取最匹配的结果

Date类型用于sort时，如何显示

Joining queries

Span queries 实际使用的意义。

?size=0

Qt udpsocket pendingDatagramSize() 超过udp报文最大长度怎么办？

Typedef 和宏区别

1. Logstash
   1. 参数

–e 执行

-f 或 --config 读取配置文件

-t 配置文件语法能否正常解析

logstash -f my.conf

加上-t测试结果是否正确，但是会屏蔽正常的输出内容。

-r 支持重新读取配置文件，自动发现培训文件修改了。

logstash -f config/dclog2stdout.conf -r

* 1. 语法

%{变量名} 引用该变量的值

[geoip][location][0] 获取字段内容，location是字段，0是其第一个，支持下标，-1代表最后一个元素。

%{PATTERN\_NAME:capture\_name:data\_type}，capture\_name就是

* 1. Output
     1. 输出到Elasticsearch

注意索引名中不能有大写字母， 否则 ES 在日志中会报InvalidIndexNameException

映射模板里面，索引名称的匹配符是logstash-\*，logstash的配置文件中conf的es部分索引名称需要符合规则。

如果发现这个索引同时匹配上了多个 template ， 那么就会先应用 order 数值小的 template 设置，然后再应用一遍 order 数值高的作为覆盖， 最终达到一个 merge 的效果。所以需要覆盖的模板内容在自定义里面写，其余的用默认模板。

1. Elasticsearch
   1. 搜索
      1. 多个索引中检索

，支持名称以\*作为通配符。

/g\*,u\*/\_search 后面再接查询体

* + 1. 结果分页

从第10个文档开始，显示5条文档，内部其实前10个文档也要检索并排序，所以检索靠后的文档该方式效率低。

GET /\_search?size=5&from=10

* + 1. 只检索部分字段

GET /itest/ttest/00011?\_source=age,school

只得到\_source字段而不要其他的元数据

* + 1. DSL包括过滤和查询两种情况

过滤：不评价相关性，只回答yes还是no。

Filter context is in effect whenever a query clause is passed to a filter parameter, such as the filter or must\_not parameters in the bool query, the filter parameter in the constant\_score query, or the filter aggregation

查询：用于全文搜索场景

* + 1. 单查询

GET 索引/\_search

{

"query": 后面是具体的查询体

}

查询体结构

{

QUERY\_NAME: {

ARGUMENT: VALUE,

ARGUMENT: VALUE,...

}

}

QUERY\_NAME 包括如下

* + 1. Match 例子{ "match": { "tweet": "About Search" }}，如果字段tweet是text类型，则全文检索（如果想精确匹配使用match\_phrase）；如果字段是keyword类型，则精确匹配。

"match" : {

"message" : {

"query" : "this is a test",

"operator" : "and" #全文检索时，分割的多个语句的bool关系。

}

}

Ref: The operator flag can be set to or or and to control the boolean clauses

* + 1. multi\_match 在多个字段查询

"multi\_match": {

"query": "full text search",

"fields": [ "title", "body" ] #支持通配符\*\_name

}

* + 1. range 查询找出那些落在指定区间内的数字或者时间

"range": {

"logDate": {

"gt": "2018-03-21||-1M",

"lte": "2018-03-21"

}

}

字段可以为range类型，从而进行精确匹配。

日期的格式为yyyy-MM-dd HH:mm:ss

* + 1. range的日期参数支持计算

Assuming now is 2001-01-01 12:00:00, some examples are:

now+1h now in milliseconds plus one hour. Resolves to: 2001-01-01 13:00:00

now-1h now in milliseconds minus one hour. Resolves to: 2001-01-01 11:00:00

now-1h/d now in milliseconds minus one hour, rounded down to UTC 00:00. Resolves to: 2001-01-01 00:00:00`

2001.02.01\|\|+1M/d 2001-02-01 in milliseconds plus one month. Resolves to: 2001-03-01 00:00:00

* + 1. Term 是精确匹配，要求该字段为keyword类型，数字和日期字段天生支持精确匹配。

term不对查询参数进行分词，直接从倒排索引去搜索，所以需要字段为keyword类型。Text类型字段存储到倒排索引中时是分词之后的结果，完整结果不在分词中（解决方案是对该字段存储的值使用fields修改，支持text和field两个方式去解析。）

