**ElasticSearch简单交流**

1. ElasticSearch介绍

Elasticsearch 是一个开源的搜索引擎，建立在一个全文搜索引擎库 [Apache Lucene™](https://lucene.apache.org/core/" \t "https://www.elastic.co/guide/cn/elasticsearch/guide/current/_top) 基础之上。 Lucene 可以说是当下最先进、高性能、全功能的搜索引擎库--无论是开源还是私有。

但是 Lucene 仅仅只是一个库。为了充分发挥其功能，你需要使用 Java 并将 Lucene 直接集成到应用程序中。 更糟糕的是，您可能需要获得信息检索学位才能了解其工作原理。Lucene 非常 复杂。

Elasticsearch 也是使用 Java 编写的，它的内部使用 Lucene 做索引与搜索，但是它的目的是使全文检索变得简单， 通过隐藏 Lucene 的复杂性，取而代之的提供一套简单一致的 RESTful API。

然而，Elasticsearch 不仅仅是 Lucene，并且也不仅仅只是一个全文搜索引擎。 它可以被下面这样准确的形容：

一个分布式的实时文档存储，每个字段 可以被索引与搜索；

一个分布式实时分析搜索引擎；

能胜任上百个服务节点的扩展，并支持 PB 级别的结构化或者非结构化数据；

Elasticsearch 将所有的功能打包成一个单独的服务，这样你可以通过程序与它提供的简单的 RESTful API 进行通信， 可以使用自己喜欢的编程语言充当 web 客户端，甚至可以使用命令行（去充当这个客户端）。

Elasticsearch是一个基于Lucene的实时的分布式搜索和分析引擎。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。基于RESTful接口。

1)普通请求是...get?a=1

2)rest请求....get/a/1

Elasticsearch和MySQL对比情况：

|  |  |
| --- | --- |
| **MySQL** | **elasticsearch** |
| database(数据库) | index(索引库) |
| table(表) | type(类型) |
| row(行) | document(文档) |
| column(列) | field(字段) |

Elasticsearch的名词简单介绍：

(1)Index（索引）

     索引就是像关系数据库中的“数据库”。通过映射可以定义成多种类型。索引是一个逻辑命名空间映射到一个或多个主要的分片，可以有零个或多个副本分片。

(2)Type(类型)

     Type是相当于关系数据库中的“表”。每种类型都有一列字段，用来定义文档的类型。映射定义了对在文档中的每个字段如何进行分析。

(3) Id（标识）

     每个文档ID标识了一个文档。一个文档的索引/类型/ ID必须是唯一的。如果没有提供ID，将是自动生成。（还可以看到路由<routing>）。

(4)Cluster （集群）

     一个集群包含一个或多个分配了相同的集群名称的节点。每个集群都有一个主节点是集群自动选择产生,并且可以决定如果当前主节点失败，哪些可以替换。

(5)Node（节点）

     节点是属于elasticsearch群集的运行实例。测试的时候，在一台服务器可以启动多个节点，但通常情况下应该在一台服务器运行一个节点。在启动时，节点将使用单播（或组播，但是必须指定）来发现使用相同的群集名称的群集，并会尝试加入该群集。

(6)Shard（分片）

      一个分片是一个单一的Lucene的实例。这是一个低级别的通过ElasticSearch自动管理的“工作者”单元。索引是一个逻辑命名空间指向主分片和副本分片。索引的主分片和副本分片的数量需要明确的指定。然而你的代码应该只处理一个索引。Elasticsearch分配集群中所有节点的分片。在节点出现故障或增加新节点的时候，可以自动的将一个节点上的分片移动到另一个节点上。

(7)Primary shard（主分片）

     每个文档都存储在一个主要分片上。当你索引一个文档时，索引首先生成在主分片上，然后才到主分片的所有副本上。默认情况下，索引有5个主分片。您可以指定更多或更少的主分片来适应索引可以处理的文档数。一旦创建了索引，就不能改变索引中主分片的数量。

(8)Replica shard（副本分片）

      每个主分片可以有零个或多个副本。副本是主分片的一个拷贝，有两个作用：

         1、故障转移：如果主分片有问题，副本分片可以提升为主分片；

         2、提高性能：获取和搜索请求可以处理主分片或副本分片。

      默认情况下，每个主分片有一个副本，不过索引的副本数量可以动态地改变。在同一个节点上，一个副本分片将永远不会和其主分片一起运行。

(9)Text(文本)

     文本（或全文）是普通非结构化的文本，如本段。默认情况下，文本将被分析成术语，术语才是实际存储在索引中。文本字段在索引时需要进行分析，以便全文搜索，全文查询的关键字在搜索时，必须分析产生（搜索）与索引时相同的术语。

(10)Field（字段）

     文档中包含的一组字段或键值对。字段的值可以是一个简单的（标量）值（如字符串，整数，日期），或者一个嵌套的结构就像一个数组或对象。一个字段就是类似关系数据库表中的一列。映射的每个字段有一个字段的类型“type”（不要与文档类型混淆），表示那种类型的数据可以存储在该字段里，如：整数<integer>，字符串<string>，对象<object>。映射还允许你定义（除其他事项外）一个字段的值如何进行分析。

