0 1 2 3 4
☆ elk-node2 192.168.1.161 heap disk cpu load	0 1 2 3 4	0 1 2 3 4

show log

4) 即写到 `elasticsearch` 中又写在文件中一份

```
[root@elk-node1 ~]# /opt/logstash/bin/logstash -e 'input { stdin{} } output
{ elasticsearch { hosts => ["192.168.1.160:9200"]} stdout{ codec => rubyde
bug}}'
```

Settings: Default filter workers: 1

Logstash startup completed

huanqiupc

```
{
  "message" => "huanqiupc",
  "@version" => "1",
  "@timestamp" => "2016-11-11T07:27:42.012Z",
  "host" => "elk-node1"
}
```

wangshiboqun

```
{
  "message" => "wangshiboqun",
  "@version" => "1",
  "@timestamp" => "2016-11-11T07:27:55.396Z",
  "host" => "elk-node1"
}
```

以上文本可以长期保留、操作简单、压缩比大。下面登陆 [elasticsearch](#) 界面中查看;

112.110.115.10:19200/_plugin/head/

Elasticsearch [http://103.10.86.7:19200/](#) [连接](#) huanqiu 集群健康

概览 索引 数据浏览 基本查询 [\[+\]](#) 复合查询 [\[+\]](#)

数据浏览

所有索引 ▼

索引

- index-demo
- logstash-2016.11.11

类型

- _default_
- logs
- test

查询 10 个分片中用的 10 个. 6 命中. 耗时 0.015 秒

_index	_type	_id
logstash-2016.11.11	logs	AVhSMhv85S5U0
logstash-2016.11.11	logs	AVhSMjOs5S5U0
logstash-2016.11.11	logs	AVhSSSk75S5U0
logstash-2016.11.11	logs	AVhSMjhS5S5U0
index-demo	test	AVg-OkP4s1YzW2
logstash-2016.11.11	logs	AVhSSPX45S5U0

logstash 的配置和文件的编写

1) logstash 的配置

简单的配置方式:

```
[root@elk-node1 ~]# vim /etc/logstash/conf.d/01-logstash.conf
```

```
input { stdin { } }
```

```
output {
```

```
  elasticsearch { hosts => ["192.168.1.160:9200"] }
```

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

```
}
```

它的执行:

```
[root@elk-node1 ~]# /opt/logstash/bin/logstash -f /etc/logstash/conf.d/01-logst  
ash.conf
```

Settings: Default filter workers: 1

Logstash startup completed

beijing

#输入内容

{

#输出下面信息

"message" => "beijing",

"@version" => "1",

"@timestamp" => "2016-11-11T07:41:48.401Z",

"host" => "elk-node1"

}

参考内容:

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

<https://www.elastic.co/guide/en/logstash/current/configuration-file-structure.htm>

|

← → ↻ ⓘ 112.110.115.10:19200/_plugin/head/

应用 网址导航 百度 爱淘宝 天猫 游戏大全 网址大全 hoo 百度Hao123网址导航 + 36

Elasticsearch

http://103.10.86.7:19200/ 连接 huanqiu 集群健康

概览 索引 数据浏览 基本查询 [+]
复合查询 [+]

数据浏览

所有索引 ▾

索引

- index-demo
- logstash-2016.11.11

类型

- _default_
- logs
- test

字段

查询 10 个分片中用的 10 个, 7 命中, 耗时 0.014 秒

_index	_type	_id
logstash-2016.11.11	logs	AVhSMhv85S5U0
logstash-2016.11.11	logs	AVhSMjOs5S5U0
logstash-2016.11.11	logs	AVhSVd165S5U0
logstash-2016.11.11	logs	AVhSSSk75S5U0
logstash-2016.11.11	logs	AVhSMjhS5S5U0
index-demo	test	AVg-OkP4s1YzW2
logstash-2016.11.11	logs	AVhSSPX45S5U0

2) 收集系统日志

```
1 [root@elk-node1 ~]# vim file.conf
2 input {
3     file {
4         path => "/var/log/messages"
5         type => "system"
6         start_position => "beginning"
7     }
8 }
```

9	
10	output {
11	elasticsearch {
12	hosts => ["192.168.1.160:9200"]
13	index => "system-%{+YYYY.MM.dd}"
14	}
15	}

执行上面日志信息的收集，如下，这个命令会一直在执行中，表示日志在监控收集；如果中断，就表示日志不在收集！所以需要放在后台执行~

[root@elk-node1 ~]# /opt/logstash/bin/logstash -f file.conf &

登陆 [elasticsearch](#) 界面，查看本机系统日志的信息：

The screenshot shows the Kibana interface for Elasticsearch. The browser address bar displays `112.110.115.10:19200/_plugin/head/`. The page title is "Elasticsearch" with a URL `http://103.10.86.7:19200/`. Navigation tabs include "概览" (Overview), "索引" (Index), "数据浏览" (Data Explorer), "基本查询" (Basic Search), and "复合查询" (Advanced Search). The "集群概览" (Cluster Overview) section is active, showing "集群排序" (Cluster Sort), "Sort Indices", "View Aliases", and an "Index Filter".

Two index patterns are displayed:

- system-2016.11.11**: size: 525ki (1.03Mi), docs: 1,627 (3,254). It has "信息" (Info) and "动作" (Actions) buttons.
- logstash-2016.11.11**: size: 24.8ki (49.5ki), docs: 6 (12). It also has "信息" (Info) and "动作" (Actions) buttons.

Below the index patterns, the shard status for two nodes is shown:

- elk-node1** (star icon): Shards 0, 1, 2, 3, 4 are all green, indicating they are in a healthy state.
- elk-node2** (circle icon): Shards 0, 1, 2, 3, 4 are all green, indicating they are in a healthy state.

Elasticsearch

<http://103.10.86.7:19200/>

连接

huanqiu

集群健

[概览](#) [索引](#) [数据浏览](#) [基本查询](#) [复合查询](#)

数据浏览

所有索引 ▼

索引

index-demo

logstash-2016.11.11

system-2016.11.11

类型六

default

logs

system

test

字段

► @timestamp

► @version

▶ geoip.jp

- ▶ `geoip.latitude`

- ▶ `geoip.longitude`

► host

► mesa

► message

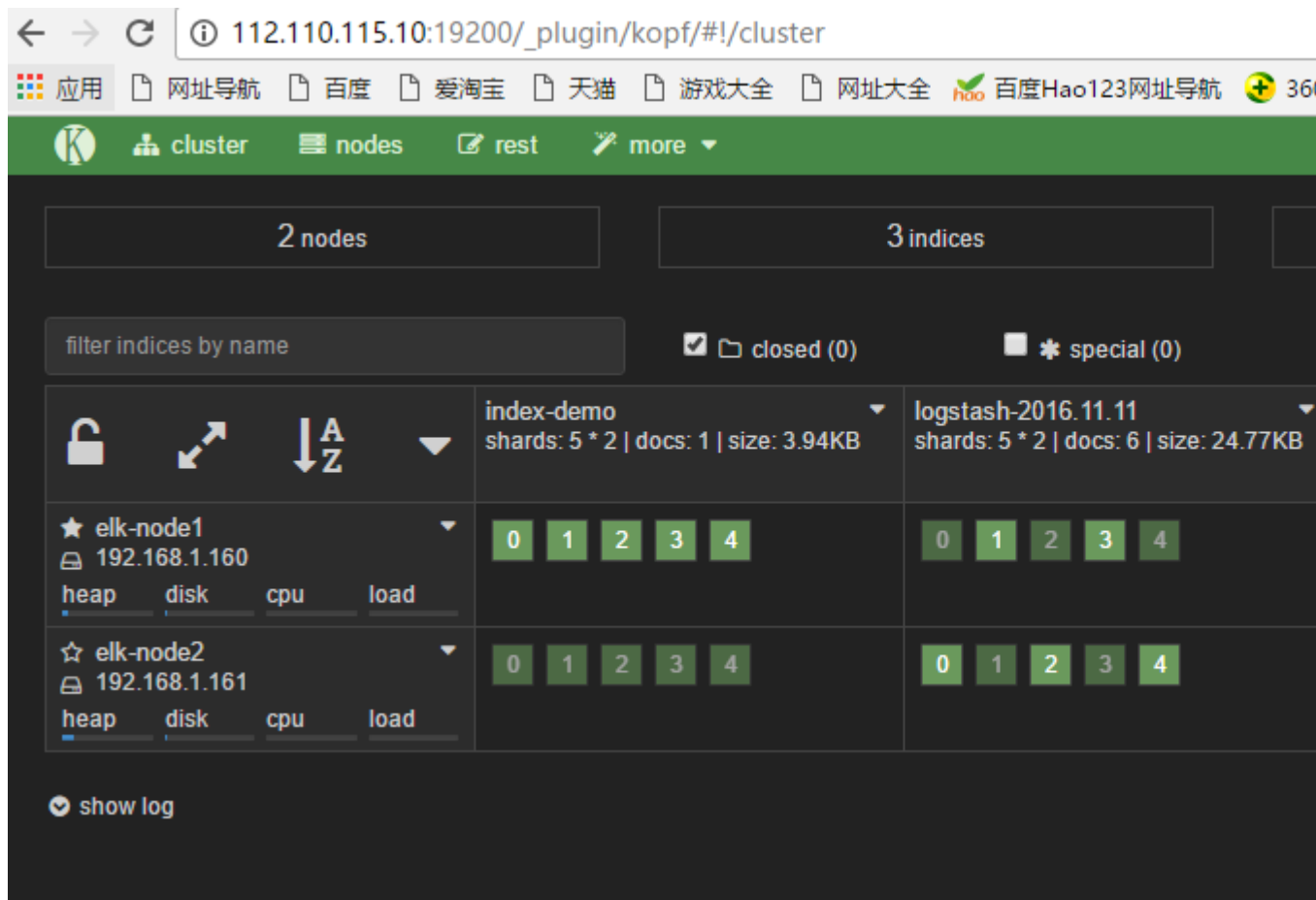
► path

► type

user

查询 10 个分片中用的 10 个, 1633 命中, 耗时 0.031 秒

[illegible]



参考内容:

<https://www.elastic.co/guide/en/logstash/current/plugins-outputs-elasticsearch.html>

3) 收集 java 日志，其中包含上面讲到的日志收集

```
1 [root@elk-node1 ~]# vim file.conf
2 input {
3     file {
4         path => "/var/log/messages"
5         type => "system"
6         start_position => "beginning"
7     }
8 }
9
10 input {
11     file {
12         path => "/var/log/elasticsearch/huanqiu.log"
```

```

13         type => "es-error"
14         start_position => "beginning"
15     }
16 }
17
18
19 output {
20
21     if [type] == "system"{
22         elasticsearch {
23             hosts => ["192.168.1.160:9200"]
24             index => "system-%{+YYYY.MM.dd}"
25         }
26     }
27
28     if [type] == "es-error"{
29         elasticsearch {
30             hosts => ["192.168.1.160:9200"]
31             index => "es-error-%{+YYYY.MM.dd}"
32         }
33     }
34 }

```

注意：

如果你的日志中有 **type** 字段 那你就不能在 **conf** 文件中使用 **type**

执行如下命令收集：

[root@elk-node1 ~]# /opt/logstash/bin/logstash -f file.conf &

登陆 [elasticsearch](#) 界面，查看数据：

← → ↻ ⓘ 112.110.115.10:19200/_plugin/head/

应用 网址导航 百度 爱淘宝 天猫 游戏大全 网址大全 hao 百度Hao123网址导航 360

Elasticsearch

http://103.10.86.7:19200/ 连接 huanqiu 集群健康

概览 索引 数据浏览 基本查询 [+] 复合查询 [+] 集群概览 集群排序 ▾ Sort Indices ▾ View Aliases ▾ Index Filter

system-2016.11.11

size: 558ki (1.09Mi)
docs: 1,645 (3,290)

信息 ▾ 动作 ▾

logstash-2016.11.11

size: 24.8ki (49.5ki)
docs: 6 (12)

信息 ▾ 动作 ▾

★ **elk-node1**

信息 ▾ 动作 ▾

0 1 2 3 4

● **elk-node2**

信息 ▾ 动作 ▾

0 1 2 3 4

Elasticsearch

http://103.10.86.7:19200/ 连接 huanqiu 集群健康

概览 索引 数据浏览 基本查询 [+] 复合查询 [+] 数据浏览

所有索引 ▾

索引

es-error-2016.11.11

index-demo

logstash-2016.11.11

system-2016.11.11

类型

default

es-error

logs

system

test

字段

▶ @timestamp ?