Terms，

"terms" : { "user" : ["kimchy", "elasticsearch"]}

* + 1. Exist
  1. 组合查询
     1. Bool

对查询进行组合，should 表示有则score更高（没有must时，必须要满足一个should）。Must和should参与score的打分，Filter不参与打分，所以进行过滤的must，range等放到filter字段。

* + 1. Named Queries

查看每个结果中，是哪一个bool字句生效

{"match" : { "name.first" : {"query" : "shay", "\_name" : "first"} }}

* 1. 排序
  2. 聚合
     1. 两个主要的概念：

桶（Buckets）满足特定条件的文档的集合

指标（Metrics）对桶内的文档进行统计计算

* + 1. 模板

"aggregations" : {

"<aggregation\_name>" : {

"<aggregation\_type>" : {

<aggregation\_body>

}

[,"meta" : { [<meta\_data\_body>] } ]?

[,"aggregations" : { [<sub\_aggregation>]+ } ]?

}

[,"<aggregation\_name\_2>" : { ... } ]\*

}

* + 1. Terms

一个索引内所有文档分散在多个主分片中（A，B，C），统计某字段值的数量，每个分片只统计前Size个值，从而合并A B C分片结果时，若该值在B分片的排序在第Size名后面，则合并时只能统计A C分片结果，少统计了结果。

思考：一个索引就一个分片。

"show\_term\_doc\_count\_error": true 结果中显示每个桶元素结果被 少统计的数量.

* + 1. Composite

Sources字段

* + 1. 值得关注的

Terms Aggregation

Composite Aggregation

Filter Aggregation

Range Aggregation

Date Range Aggregation

Es2.5 里面的《过滤和聚合》

* 1. 常见命令
     1. 改变\_score值

The \_score can be changed with the boost parameter。

{ "boost" : 1.2 }

* + 1. 解释查询体含义

GET \_validate/query?explain

{具体查询体}

* + 1. 查询分词器如何对查询参数进行分词

GET /kibana\_sample\_data\_ecommerce/\_analyze //不要加上文档名称

{

"field": "customer\_full\_name", //字段名称

"text": "Mostafa Long" //查询内容

}

* + 1. 只显示想要的字段

和"query"同级别加上 "\_source":["customer\_full\_name","currency"]

* + 1. Index.html显示完整目录

在html中，删除style.css

* + 1. 集群健康状况

GET /\_cluster/health

* + 1. 文档数量

GET /indexName/docName/\_count

加上\_count就是数量统计

* + 1. 索引情况

curl -XGET http://192.168.245.3:9200/\_cat/indices?v

* + 1. 高亮

"highlight": {

"fields": {

"school": {}

}

}

* 1. 元数据

\_all 字段在6.0版本废弃了。

\_type字段也没有用了。

* 1. 映射与分析
     1. Template

遇到索引名称匹配的，对于字段的映射使用该模板，顺序从小到大，小的优先级高。

PUT /\_template/logstash\_my

{

"order": 1,

"index\_patterns": "logstash-\*", //匹配以logstash-开头的索引，自动使用该映射模板

"settings": {

"index.refresh\_interval": "10s"

},

"mappings": {

"doc": {

"dynamic\_templates": [

{

"string\_fields": {

"match\_mapping\_type": "string",

"mapping": {

"type": "keyword"

}

}

},

{

"message\_field": {

"path\_match": "message",

"match\_mapping\_type": "string",

"mapping": {

"type": "keyword"

}

}

}

]

}

}

}

* + 1. 查看索引映射

GET /logstash-dc\_log/\_mapping/

* + 1. 分析器

把单词复数形式，大小写，同义词 映射到 同一个单词。查询时，也根据该字段进行分词。

* + 1. 查询类型各字段的映射

即理解为整形还是字符串还是日期还是bool

GET /itest/\_mapping/ttest

结果分析：

"type" : "text", 进行全文搜索

"type" : "keyword", 字段不拆分，只能全部匹配。主要用于sorting and aggregations。

如下，一个字段可以使用多种分词方式，利用fields字段，category字段是全文匹配。Category.keyword是category的别名，搜索时使用Category.keyword=值，表示精确匹配。