(11)Source field（源字段）

     默认情况下，你的JSON文档将被索引存储在\_source字段里面，所有的get（获取）和search（搜索）请求将返回的该字段。这将允许你直接从搜索结果中访问到源数据，而不需要再次发起请求检索。

     注：索引将返回完整的的JSON字符串给你，即使它包含无效的JSON。此字段里的内容不表示任何该对象里面的数据如何被索引。

Mapping（映射）

     映射是像关系数据库中的”模式定义“。每个索引都有一个映射，它定义了每个索引的类型，再加上一些索引范围的设置。映射可以被明确地定义，或者在一个文档被索引的时候自动生成。

总结下名词之间的行为可以概括为：

     Cluster包含多个node，Indices不应该理解成动词索引，Indices可理解成关系数据库中的databases，Indices可包含多个Index，Index对应关系数据库中的database，它是用来存储相关文档的。

Elasticsearch与关系数据的类比对应关系如下：

IMG_256

    document的可以理解为一个JSON序列对象。每个document可包含多个field。

    每个Index（对应Database）包含多个Shard，默认是5个，分散在不同的Node上，但不会存在两个相同的Shard存在一个Node上，这样就没有备份的意义了。Shard是一个最小的Lucene索引单元。

     当来一个document的时候，Elasticsearch通过对docid进行hash来确定其放在哪个shard上面，然后在shard上面进行索引存储。

Replicas就是备份，Elasticsearch采用的是PushReplication模式，当你往 master主分片上面索引一个文档，该分片会复制该文档(document)到剩下的所有 replica副本分片中，这些分片也会索引这个文档。

     当进行查询是，如果提供了查询的DocID，Elasticsearch通过hash就知道Doc存在哪个shard上面，再通过routingtable查询就知道再哪个node上面，让后去node上面去取就好了。如果不提供DocID,那么Elasticsearch会在该Index（indics）shards所在的所有node上执行搜索预警，然后返回搜索结果，由coordinating node gather之后返回给用户。

1. ElasticSearch安装

2.1 Elasticsearch安装(1.4.4版本)

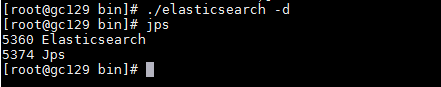
1)java版本要求：最低1.7

2)下载地址：<https://www.elastic.co/downloads/past-releases/1-4-4>

(下载ZIP包后使用unzip命令解压缩到/usr/local/gc/目录下 )

3)启动cd /usr/local/gc/elasticsearch-1.4.4

bin/elasticsearch -d(后台运行)



(需要kill -9 5360 才能停止Elasticsearch，可以安装servicewrapper插件解决)

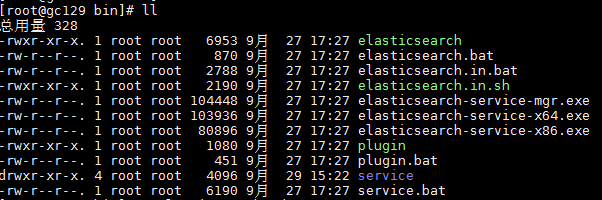
4)验证elasticsearch是否安装成功：



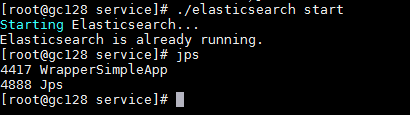
2.2elasticsearch-servicewrapper插件安装

1)下载地址https://github.com/elastic/elasticsearch-servicewrapper

2)安装：把server目录拷贝到ES\_HOME/bin/目录下面



1. 后台运行elasticsearch：ES\_HOME/bin/service/elasticsearch start
2. 后台停止elasticsearch：ES\_HOME/bin/service/elasticsearch stop



3和常见命令操作ES命令行操作

1)创建一个索引

现在让我们创建一个叫做“customer”的索引，然后再列出所有的索引：

curl -XPUT 'localhost:9200/customer?pretty'

curl 'localhost:9200/\_cat/indices?v'

第一个命令使用PUT创建了一个叫做“customer”的索引。我们简单地将pretty附加到调用的尾部，使其以美观的形式打印出JSON响应（如果有的话）。

响应如下：

curl -XPUT 'localhost:9200/customer?pretty'

{

"acknowledged" : true

}

curl 'localhost:9200/\_cat/indices?v'

health index pri rep docs.count docs.deleted store.size pri.store.size

yellow customer 5 1 0 0 495b 495b

第二个命令的结果告知我们，我们现在有一个叫做customer的索引，并且它有5个主分片和1份复制（都是默认值），其中包含0个文档。

你可能也注意到了这个customer索引有一个黄色健康标签。回顾我们之前的讨论，黄色意味着某些复制没有（或者还未）被分配。这个索引之所以这样，是 因为Elasticsearch默认为这个索引创建一份复制。由于现在我们只有一个节点在运行，那一份复制就分配不了了（为了高可用），直到当另外一个节 点加入到这个集群后，才能分配。一旦那份复制在第二个节点上被复制，这个节点的健康状态就会变成绿色。