▶ @version

▶ geoip.ip

查询 5 个分片中用的 5 个. 12 命中. 耗时 0.013 秒

_index	_type	_id
es-error-2016.11.11	es-error	AVhSc9m85S5U0
es-error-2016.11.11	es-error	AVhSc9m85S5U0
es-error-2016.11.11	es-error	AVhSc83c5S5U0I
es-error-2016.11.11	es-error	AVhSc83c5S5U0I
es-error-2016.11.11	es-error	AVhSc83c5S5U0I
es-error-2016.11.11	es-error	AVhSc9m85S5U0
es-error-2016.11.11	es-error	AVhSc83c5S5U0I
es-error-2016.11.11	es-error	AVhSc83c5S5U0I
es-error-2016.11.11	es-error	AVhSc83c5S5U0I
es-error-2016.11.11	es-error	AVhSc83c5S5U0I
es-error-2016.11.11	es-error	AVhSc83c5S5U0I
es-error-2016.11.11	es-error	AVhSc9m85S5U0

参考内容:

<https://www.elastic.co/guide/en/logstash/current/event-dependent-configuration.>

html

```
=====
=====
```

有个问题：

每个报错都给收集成一行了，不是按照一个报错，一个事件模块收集的。

下面将行换成事件的方式展示：

```
1 [root@elk-node1 ~]# vim multiline.conf
2 input {
3     stdin {
4         codec => multiline {
5             pattern => "^\[\"
6             negate => true
7             what => "previous"
8         }
9     }
10 }
11 output {
12     stdout {
13         codec => "rubydebug"
14     }
15 }
```

执行命令：

```
1 [root@elk-node1 ~]# /opt/logstash/bin/logstash -f multiline.conf
2 Settings: Default filter workers: 1
3 Logstash startup completed
4 123
5 456
6 [123
7 {
8     "@timestamp" => "2016-11-11T09:28:56.824Z",
9     "message" => "123\n456",
10    "@version" => "1",
11    "tags" => [
12        [0] "multiline"
13    ],
14    "host" => "elk-node1"
15 }
16 123]
17 [456]
18 {
19    "@timestamp" => "2016-11-11T09:29:09.043Z",
20    "message" => "[123\n123]",
21    "@version" => "1",
```

```

22         "tags" => [
23             [0] "multiline"
24         ],
25         "host" => "elk-node1"
26     }

```

在没有遇到【】的时候，系统不会收集，只有遇见【】的时候，才算是一个事件，才收集起来。

```

=====
=====

```

参考内容

<https://www.elastic.co/guide/en/logstash/current/plugins-codecs-multiline.html>

```

=====
=====

```

(3) Kibana 安装配置

1) kibana 的安装:

```

[root@elk-node1 ~]# cd /usr/local/src
[root@elk-node1 src]# wget https://download.elastic.co/kibana/kibana/kibana-4.3.1-linux-x64.tar.gz
[root@elk-node1 src]# tar zxf kibana-4.3.1-linux-x64.tar.gz
[root@elk-node1 src]# mv kibana-4.3.1-linux-x64 /usr/local/
[root@elk-node1 src]# ln -s /usr/local/kibana-4.3.1-linux-x64/ /usr/local/kibana

```

2) 修改配置文件:

```

[root@elk-node1 config]# pwd
/usr/local/kibana/config
[root@elk-node1 config]# cp kibana.yml kibana.yml.bak
[root@elk-node1 config]# vim kibana.yml

```

server.port: 5601

server.host: "0.0.0.0"

elasticsearch.url: "http://192.168.1.160:9200"

kibana.index: ".kibana" **#注意这个.Kibana索引用来存储数据,千万不要删除了它。它是将 es 数据通过 kibana 进行 web 展示的关键。这个配置后,在 es 的 web 界面里就会看到这个.kibana 索引。**

因为他一直运行在前台,要么选择开一个窗口,要么选择使用 screen。

安装并使用 screen 启动 kibana:

```

[root@elk-node1 ~]# yum -y install screen
[root@elk-node1 ~]# screen #这样就另开启了一个终端窗口
[root@elk-node1 ~]# /usr/local/kibana/bin/kibana
log [18:23:19.867] [info][status][plugin:kibana] Status changed from uninitialized to green - Ready
log [18:23:19.911] [info][status][plugin:elasticsearch] Status changed from uninitialized to yellow - Waiting for Elasticsearch
log [18:23:19.941] [info][status][plugin:kbn_vislib_vis_types] Status changed from uninitialized to green - Ready
log [18:23:19.953] [info][status][plugin:markdown_vis] Status changed from u

```

ninitialized to green - Ready

log [18:23:19.963] [info][status][plugin:metric_vis] Status changed from uninitialized to green - Ready

log [18:23:19.995] [info][status][plugin:spyModes] Status changed from uninitialized to green - Ready

log [18:23:20.004] [info][status][plugin:statusPage] Status changed from uninitialized to green - Ready

log [18:23:20.010] [info][status][plugin:table_vis] Status changed from uninitialized to green - Ready

然后按 **ctrl+a+d** 组合键，这样在上面另启的 **screen** 屏里启动的 **kibana** 服务就一直运行在前台了....

```
[root@elk-node1 ~]# screen -ls
```

There is a screen on:

15041.pts-0.elk-node1 (Detached)

1 Socket in /var/run/screen/S-root.

(3) 访问 kibana: <http://112.110.115.10:15601/>

如下，如果是添加上面设置的 **java** 日志收集信息，则在下面填写 **es-error***；如果是添加上面设置的系统日志信息 **system***，以此类型(可以从 **logstash** 界面看到日志收集项)

← → ↻ ⓘ 112.110.115.10:15601/

应用 网址导航 百度 爱淘宝 天猫 游戏大全 网址大全 hao 百度Hao123网址导航 + 36

kibana

Discover Visualize Dashboard **Settings**

Indices Advanced Objects Status About

Index Patterns

Warning No default index pattern. You must select or create one to continue.

Configure an index pattern

In order to use Kibana you must configure at least one index pattern

☒ Index contains time-based events

☐ Use event times to create index names [DEPRECATED]

Index name or pattern

Patterns allow you to define dynamic index names using * as a wildcard

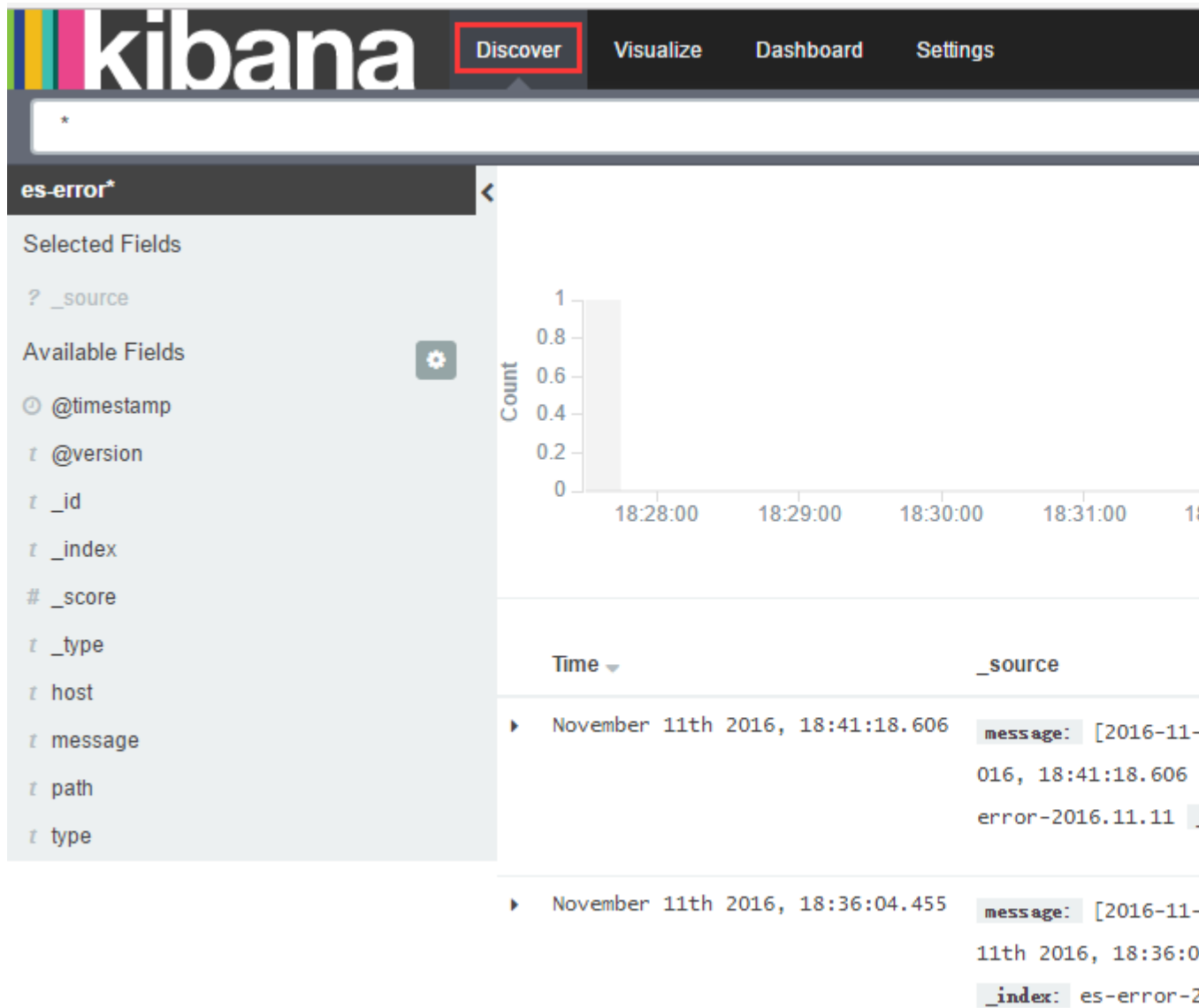
es-error*

Time-field name ⓘ refresh fields

@timestamp

Create

然后点击上面的 [Discover](#)，在 [Discover](#) 中查看：

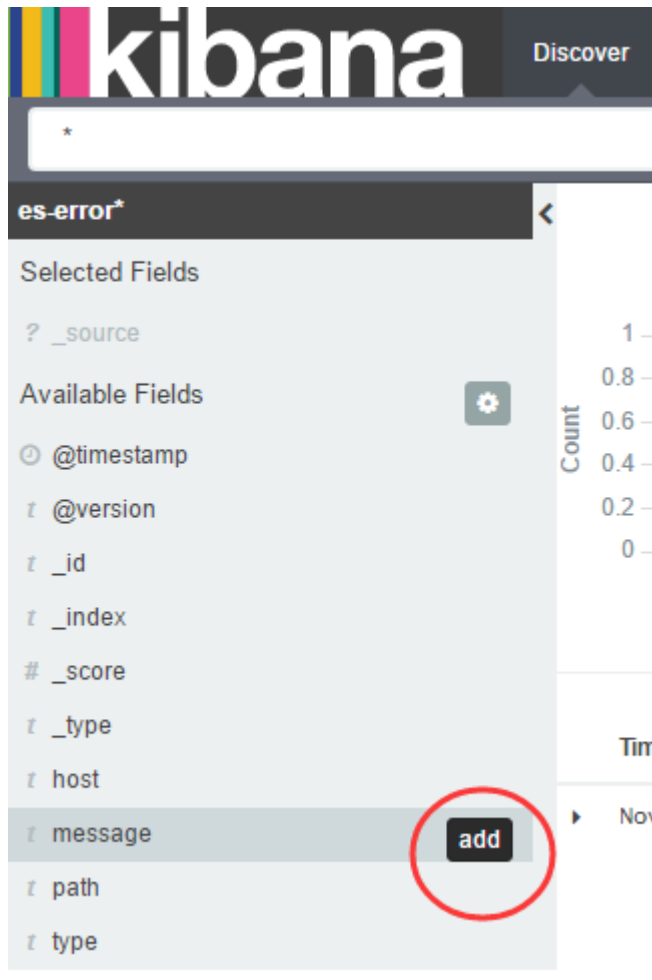


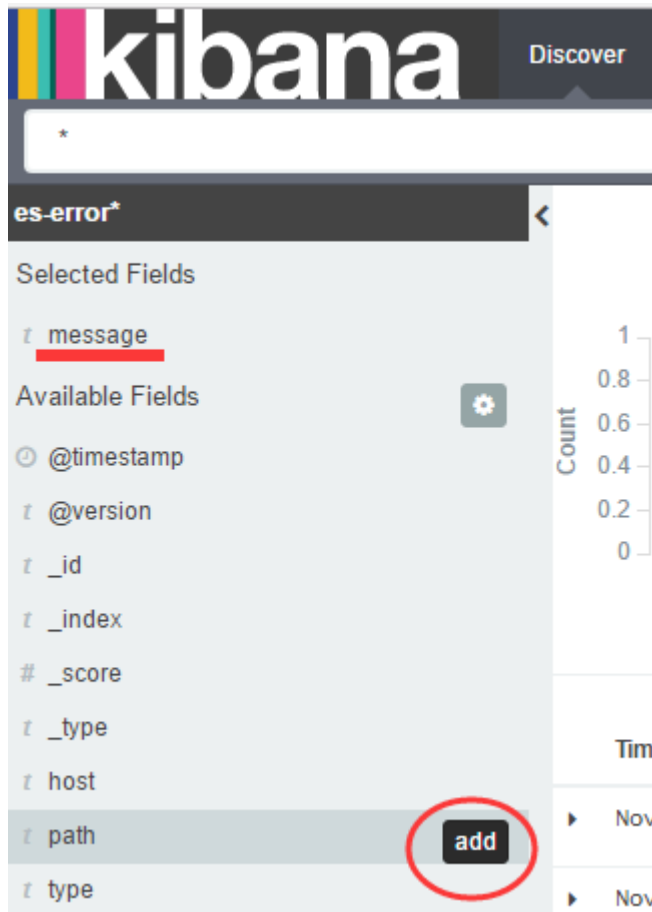
查看日志登陆，需要点击“Discover”-->“message”,点击它后面的“add”