"category" : {

"type" : "text",

"fields" : {

"keyword" : {

"type" : "keyword",

"ignore\_above" : 256 #超过256个字符就不精确匹配了

}

}

}

"type" : "date",json没有日期类型，es展现使用字符串，格式和输入时一致，存储使用long，从1970年开始。创建映射时，日期字段可以指定多个日期展示格式，

"date": {

"type": "date",

"format": "yyyy-MM-dd HH:mm:ss||yyyy-MM-dd||epoch\_millis"

}

Put日期时，默认字符串格式为2018-12-21T01:10:10Z，其中Z表示时区zone。

* + 1. 修改映射

已有字段的映射不能修改，对已有字段多种解析可以增加，新增的字段映射可以修改。如下，新增时间字段itime

PUT itemsnoid/\_mapping/item

{

"properties":{

"itime":{"type":"date"}

}

}

* + 1. 其他

每个类型拥有自己的映射(mapping)或者模式定义(schema definition)。一个映射定义了字段类型,每个字段的数据类型,以及字段被Elasticsearch处理的方式。映射还用于设置关联到类型上的元数据。

* 1. 使外部地址访问

配置文件修改为 network.host: 0.0.0.0

* 1. 笔记
     1. 客户端

节点客户端(node client):

节点客户端以无数据节点(none data node)身份加入集群,换言之,它自己不存储任何数据,但是它知道数据在集群中的具体位置,并且能够直接转发请求到对应的节点上。

传输客户端(Transport client):

这个更轻量的传输客户端能够发送请求到远程集群。它自己不加入集群,只是简单转发请求给集群中的节点。

* + 1. 分布式原理

一个节点(node)就是一个Elasticsearch实例,而一个集群(cluster)由一个或多个节点组成,它们具有相同的 cluster.name ,它们协同工作,分享数据和负载。

索引只是一个用来指向一个或多个分片(shards)的“逻辑命名空间(logical namespace)”.

一个分片(shard)是一个最小级别“工作单元(worker unit)”,它只是保存了索引中所有数据的一部分。一个文档使用一个分片存储，不会分为两个分片存储。

当索引创建完成的时候,主分片（primary shard）的数量就固定了,但是复制分片（replica shard）的数量可以随时调整。

* + 1. 数据

在Elasticsearch中,文档(document)这个术语有着特殊含义。它特指最顶层结构或者根对象(root object)序列化成的JSON数据(以唯一ID标识并存储于Elasticsearch中)。

一个文档不只有数据。它还包含了元数据(metadata)——关于文档的信息。三个必须的元数据节点是: 索引(index) 文档存储的地方、\_type 文档代表的对象的类、文档的唯一标识\_id，Elasticsearch中每个文档都有版本号,每当文档变化(包括删除)都会使 \_version 增加。元数据貌似都是以下划线开头。

\_source表示文档的原始数据，与输入的数据保持一致。

增加数据：

PUT /{index}/{type}/{id}

{

"field": "value",

...

}

自动生成id

POST /itest/ttest/

{

"name": "Leo Tong",

"school": "The 28th ",

"age": 51,

"time":null

}

更新数据

文档是不可变的——它们不能被更改,只能被替换，

多线程的安全性。同时修改一个数据，使用version关键字。在version为1时才修改成功，如果其他线程修改了该数据，version不为1，则需要重新读取version号，并进行修改。

PUT /website/blog/1?version=1

{

}

update命令内部用新的文档代替旧文档，利用上述原理保证更新操作是原子的。

批量操作

\_mget docs

有一个最佳的 bulk 请求大小，开始的数量可以在1000~5000个文档之间，一个好的批次最好保持在5-15MB大小间

* + 1. 分布式文档存储

文档映射到哪一个shard。routing 值是一个任意字符串,它默认是\_id但也可以自定义。

shard = hash(routing) % number\_of\_primary\_shards。

这也解释了为什么主分片的数量只能在创建索引时定义且不能修改:如果主分片的数量在未来改变了,所有先前的路由值就失效了,文档也就永远找不到了

1. Kibana
   1. 连接其他机器的es

修改配置文件的 Elasticsearch URL

* 1. Dev tools

Ctrl +/ 转到该命令的帮助，可以去index找对应的

Ctrl+I 自动缩进

Ctrl+space 自动完成

Ctrl+alt+L 折叠