注意：

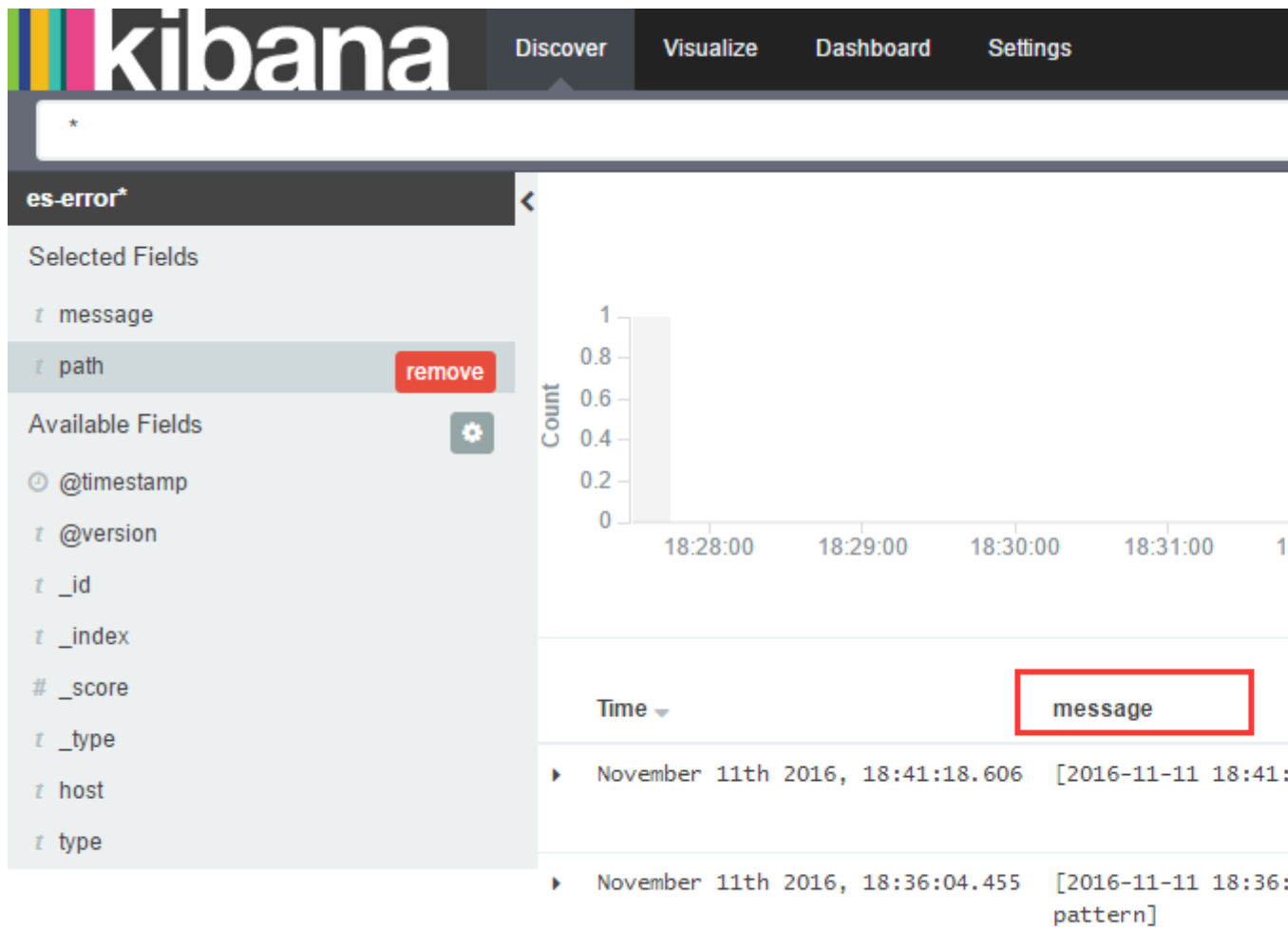
需要右边查看日志内容时带什么属性，就在左边点击相应属性后面的“add”

如下图，添加了 message 和 path 的属性：

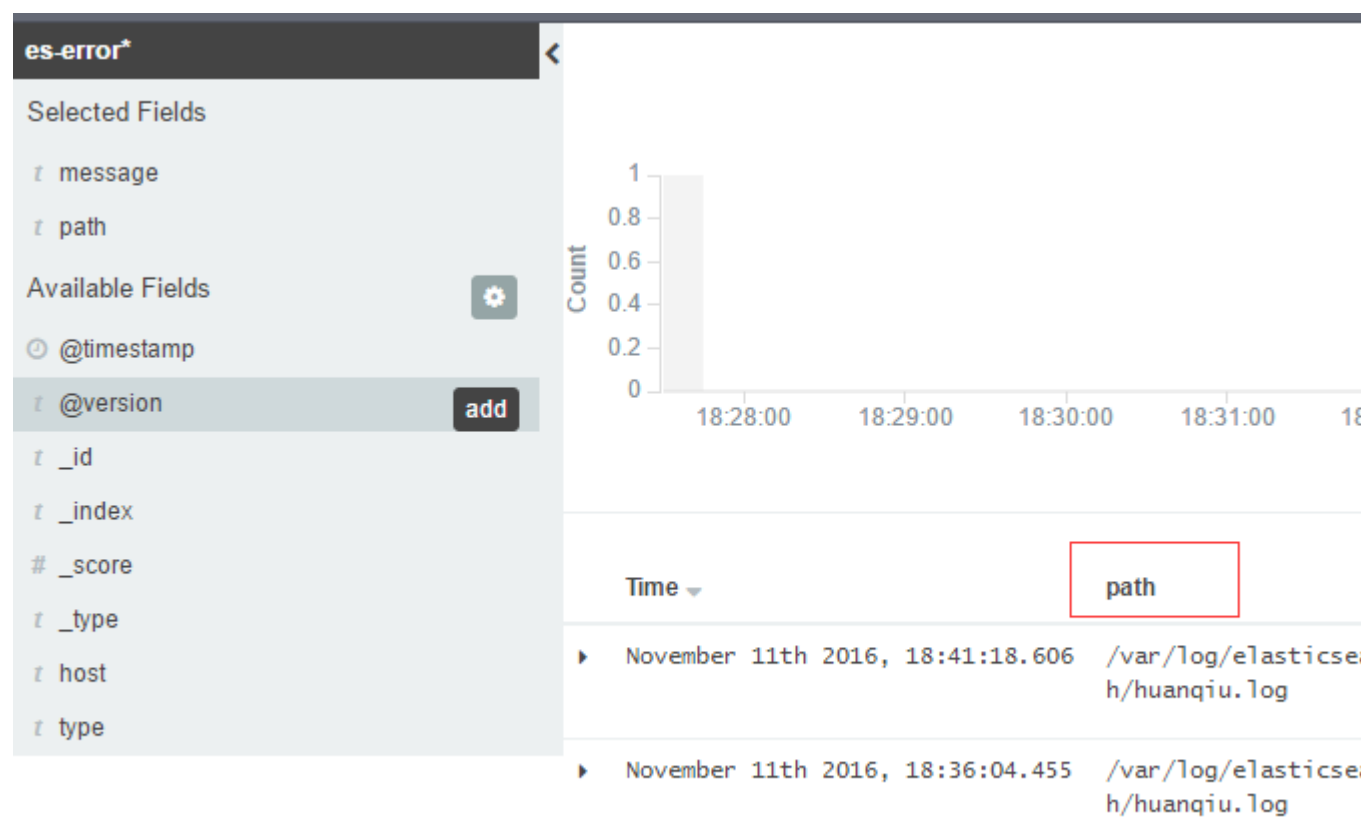
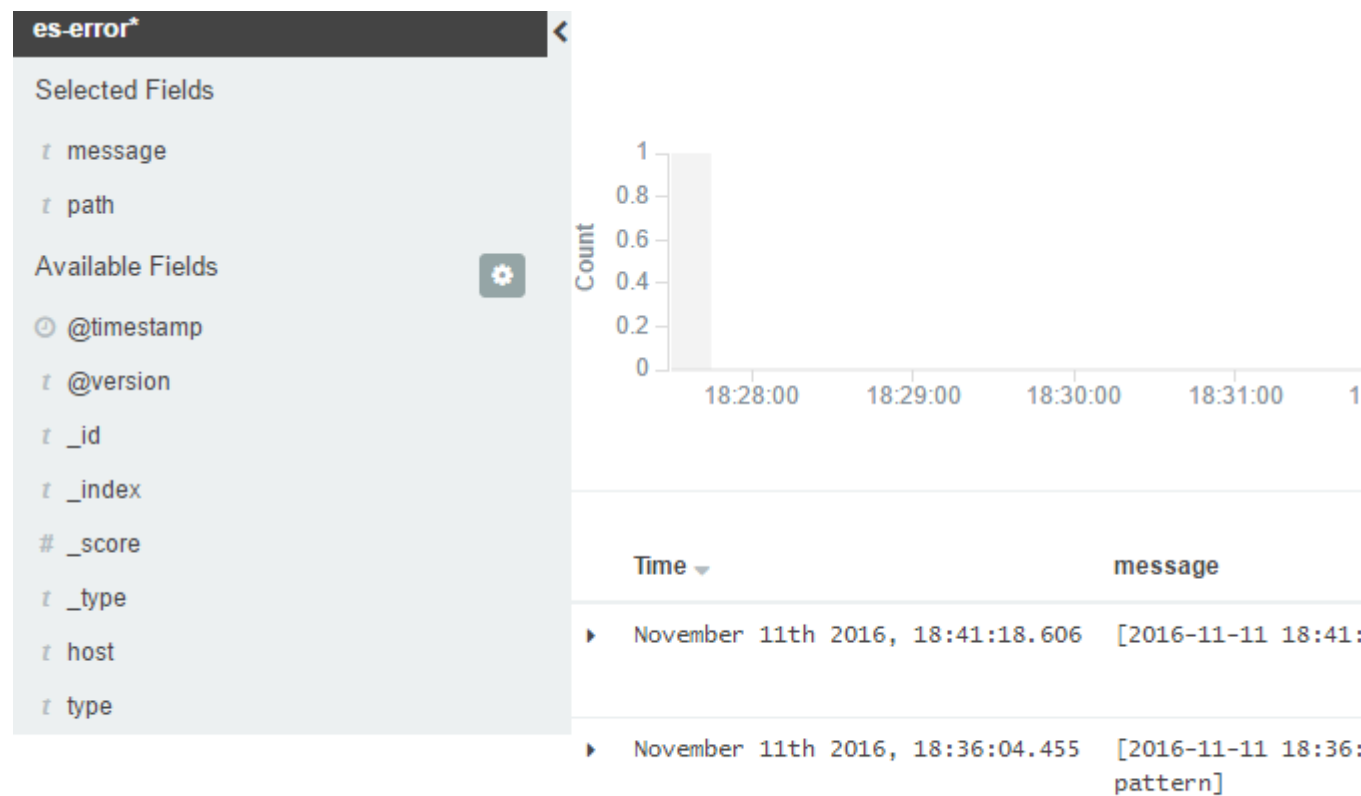




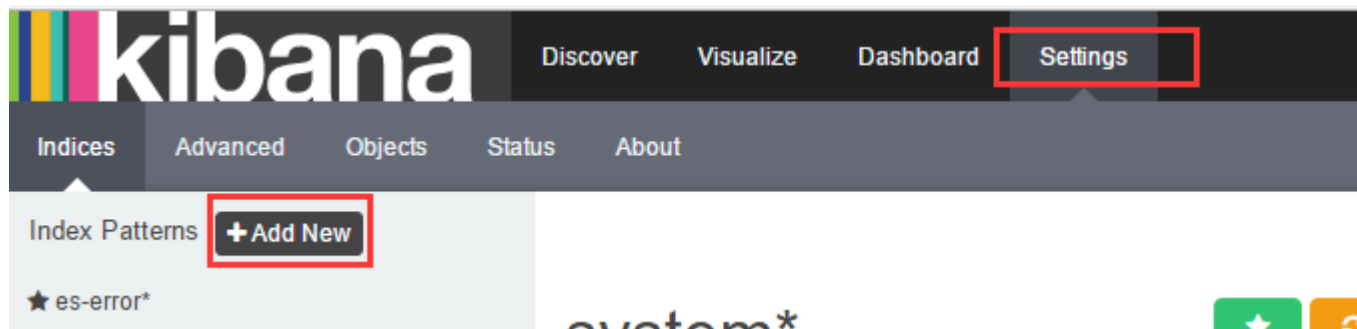
这样，右边显示的日志内容的属性就带了 message 和 path




点击右边日志内容属性后面隐藏的<<，就可将内容向前缩进



添加新的日志采集项，点击 **Settings->+Add New**，比如添加 **system** 系统日志。**注意后面的*不要忘了。**



kibana

DiscoverVisualizeDashboardSettings

IndicesAdvancedObjectsStatusAbout

Index Patterns + Add New


★ es-error*

system*

system*

★

This page lists every field in the **system*** index and the field's associated Elasticsearch's Mapping API [🔗](#)

kibana

DiscoverVisualizeDashboardSettings

*

es-error*

system*

Selected Fields

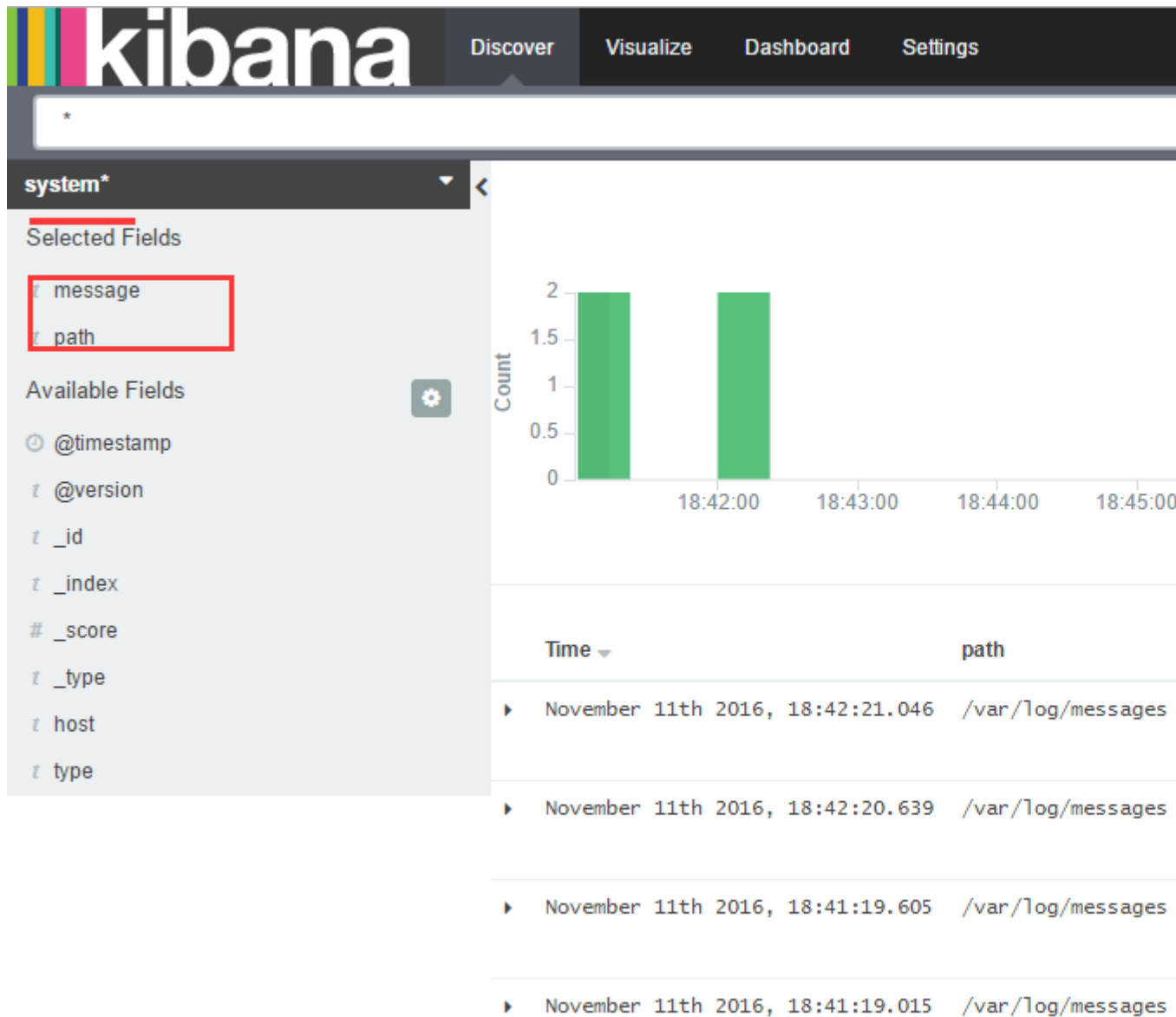
🔍 message

10.80.0.1

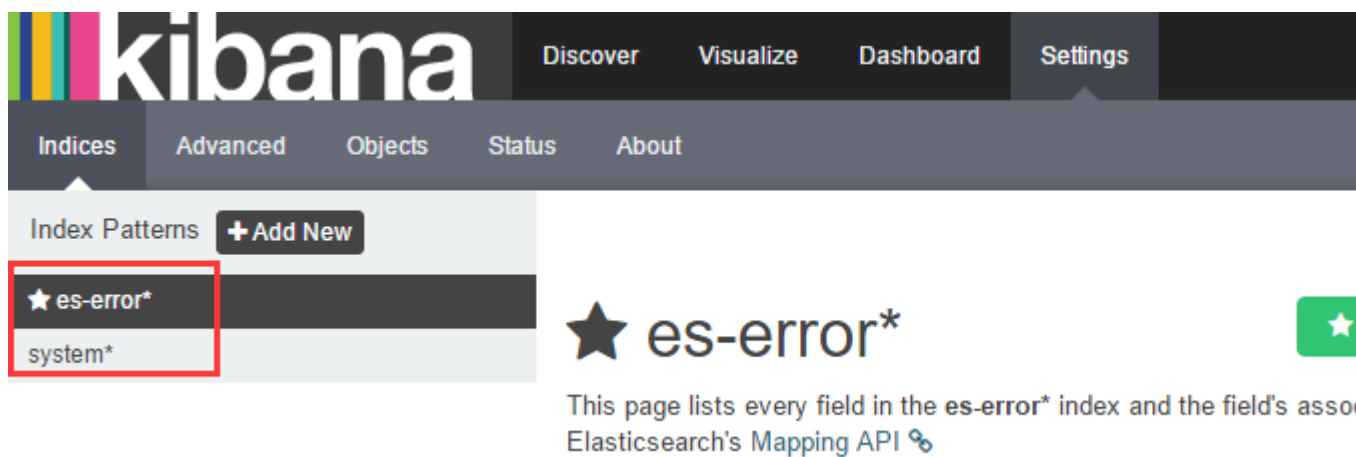
10.80.0.2

10.80.0.3

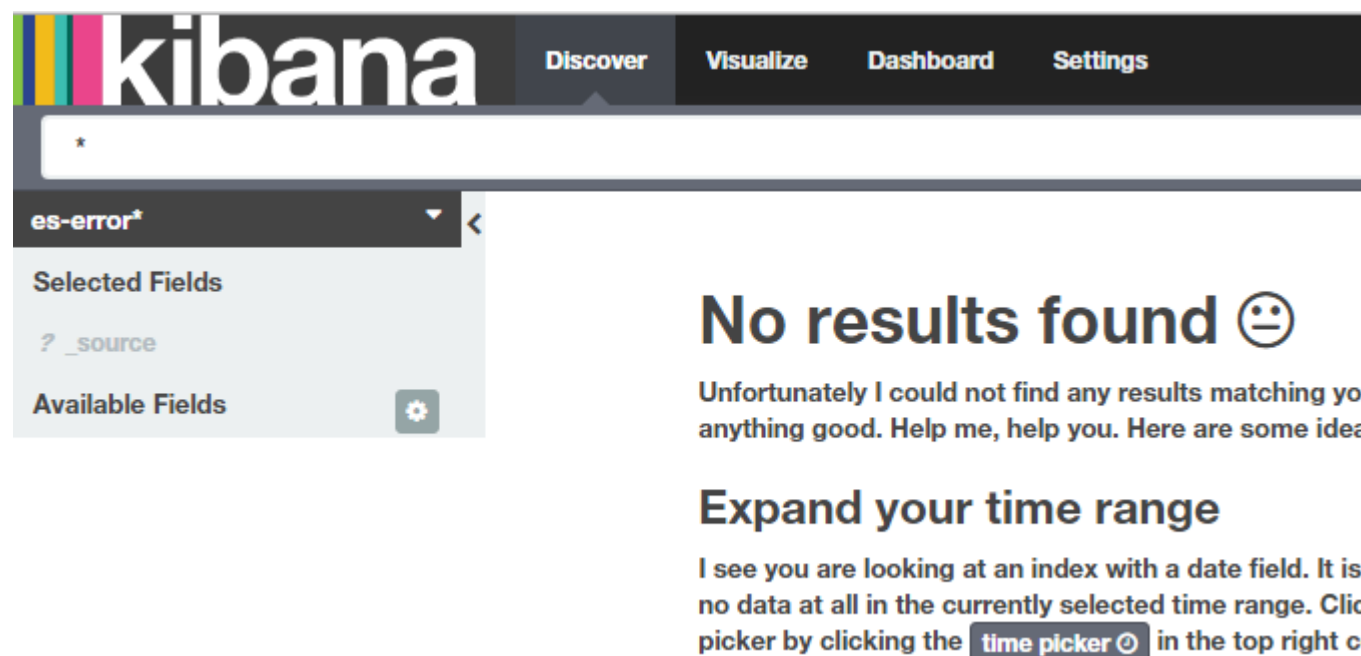
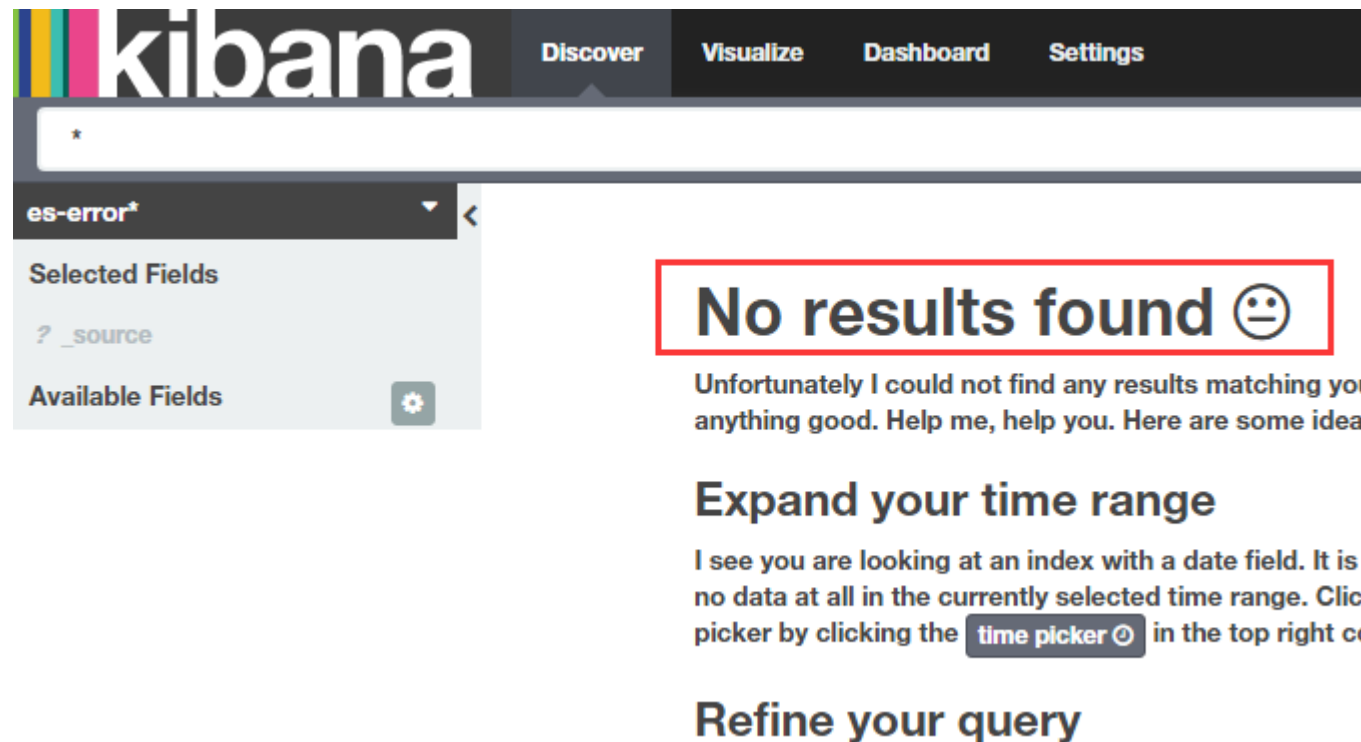
unt



删除 kibana 里的日志采集项，如下，点击删除图标即可。



如果打开 kibana 查看日志，发现没有日志内容，出现“No results found”，如下图所示，这说明要查看的日志在当前时间没有日志信息输出，可以点击右上角的时间钟来调试日志信息的查看。



Quick

Relative
Absolute

Today

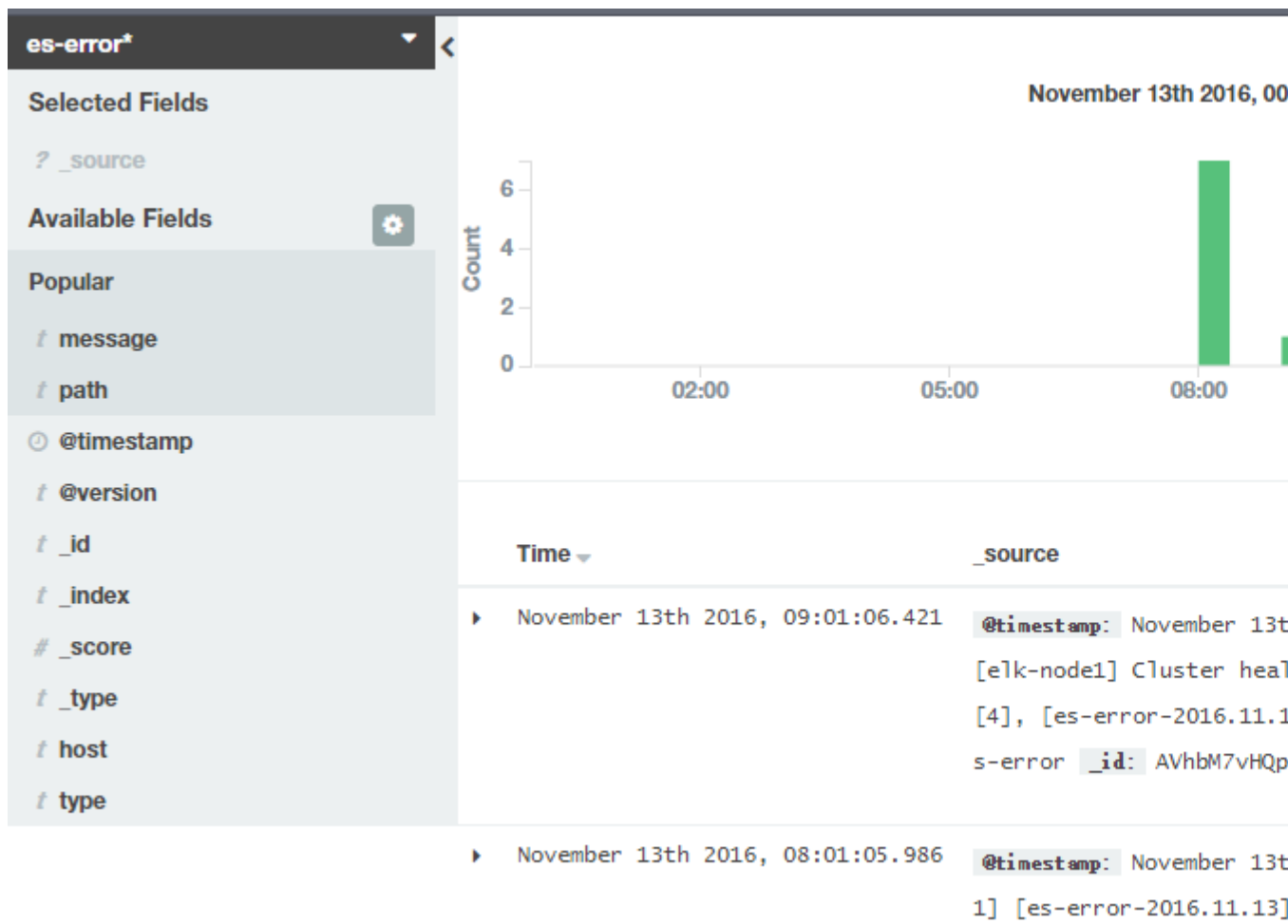
This week
This month
This year
The day so far
Week to date
Month to date
Year to date

Yesterday

Day before yesterday
This day last week
Previous week
Previous month
Previous year

Last 15 minutes

Last 30 minutes
Last 1 hour
Last 4 hours
Last 12 hours
Last 24 hours
Last 7 days



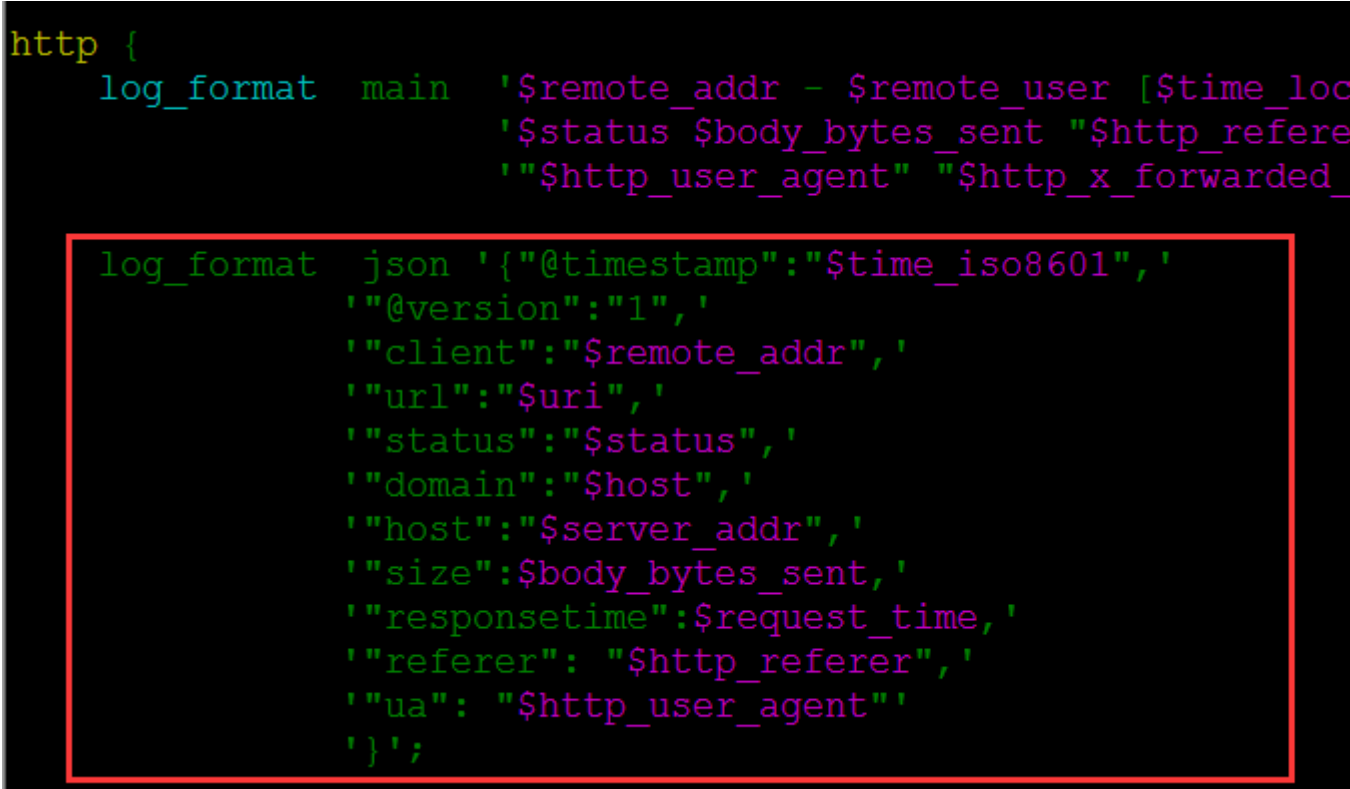
```

        "url": "$uri",'
        "status": "$status",'
        "domain": "$host",'
        "host": "$server_addr",'
        "size": $body_bytes_sent,'
        "responsetime": $request_time,'
        "referer": "$http_referer",'
        "ua": "$http_user_agent"
    }';
}

##### server 标签中
    access_log /var/log/nginx/access_json.log json;

```

截图如下：



```

http {
    log_format main '$remote_addr - $remote_user [$time_local]
                    '$status $body_bytes_sent "$http_referer"
                    "$http_user_agent" "$http_x_forwarded_for";

    log_format json '{"@timestamp": "$time_iso8601",'
                    '"@version": "1",'
                    '"client": "$remote_addr",'
                    '"url": "$uri",'
                    '"status": "$status",'
                    '"domain": "$host",'
                    '"host": "$server_addr",'
                    '"size": $body_bytes_sent,'
                    '"responsetime": $request_time,'
                    '"referer": "$http_referer",'
                    '"ua": "$http_user_agent"'
                    '}';
}

```

```

server {
    listen      80 default_server;
    listen      [::]:80 default_server;
    server_name _;
    root         /usr/share/nginx/html;
    access_log   /var/log/nginx/access_json.log json;
    # Load configuration files for the default server block
    include /etc/nginx/default.d/*.conf;

    location / {

    }

    error_page 404 /404.html;
        location = /40x.html {

    }

    error_page 500 502 503 504 /50x.html;
        location = /50x.html {

    }
}

```

启动 nginx 服务:

```

1  [root@elk-node1 ~]# systemctl start nginx
2  [root@elk-node1 ~]# systemctl status nginx
3  ● nginx.service - The nginx HTTP and reverse proxy server
4      Loaded: loaded (/usr/lib/systemd/system/nginx.service; disabled; vendor preset: enabled)
5      Active: active (running) since Fri 2016-11-11 19:06:55 CST; 3s ago
6      Process: 15119 ExecStart=/usr/sbin/nginx (code=exited, status=0/SUCCESS)
7      Process: 15116 ExecStartPre=/usr/sbin/nginx -t (code=exited, status=0/SUCCESS)
8      Process: 15114 ExecStartPre=/usr/bin/rm -f /run/nginx.pid (code=exited, status=0/SUCCESS)
9      Main PID: 15122 (nginx)
10     CGroup: /system.slice/nginx.service
11             └─15122 nginx: master process /usr/sbin/nginx
12             └─15123 nginx: worker process
13             └─15124 nginx: worker process
14
15  Nov 11 19:06:54 elk-node1 systemd[1]: Starting The nginx HTTP and reverse proxy server: [OK]
16  Nov 11 19:06:55 elk-node1 nginx[15116]: nginx: the configuration file /etc/nginx/nginx.conf is not readable (permission denied)
17  Nov 11 19:06:55 elk-node1 nginx[15116]: nginx: configuration file /etc/nginx/nginx.conf is not readable (permission denied)
18  Nov 11 19:06:55 elk-node1 systemd[1]: Started The nginx HTTP and reverse proxy server: [OK]

```

编写收集文件

这次使用 **json** 的方式收集:

```
1 [root@elk-node1 ~]# vim json.conf
2 input {
3     file {
4         path => "/var/log/nginx/access_json.log"
5         codec => "json"
6     }
7 }
8
9 output {
10     stdout {
11         codec => "rubydebug"
12     }
13 }
```

启动日志收集程序:

```
[root@elk-node1 ~]# /opt/logstash/bin/logstash -f json.conf
```

#或加个&放在后台执行

访问 **nginx** 页面 (在 **elk-node1** 的宿主机上执行访问页面的命令: **curl http://192.168.1.160**) 就会出现以下内容:

```
1 [root@elk-node1 ~]# /opt/logstash/bin/logstash -f json.conf
2 Settings: Default filter workers: 1
3 Logstash startup completed
4 {
5     "@timestamp" => "2016-11-11T11:10:53.000Z",
6     "@version" => "1",
7     "client" => "192.168.1.7",
8     "url" => "/index.html",
9     "status" => "200",
10    "domain" => "192.168.1.160",
11    "host" => "192.168.1.160",
12    "size" => 3700,
13    "responsetime" => 0.0,
14    "referer" => "-",
15    "ua" => "curl/7.19.7 (x86_64-redhat-linux-gnu) lib
16    zlib/1.2.3 libidn/1.18 libssh2/1.4.2",
17    "path" => "/var/log/nginx/access_json.log"
18 }
```

注意:

上面的 **json.conf** 配置只是将 **nginx** 日志输出, 还没有输入到 **elasticsearch** 里, 所以这个时候在 **elasticsearch** 界面里是采集不到 **nginx** 日志的。

需要配置一下，将 **nginx** 日志输入到 **elasticsearch** 中，将其汇总到总文件 **file.conf** 里，如下也将 **nginx-log** 日志输入到 **elasticsearch** 里：（后续就可以只用这个汇总文件，把要追加的日志汇总到这个总文件里即可）

```
1 [root@elk-node1 ~]# cat file.conf
2 input {
3     file {
4         path => "/var/log/messages"
5         type => "system"
6         start_position => "beginning"
7     }
8
9     file {
10        path => "/var/log/elasticsearch/huanqiu.log"
11        type => "es-error"
12        start_position => "beginning"
13        codec => multiline {
14            pattern => "^\[\"
15            negate => true
16            what => "previous"
17        }
18    }
19    file {
20        path => "/var/log/nginx/access_json.log"
21        codec => json
22        start_position => "beginning"
23        type => "nginx-log"
24    }
25 }
26
27
28 output {
29
30     if [type] == "system"{
31         elasticsearch {
32             hosts => ["192.168.1.160:9200"]
33             index => "system-%{+YYYY.MM.dd}"
34         }
35     }
36
37     if [type] == "es-error"{
38         elasticsearch {
39             hosts => ["192.168.1.160:9200"]
40             index => "es-error-%{+YYYY.MM.dd}"
```

```

41         }
42     }
43     if [type] == "nginx-log" {
44         elasticsearch {
45             hosts => ["192.168.1.160:9200"]
46             index => "nignx-log-%{+YYYY.MM.dd}"
47         }
48     }
49 }

```

可以加上**--configtest** 参数，测试下配置文件是否有语法错误或配置不当的地方，这个很重要！！

```
[root@elk-node1 ~]# /opt/logstash/bin/logstash -f file.conf --configtest
```

Configuration OK

然后接着执行 **logstash** 命令（由于上面已经将这个执行命令放到了后台，所以这里其实不用执行，也可以先 **kill** 之前的，再放后台执行），然后可以再访问 **nginx** 界面测试下

```
[root@elk-node1 ~]# /opt/logstash/bin/logstash -f file.conf &
```

登陆 **elasticsearch** 界面查看：

The screenshot shows the Elasticsearch Kibana interface. The browser address bar indicates the URL is `112.110.115.10:19200/_plugin/head/`. The page title is "Elasticsearch" and the address bar shows `http://103.10.86.7:19200/`. The interface includes a navigation bar with tabs for "概览" (Overview), "索引" (Indices), "数据浏览" (Data Explorer), "基本查询" (Basic Search), and "复合查询" (Advanced Search). Below the navigation bar, there are buttons for "集群概览" (Cluster Overview), "集群排序" (Cluster Sort), "Sort Indices", "View Aliases", and "Index Filter". The main content area displays two indices: "system-2016.11.11" and "nignx-log-2016.11.11". The "nignx-log-2016.11.11" index is highlighted with a red box. Below the indices, there are two rows of nodes: "elk-node1" and "elk-node2". Each node has five green status boxes labeled 0 through 4, indicating the health of the nodes.

将 **nginx** 日志整合到 **kibana** 界面里，如下：

Index Patterns

★ es-error*

system*

Configure an index pattern

In order to use Kibana you must configure at least one index pattern. Index patterns are used to configure fields.

☒ Index contains time-based events

☐ Use event times to create index names [DEPRECATED]

Index name or pattern

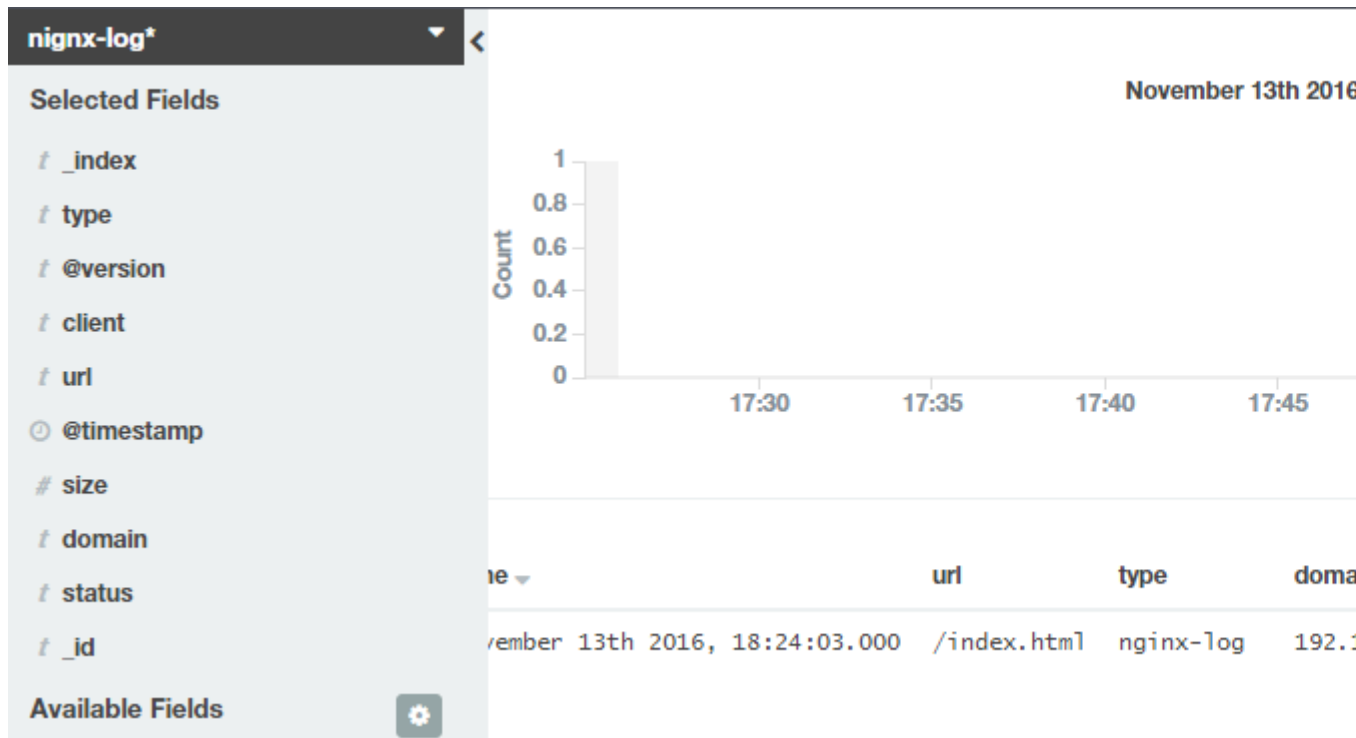
Patterns allow you to define dynamic index names using * as a wildcard.

nignx-log*

Time-field name ⓘ [refresh fields](#)

@timestamp

Create



5) 收集系统日志

编写收集文件并执行。

```
1 [root@elk-node1 ~]# cat syslog.conf
2 input {
3     syslog {
4         type => "system-syslog"
5         host => "192.168.1.160"
6         port => "514"
7     }
8 }
9
10 output {
11     stdout {
12         codec => "rubydebug"
13     }
14 }
```

对上面的采集文件进行执行：

```
[root@elk-node1 ~]# /opt/logstash/bin/logstash -f syslog.conf
```

重新开启一个窗口，查看服务是否启动：

```
[root@elk-node1 ~]# netstat -ntlp|grep 514
```

```
tcp6 0 0 192.168.1.160:514 :::* LISTEN 17842/java
```

```
[root@elk-node1 ~]# vim /etc/rsyslog.conf
```

```
#*. * @remote-host:514
```

【在此

行下面添加如下内容】

```
*. * @@192.168.1.160:514
```



```
[root@elk-node1 ~]# systemctl restart rsyslog
```

回到原来的窗口(即上面采集文件的执行终端)，就会出现数据：

```
1 [root@elk-node1 ~]# /opt/logstash/bin/logstash -f syslog.conf
2 Settings: Default filter workers: 1
3 Logstash startup completed
4 {
5     "message" => "Stopping System Logging Service...\n",
6     "@version" => "1",
7     "@timestamp" => "2016-11-13T10:35:30.000Z",
8     "type" => "system-syslog",
9     "host" => "192.168.1.160",
10    "priority" => 30,
11    "timestamp" => "Nov 13 18:35:30",
12    "logsource" => "elk-node1",
13    "program" => "systemd",
14    "severity" => 6,
15    "facility" => 3,
16    "facility_label" => "system",
17    "severity_label" => "Informational"
18 }
19 .....
20 .....
```

再次添加到总文件 **file.conf** 中：

```
1 [root@elk-node1 ~]# cat file.conf
2 input {
3     file {
4         path => "/var/log/messages"
5         type => "system"
6         start_position => "beginning"
7     }
8
9     file {
10        path => "/var/log/elasticsearch/huanqiu.log"
11        type => "es-error"
12        start_position => "beginning"
13        codec => multiline {
14            pattern => "^\["
15            negate => true
16            what => "previous"
17        }
18    }
19    file {
```

```

20         path => "/var/log/nginx/access_json.log"
21         codec => json
22         start_position => "beginning"
23         type => "nginx-log"
24     }
25     syslog {
26         type => "system-syslog"
27         host => "192.168.1.160"
28         port => "514"
29     }
30 }
31
32
33 output {
34
35     if [type] == "system"{
36         elasticsearch {
37             hosts => ["192.168.1.160:9200"]
38             index => "system-%{+YYYY.MM.dd}"
39         }
40     }
41
42     if [type] == "es-error"{
43         elasticsearch {
44             hosts => ["192.168.1.160:9200"]
45             index => "es-error-%{+YYYY.MM.dd}"
46         }
47     }
48     if [type] == "nginx-log"{
49         elasticsearch {
50             hosts => ["192.168.1.160:9200"]
51             index => "nginx-log-%{+YYYY.MM.dd}"
52         }
53     }
54     if [type] == "system-syslog"{
55         elasticsearch {
56             hosts => ["192.168.1.160:9200"]
57             index => "system-syslog-%{+YYYY.MM.dd}"
58         }
59     }
60 }

```

执行总文件(先测试下总文件配置是否有误，然后先 **kill** 之前在后台启动的 **file.conf** 文件，再次执行):

```
[root@elk-node1 ~]# /opt/logstash/bin/logstash -f file.conf --configtest
```

Configuration OK

```
[root@elk-node1 ~]# /opt/logstash/bin/logstash -f file.conf &
```

测试:

向日志中添加数据, 看 [elasticsearch](#) 和 [kibana](#) 的变化:

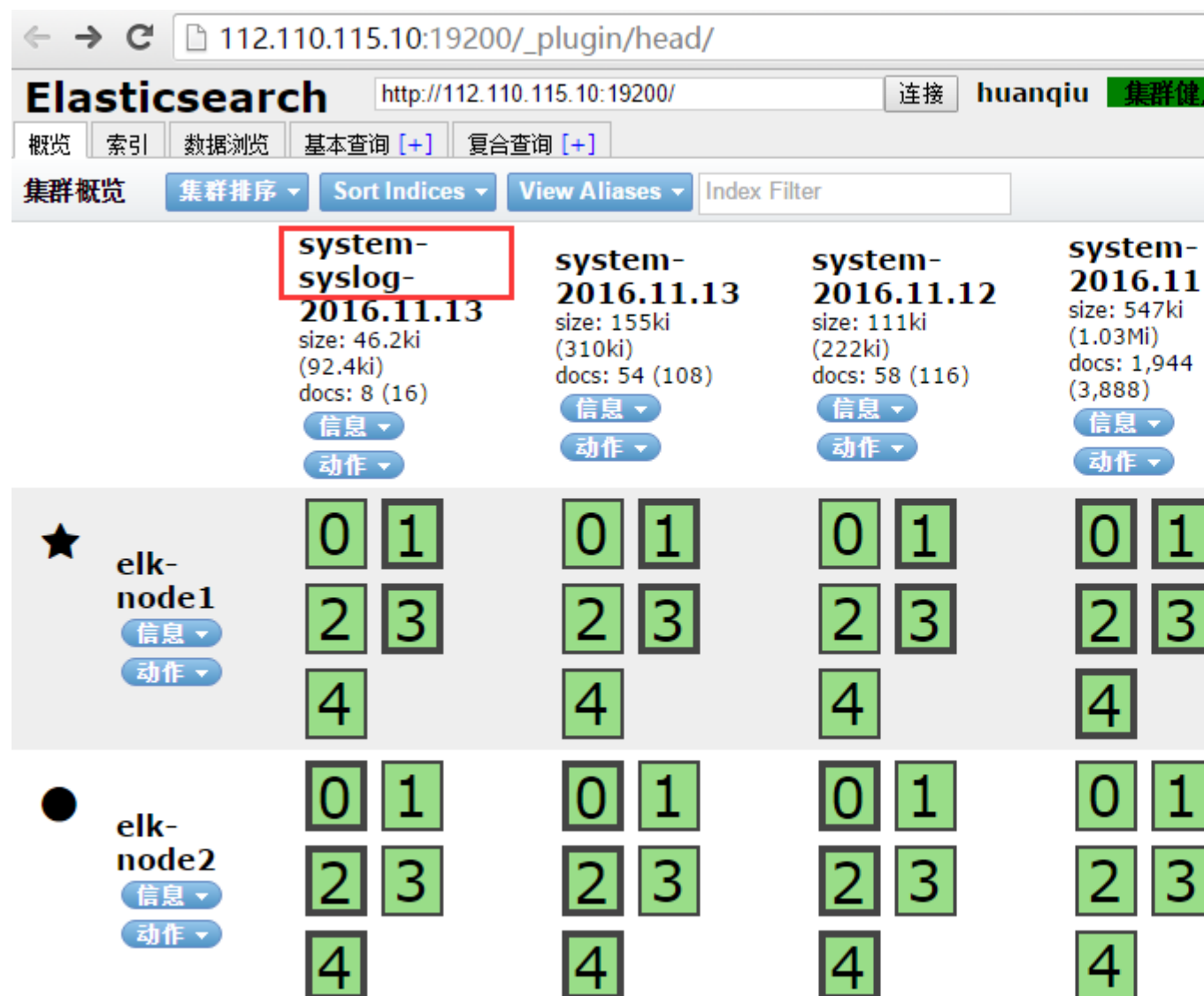
```
[root@elk-node1 ~]# logger "hehehehehehe1"
```

```
[root@elk-node1 ~]# logger "hehehehehehe2"
```

```
[root@elk-node1 ~]# logger "hehehehehehe3"
```

```
[root@elk-node1 ~]# logger "hehehehehehe4"
```

```
[root@elk-node1 ~]# logger "hehehehehehe5"
```



添加到 [kibana](#) 界面中:

Index Patterns

★ es-error*

nginx-log*

system*

Configure an index pattern

In order to use Kibana you must configure at least one index pattern. It is used to configure fields.

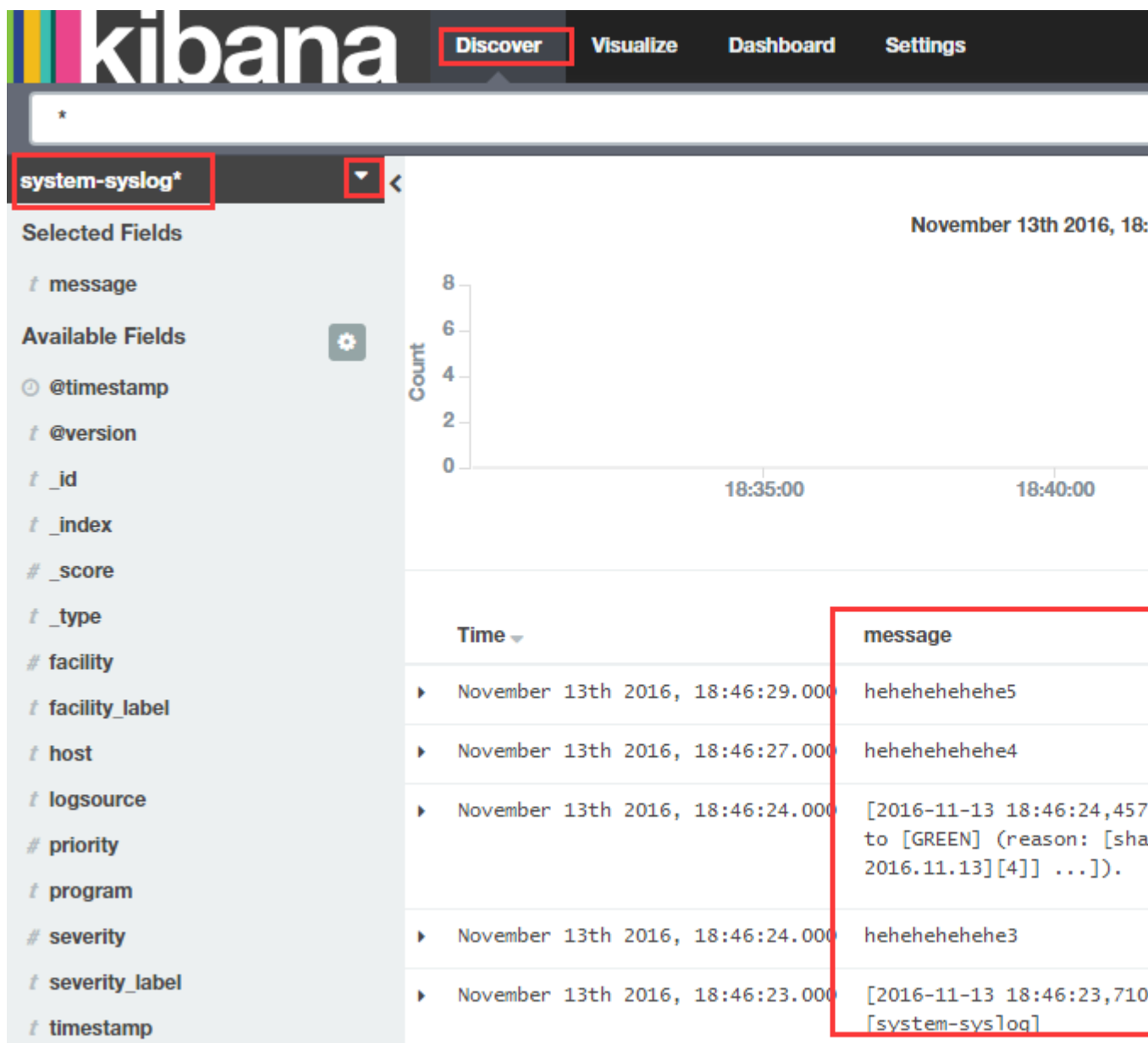
- ☒ Index contains time-based events
- ☐ Use event times to create index names [DEPRECATED]

Index name or pattern

Patterns allow you to define dynamic index names using * as a wildcard.

Time-field name ⓘ [refresh fields](#)

Create



6) TCP 日志的收集

编写日志收集文件，并执行：（有需要的话，可以将下面收集文件的配置汇总到上面的总文件 `file.conf` 里，进而输入到 `elasticsearch` 界面里和 `kibana` 里查看）

```
[root@elk-node1 ~]# cat tcp.conf
```

```
input {
  tcp {
    host => "192.168.1.160"
    port => "6666"
  }
}
output {
  stdout {
    codec => "rubydebug"
```

```
}  
}
```

```
[root@elk-node1 ~]# /opt/logstash/bin/logstash -f tcp.conf
```

开启另外一个窗口，测试一（安装 nc 命令：yum install -y nc）：

```
[root@elk-node1 ~]# nc 192.168.1.160 6666 </etc/resolv.conf
```

回到原来的窗口(即上面采集文件的执行终端)，就会出现数据：

```
[root@elk-node1 ~]# /opt/logstash/bin/logstash -f tcp.conf
```

```
Settings: Default filter workers: 1
```

```
Logstash startup completed
```

```
{  
  "message" => "",  
  "@version" => "1",  
  "@timestamp" => "2016-11-13T11:01:15.280Z",  
  "host" => "192.168.1.160",  
  "port" => 49743  
}
```

测试二：

```
[root@elk-node1 ~]# echo "hehe" | nc 192.168.1.160 6666
```

```
[root@elk-node1 ~]# echo "hehe" > /dev/tcp/192.168.1.160/6666
```

回到之前的执行端口，在去查看，就会显示出来：

1	[root@elk-node1 ~]# /opt/logstash/bin/logstash -f tcp.conf
2	Settings: Default filter workers: 1
3	Logstash startup completed
4	{
5	"message" => "hehe",
6	"@version" => "1",
7	"@timestamp" => "2016-11-13T11:39:58.263Z",
8	"host" => "192.168.1.160",
9	"port" => 53432
10	}
11	{
12	"message" => "hehe",
13	"@version" => "1",
14	"@timestamp" => "2016-11-13T11:40:13.458Z",
15	"host" => "192.168.1.160",
16	"port" => 53457
17	}

7) 使用 filter

编写文件：

1	[root@elk-node1 ~]# cat grok.conf
2	input {
3	stdin{}

4	}
5	filter {
6	grok {
7	match =>
8	{ "message" => "%{IP:client} %{WORD:method} %{URIPATHPARAM:request} %{NUMBER"
9	}
10	}
11	output {
12	stdout{
13	codec => "rubydebug"
14	}
	}

执行检测:

1	[root@elk-node1 ~]# /opt/logstash/bin/logstash -f grok.conf	
2	Settings: Default filter workers: 1	
3	Logstash startup completed	
4	55.3.244.1 GET /index.html 15824 0.043	#转
5	形式	
6	{	
7	"message" => "55.3.244.1 GET /index.html 15824 0.043",	
8	"@version" => "1",	
9	"@timestamp" => "2016-11-13T11:45:47.882Z",	
10	"host" => "elk-node1",	
11	"client" => "55.3.244.1",	
12	"method" => "GET",	
13	"request" => "/index.html",	
14	"bytes" => "15824",	
15	"duration" => "0.043"	
	}	

其实上面使用的那些变量在程序中都有定义:

1	[root@elk-node1 ~]# cd /opt/logstash/vendor/bundle/jruby/1.9/gems/logstash-patt
2	[root@elk-node1 patterns]# ls
3	aws bro firewalls haproxy junos mcollec
4	bacula exim grok-patterns java linux-syslog mcollective-pattern
5	[root@elk-node1 patterns]# cat grok-patterns
6	filter {
7	# drop sleep events
8	grok {
9	match => { "message" =>"SELECT SLEEP" }
10	add_tag => ["sleep_drop"]
11	tag_on_failure => [] # prevent default _grokparsefailure tag
12	}

```

13         if "sleep_drop" in [tags] {
14             drop {}
15         }
16         grok {
17             match => [ "message", "(?m)^# User@Host: %{USER:user}\[[@\]]"
18 Query_time: %{NUMBER:query_time:float}\s+Lock_time: %{NUMBER:lock_time:float}\s
19 timestamp=%{NUMBER:timestamp};\s*(?<query>(?(action>\w+)\s+.*)\n#\s*" ]
20         }
21         date {
22             match => [ "timestamp", "UNIX" ]
23             remove_field => [ "timestamp" ]
24         }
25     }

```

8) mysql 慢查询

收集文件:

```

1 [root@elk-node1 ~]# cat mysql-slow.conf
2 input {
3     file {
4         path => "/root/slow.log"
5         type => "mysql-slowlog"
6         codec => multiline {
7             pattern => "^# User@Host"
8             negate => true
9             what => "previous"
10        }
11    }
12 }
13
14 filter {
15     # drop sleep events
16     grok {
17         match => { "message" => "SELECT SLEEP" }
18         add_tag => [ "sleep_drop" ]
19         tag_on_failure => [] # prevent default _grokparsefailure tag
20     }
21     if "sleep_drop" in [tags] {
22         drop {}
23     }
24     grok {
25         match => [ "message", "(?m)^# User@Host: %{USER:user}\[[@\]]"
26 Query_time: %{NUMBER:query_time:float}\s+Lock_time: %{NUMBER:lock_time:float}\s
27 timestamp=%{NUMBER:timestamp};\s*(?<query>(?(action>\w+)\s+.*)\n#\s*" ]
28     }

```



```

29         date {
30             match => [ "timestamp", "UNIX" ]
31             remove_field => [ "timestamp" ]
32         }
33     }
34
35
36     output {
37         stdout {
38             codec => "rubydebug"
39         }
40     }

```

执行检测:

上面需要的/root/slow.log 是自己上传的, 然后自己插入数据保存后, 会显示:

```

1  [root@elk-node1 ~]# /opt/logstash/bin/logstash -f mysql-slow.conf
2  Settings: Default filter workers: 1
3  Logstash startup completed
4  {
5      "@timestamp" => "2016-11-14T06:53:54.100Z",
6      "message" => "# Time: 161114 11:05:18",
7      "@version" => "1",
8      "path" => "/root/slow.log",
9      "host" => "elk-node1",
10     "type" => "mysql-slowlog",
11     "tags" => [
12         [0] "_grokparsefailure"
13     ]
14 }
15 {
16     "@timestamp" => "2016-11-14T06:53:54.105Z",
17     "message" => "# User@Host: test[test] @ [124.65.197.154]\n# C
18 test_zh_o2o_db;\nSET timestamp=1479092718;\nSELECT trigger_name, event_manipula
19 information_schema.triggers WHERE BINARY event_object_schema='test_zh_o2o_db' A
20     "@version" => "1",
21     "tags" => [
22         [0] "multiline",
23         [1] "_grokparsefailure"
24     ],
25     "path" => "/root/slow.log",
26     "host" => "elk-node1",
27     "type" => "mysql-slowlog"
28 }
29 .....

```


--	-------

=====

=====

接下来描述会遇见到的一个问题：

一旦我们的 **elasticsearch** 出现问题，就不能进行日志采集处理了！

这种情况下该怎么办呢？

解决方案；

可以在 **client** 和 **elasticsearch** 之间添加一个中间件作为缓存，先将采集到的日志内容写到中间件上，然后再从中间件输入到 **elasticsearch** 中。

这就完美的解决了上述的问题了。

(4) ELK 中使用 redis 作为中间件，缓存日志采集内容

1) redis 的配置和启动

[root@elk-node1 ~]# vim /etc/redis.conf #修改下面两行内容

daemonize yes

bind 192.168.1.160

[root@elk-node1 ~]# systemctl start redis

[root@elk-node1 ~]# lsof -i:6379

COMMAND PID USER FD TYPE DEVICE SIZE/OFF NODE NAME

redis-ser 19474 redis 4u IPv4 1344465 0t0 TCP elk-node1:6379 (LISTEN)

[root@elk-node1 ~]# redis-cli -h 192.168.1.160

192.168.1.160:6379> info

Server

redis_version:2.8.19

.....

2) 编写从 Client 端收集数据的文件

1	[root@elk-node1 ~]# vim redis-out.conf
2	input {
3	stdin {}
4	}
5	
6	output {
7	redis {
8	host => "192.168.1.160"
9	port => "6379"
10	db => "6"
11	data_type => "list"
12	key => "demo"
13	}
14	}

3) 执行收集数据的文件，并输入数据 hello redis

[root@elk-node1 ~]# /opt/logstash/bin/logstash -f redis-out.conf

Settings: Default filter workers: 1

Logstash startup completed #下面输入数据 hello redis

hello redis

4) 在 redis 中查看数据

```
[root@elk-node1 ~]# redis-cli -h 192.168.1.160
```

```
192.168.1.160:6379> info
```

```
# Server
```

```
.....
```

```
.....
```

```
# Keyspace
```

```
db6:keys=1,expires=0,avg_ttl=0
```

#在最下面一行，显示是 db6

```
192.168.1.160:6379> select 6
```

```
OK
```

```
192.168.1.160:6379[6]> keys *
```

```
1) "demo"
```

```
192.168.1.160:6379[6]> LINDEX demo -1
```

```
"{\"message\":\"hello redis\",\"@version\":\"1\",\"@timestamp\":\"2016-11-14T08:04:25.981Z\",\"host\":\"elk-node1\"}"
```

5) 继续随便写点数据

1	[root@elk-node1 ~]# /opt/logstash/bin/logstash -f redis-out.conf
2	Settings: Default filter workers: 1
3	Logstash startup completed
4	hello redis
5	123456
6	asdf
7	ert
8	wang
9	shi
10	bo
11	guohuihui
12	as
13	we
14	r
15	g
16	
17	asdfjkdfsak
18	5423wer
19	34rt3
20	6y
21	7uj
22	u
23	io9
24	sdjfh sdk890
25	huanqiu

26	huanqiuchain
27	hqsab
28	asda

6) 在 redis 中查看

在 redis 中查看长度:

```
[root@elk-node1 ~]# redis-cli -h 192.168.1.160
```

```
192.168.1.160:6379> info
```

```
# Server
```

```
redis_version:2.8.19
```

```
.....
```

```
.....
```

```
# Keyspace
```

```
db6:keys=1,expires=0,avg_ttl=0      #显示是 db6
```

```
192.168.1.160:6379> select 6
```

```
OK
```

```
192.168.1.160:6379[6]> keys *
```

```
1) "demo"
```

```
192.168.1.160:6379[6]> LLEN demo
```

```
(integer) 24
```

7) 将 redis 中的内容写到 ES 中

1	[root@elk-node1 ~]# vim redis-in.conf
2	input {
3	redis {
4	host => "192.168.1.160"
5	port => "6379"
6	db => "6"
7	data_type => "list"
8	key => "demo"
9	}
10	}
11	
12	output {
13	elasticsearch {
14	hosts => ["192.168.1.160:9200"]
15	index => "redis-in-%{+YYYY.MM.dd}"
16	}
17	}

执行:

```
[root@elk-node1 ~]# /opt/logstash/bin/logstash -f redis-in.conf --configtest
```

```
Configuration OK
```

```
[root@elk-node1 ~]# /opt/logstash/bin/logstash -f redis-in.conf &
```

在 [redis](#) 中查看，发现数据已被读出：

```
192.168.1.160:6379[6]> LLEN demo
```

```
(integer) 0
```

=====温馨提示=====

- 1 redis 默认只有 16 个数据库，也就是说最多只能有 16 个 db，即 db01-db15
- 2 但是 key 值可以设置不同，也就是针对不同日志的 key 前缀可以设置不同.
- 3 比如：
- 4 key => "nginx.log"的值最多可以设置 16 个 db，即 db01-db15
- 5 key => "mysql.log"的值最多可以设置 16 个 db，即 db01-db15
- 6 key => "tomcat.log"的值最多可以设置 16 个 db，即 db01-db15

登陆 [elasticsearch](#) 界面查看：

← → ↻ ⓘ 112.110.115.10:19200/_plugin/head/

应用 网址导航 百度 爱淘宝 天猫 游戏大全 网址大全 hao 百度Hao123网址导航 360

Elasticsearch

http://112.110.115.10:19200/ 连接 huanqiu 集群健康

概览 索引 数据浏览 基本查询 [+] 复合查询 [+]

集群概览 集群排序 ▾ Sort Indices ▾ View Aliases ▾ Index Filter

	system-syslog-2016.11.14	system-syslog-2016.11.13	system-2016.11.14	system-2016.11.13
size:	169ki (331ki)	268ki (529ki)	164ki (323ki)	138ki (277ki)
docs:	100 (200)	139 (278)	49 (98)	98 (19)
信息 ▾	信息 ▾	信息 ▾	信息 ▾	信息 ▾
动作 ▾	动作 ▾	动作 ▾	动作 ▾	动作 ▾

★ elk-node1

信息 ▾ 动作 ▾

● elk-node2

信息 ▾ 动作 ▾



```
1 [root@elk-node1 ~]# vim shipper.conf
2 input {
3     file {
4         path => "/var/log/messages"
5         type => "system"
6         start_position => "beginning"
7     }
8 }
```

```

8
9         file {
10             path => "/var/log/elasticsearch/huanqiu.log"
11             type => "es-error"
12             start_position => "beginning"
13             codec => multiline {
14                 pattern => "^\[\"
15                 negate => true
16                 what => "previous"
17             }
18         }
19         file {
20             path => "/var/log/nginx/access_json.log"
21             codec => json
22             start_position => "beginning"
23             type => "nginx-log"
24         }
25         syslog {
26             type => "system-syslog"
27             host => "192.168.1.160"
28             port => "514"
29         }
30
31     }
32
33
34     output {
35         if [type] == "system"{
36             redis {
37                 host => "192.168.1.160"
38                 port => "6379"
39                 db => "6"
40                 data_type => "list"
41                 key => "system"
42             }
43         }
44
45         if [type] == "es-error"{
46             redis {
47                 host => "192.168.1.160"
48                 port => "6379"
49                 db => "6"
50                 data_type => "list"

```

```

51             key => "demo"
52         }
53     }
54     if [type] == "nginx-log"{
55         redis {
56             host => "192.168.1.160"
57             port => "6379"
58             db => "6"
59             data_type => "list"
60             key => "nginx-log"
61         }
62     }
63     if [type] == "system-syslog"{
64         redis {
65             host => "192.168.1.160"
66             port => "6379"
67             db => "6"
68             data_type => "list"
69             key => "system-syslog"
70         }
71     }
72 }

```

执行上面的文件（提前将上面之前启动的 **file.conf** 文件的执行给结束掉！）

```
[root@elk-node1 ~]# /opt/logstash/bin/logstash -f shipper.conf --configtest
```

Configuration OK

```
[root@elk-node1 ~]# /opt/logstash/bin/logstash -f shipper.conf
```

Settings: Default filter workers: 1

Logstash startup completed

在 **redis** 中查看:

```
[root@elk-node1 ~]# redis-cli -h 192.168.1.160
```

```
192.168.1.160:6379> info
```

```
# Server
```

```
redis_version:2.8.19
```

```
.....
```

```
.....
```

```
# Keyspace
```

```
db6:keys=1,expires=0,avg_ttl=0
```

#显示是 db6

```
192.168.1.160:6379> select 6
```

```
OK
```

```
192.168.1.160:6379[6]> keys *
```

```
1) "demo"
```

```
2) "system"
```

```
192.168.1.160:6379[6]> keys *
```


- 1) "nginx-log"
- 2) "demo"
- 3) "system"

另开一个窗口，添加点日志：

```
[root@elk-node1 ~]# logger "12325423"
[root@elk-node1 ~]# logger "12325423"
[root@elk-node1 ~]# logger "12325423"
[root@elk-node1 ~]# logger "12325423"
[root@elk-node1 ~]# logger "12325423"
[root@elk-node1 ~]# logger "12325423"
```

又会增加日志：

192.168.1.160:6379[6]> keys *

- 1) "system-syslog"
- 2) "nginx-log"
- 3) "demo"
- 4) "system"

其实可以在任意的一台 ES 中将数据从 redis 读取到 ES 中。

下面咱们在 **elk-node2** 节点，将数据从 **redis** 读取到 **ES** 中：

编写文件：

```
1 [root@elk-node2 ~]# cat file.conf
2 input {
3     redis {
4         type => "system"
5         host => "192.168.1.160"
6         port => "6379"
7         db => "6"
8         data_type => "list"
9         key => "system"
10    }
11
12    redis {
13        type => "es-error"
14        host => "192.168.1.160"
15        port => "6379"
16        db => "6"
17        data_type => "list"
18        key => "es-error"
19    }
20    redis {
21        type => "nginx-log"
22        host => "192.168.1.160"
23        port => "6379"
24        db => "6"
```

```

25         data_type => "list"
26         key => "nginx-log"
27     }
28     redis {
29         type => "system-syslog"
30         host => "192.168.1.160"
31         port => "6379"
32         db => "6"
33         data_type => "list"
34         key => "system-syslog"
35     }
36
37 }
38
39
40 output {
41
42     if [type] == "system"{
43         elasticsearch {
44             hosts => ["192.168.1.160:9200"]
45             index => "system-%{+YYYY.MM.dd}"
46         }
47     }
48
49     if [type] == "es-error"{
50         elasticsearch {
51             hosts => ["192.168.1.160:9200"]
52             index => "es-error-%{+YYYY.MM.dd}"
53         }
54     }
55     if [type] == "nginx-log"{
56         elasticsearch {
57             hosts => ["192.168.1.160:9200"]
58             index => "nignx-log-%{+YYYY.MM.dd}"
59         }
60     }
61     if [type] == "system-syslog"{
62         elasticsearch {
63             hosts => ["192.168.1.160:9200"]
64             index => "system-syslog-%{+YYYY.MM.dd}"
65         }
66     }
67 }

```

执行:

```
[root@elk-node2 ~]# /opt/logstash/bin/logstash -f file.conf --configtest
```

Configuration OK

```
[root@elk-node2 ~]# /opt/logstash/bin/logstash -f file.conf &
```

去 redis 中检查, 发现数据已经被读出到 elasticsearch 中了。

```
192.168.1.160:6379[6]> keys *
```

(empty list or set)

同时登陆 logstash 和 kibana 看, 发现可以正常收集到日志了。

可以执行这个 去查看 nginx 日志

```
[root@elk-node1 ~]# ab -n10000 -c1 http://192.168.1.160/
```

也可以启动多个 redis 写到 ES 中, 具体根据自己的实际情况而定。

=====logstash 配置 java 环境=====

由于新版的 ELK 环境要求 java1.8, 但是有些服务器由于业务代码自身限制只能用 java6 或 java7。

这种情况下, 要安装 Logstash, 就只能单独配置 Logstash 自己使用的 java 环境了。

1	操作如下:
2	0) 使用 rpm 包安装 logstash
3	
4	1) 安装 java8, 参考: http://www.cnblogs.com/kevingrace/p/7607442.html
5	
6	2) 在/etc/sysconfig/logstash 文件结尾添加下面两行内容:
7	[root@cx-app01 ~]# vim /etc/sysconfig/logstash
8
9	JAVA_CMD=/usr/local/jdk1.8.0_172/bin
10	JAVA_HOME=/usr/local/jdk1.8.0_172
11	
12	3) 在/opt/logstash/bin/logstash.lib.sh 文件添加下面一行内容:
13	[root@cx-app02 ~]# vim /opt/logstash/bin/logstash.lib.sh
14
15	export JAVA_HOME=/usr/local/jdk1.8.0_172
16	
17	4) 然后使用 logstash 收集日志, 就不会报 java 环境错误了。

=====配置范例=====

1	如下的配置范例:
2	192.168.10.44 为 elk 的 master 节点, 同时也是 redis 节点
3	
4	[root@client-node01 opt]# pwd
5	/opt
6	[root@client-node01 opt]# cat redis-in.conf
7	input {
8	file {
9	path => "/usr/local/tomcat8/logs/catalina.out"
10	type => "tomcat8-logs"

```

11         start_position => "beginning"
12         codec => multiline {
13             pattern => "^\[\"           //表示收集以\"[\"开
14             negate => true
15             what => "previous"
16         }
17     }
18 }
19
20 output {
21     if [type] == "tomcat8-logs"{
22         redis {
23             host => "192.168.10.44"
24             port => "6379"
25             db => "1"
26             data_type => "list"
27             key => "tomcat8-logs"
28         }
29     }
30 }
31
32 [root@client-node01 opt]# cat redis-input.conf
33 input {
34     file {
35         path => "/var/log/messages"
36         type => "systemlog"
37         start_position => "beginning"
38         stat_interval => "2"
39     }
40 }
41
42 output {
43     if [type] == "systemlog" {
44         redis {
45             data_type => "list"
46             host => "192.168.10.44"
47             db => "2"
48             port => "6379"
49             key => "systemlog"
50         }
51     }
52 }
53 }

```

```
54
55 [root@client-node01 opt]# cat file.conf
56 input {
57     redis {
58         type => "tomcat8-logs"
59         host => "192.168.10.44"
60         port => "6379"
61         db => "1"
62         data_type => "list"
63         key => "tomcat8-logs"
64     }
65
66     redis {
67         type => "systemlog"
68         host => "192.168.10.44"
69         port => "6379"
70         db => "2"
71         data_type => "list"
72         key => "systemlog"
73     }
74 }
75
76
77
78 output {
79
80     if [type] == "tomcat8-logs"{
81         elasticsearch {
82             hosts => ["192.168.10.44:9200"]
83             index => "elk-node2-tomcat8-logs-%{+YYYY.MM.dd}"
84         }
85     }
86
87     if [type] == "systemlog"{
88         elasticsearch {
89             hosts => ["192.168.10.44:9200"]
90             index => "elk-node2-systemlog-%{+YYYY.MM.dd}"
91         }
92     }
93 }
94
95
96 [root@client-node01 opt]# /opt/logstash/bin/logstash -f /opt/redis-in.conf --c
```

```

97 Configuration OK
98 [root@client-node01 opt]# /opt/logstash/bin/logstash -f /opt/redis-input.conf
99 Configuration OK
100 [root@client-node01 opt]# /opt/logstash/bin/logstash -f /opt/file.conf --confi
101 Configuration OK
102
103 启动 logstash
104 [root@client-node01 opt]# /opt/logstash/bin/logstash -f /opt/redis-in.conf &
105 [root@client-node01 opt]# /opt/logstash/bin/logstash -f /opt/redis-input.conf
106 [root@client-node01 opt]# /opt/logstash/bin/logstash -f /opt/file.conf &
107
108 这时候，当/usr/local/tomcat8/logs/catalina.out 和/var/log/messages 文件里有新日
109 在 redis 里就能查看到相关信息，并查看写入到 es 里。
110
111 =====
112 温馨提示：
113 当客户机的日志信息收集后，经过 redis 刚读到 es 数据库里后，如果没有新数据写入，则
114 数据的，只有当日志文件里有新的日志写入后才会触发数据展示的动作，即 es 的访问界面
115 (http://192.168.10.44:9200/_plugin/head/)
116 里才能看到日志数据的展示效果。
117 =====
118
119 假设想上面两个文件里写入测试数据
120 [root@client-node01 opt]# echo "hellohellohellohello" >> /var/log/messages
121 [root@client-node01 opt]# echo "[hahahahahahhahahahahahahahahahahah]" >> /usr/
122
123 到 redis 里发现有相关的 key，很快就会读到 es 里。可以配置到 kibana 里观察。
124
125 可以先测试下日志信息是否写到 redis 里？然后再测试下数据是否从 redis 读到 es 里？-
126
127 注意上面 redis-in.conf 文件中的下面设置，使用正则匹配，收集以哪些字符开头的日志
128 pattern => "^\[\" 表示收集以\"[\"开头的日
129 pattern => "^2018\" 表示收集以\"2018\"开头的日
130 pattern => \"^[a-zA-Z0-9]\" 表示收集以字母（大小写）或数字
pattern => \"^[a-zA-Z0-9]|[^ ]+\" 表示收集以字母（大小写）或数字或空格

```

*****当你发现自己的才华撑不起野心时，就请安静下来学习吧*****

<https://www.cnblogs.com/kevingrace/p/5919021.html>