2)索引并查询一个文档  
    现在让我们放一些东西到customer索引中。首先要知道的是，为了索引一个文档，我们必须告诉Elasticsearch这个文档要到这个索引的哪个类型（type）下。  
    让我们将一个简单的客户文档索引到customer索引、“external”类型中，这个文档的ID是1，操作如下：  
        curl -XPUT 'localhost:9200/customer/external/1?pretty' -d '  
        {  
          "name": "John Doe"  
        }'      
    响应如下：  
        curl -XPUT 'localhost:9200/customer/external/1?pretty' -d '  
        {  
          "name": "John Doe"  
        }'  
        {  
          "\_index" : "customer",  
          "\_type" : "external",  
          "\_id" : "1",  
          "\_version" : 1,  
          "created" : true  
        }     
    从上面的响应中，我们可以看到，一个新的客户文档在customer索引和external类型中被成功创建。文档也有一个内部id 1， 这个id是我们在索引的时候指定的。  
    有一个关键点需要注意，Elasticsearch在你想将文档索引到某个索引的时候，并不强制要求这个索引被显式地创建。在前面这个例子中，如果customer索引不存在，Elasticsearch将会自动地创建这个索引。  
    现在，让我们把刚刚索引的文档取出来：  
       curl -XGET 'localhost:9200/customer/external/1?pretty'  
    响应如下：  
        curl -XGET 'localhost:9200/customer/external/1?pretty'  
        {  
          "\_index" : "customer",  
          "\_type" : "external",  
          "\_id" : "1",  
          "\_version" : 1,  
          "found" : true, "\_source" : { "name": "John Doe" }  
        }     
    除了一个叫做found的字段来指明我们找到了一个ID为1的文档，和另外一个字段——\_source——返回我们前一步中索引的完整JSON文档之外，其它的都没有什么特别之处。  
 2)删除一个文档  
    现在让我们删除我们刚刚创建的索引，并再次列出所有的索引：  
        curl -XDELETE 'localhost:9200/customer?pretty'  
        curl 'localhost:9200/\_cat/indices?v'     
    响应如下：  
        curl -XDELETE 'localhost:9200/customer?pretty'  
        {  
          "acknowledged" : true  
        }  
        curl 'localhost:9200/\_cat/indices?v'  
        health index pri rep docs.count docs.deleted store.size pri.store.size  
    这表明我们成功地删除了这个索引，现在我们回到了集群中空无所有的状态。  
    在更进一步之前，我们再细看一下一些我们学过的API命令：  
        curl -XPUT 'localhost:9200/customer'  
        curl -XPUT 'localhost:9200/customer/external/1' -d '  
        {  
          "name": "John Doe"  
        }'  
        curl 'localhost:9200/customer/external/1'  
        curl -XDELETE 'localhost:9200/customer'

3)修改你的数据  
    Elasticsearch提供了近乎实时的数据操作和搜索功能。默认情况下，从你索引/更新/删除你的数据动作开始到它出现在你的搜索结果中，大概会有1秒钟的延迟。这和其它类似SQL的平台不同，数据在一个事务完成之后就会立即可用。  
索引/替换文档  
    我们先前看到，怎样索引一个文档。现在我们再次调用那个命令：  
            curl -XPUT 'localhost:9200/customer/external/1?pretty' -d '  
            {  
              "name": "John Doe"  
            }'  
     再次，以上的命令将会把这个文档索引到customer索引、external类型中，其ID是1。如果我们对一个不同（或相同）的文档应用以上的命令，Elasticsearch将会用一个新的文档来替换（重新索引）当前ID为1的那个文档。  
            curl -XPUT 'localhost:9200/customer/external/1?pretty' -d '  
            {  
              "name": "Jane Doe"  
            }'  
      以上的命令将ID为1的文档的name字段的值从“John Doe”改成了“Jane Doe”。如果我们使用一个不同的ID，一个新的文档将会被索引，当前已经在索引中的文档不会受到影响。  
            curl -XPUT 'localhost:9200/customer/external/2?pretty' -d '  
            {  
              "name": "Jane Doe"  
            }'  
        以上的命令，将会索引一个ID为2的新文档。  
      在索引的时候，ID部分是可选的。如果不指定，Elasticsearch将产生一个随机的ID来索引这个文档。Elasticsearch生成的ID会作为索引API调用的一部分被返回。  
      以下的例子展示了怎样在没有指定ID的情况下来索引一个文档：  
            curl -XPOST 'localhost:9200/customer/external?pretty' -d '  
            {  
              "name": "Jane Doe"  
            }'    
     注意，在上面的情形中，由于我们没有指定一个ID，我们使用的是POST而不是PUT。  
更新文档  
    除了可以索引、替换文档之外，我们也可以更新一个文档。但要注意，Elasticsearch底层并不支持原地更新。在我们想要做一次更新的时候，Elasticsearch先删除旧文档，然后在索引一个更新过的新文档。  
    下面的例子展示了怎样将我们ID为1的文档的name字段改成“Jane Doe”：  
        curl -XPOST 'localhost:9200/customer/external/1/\_update?pretty' -d '  
        {  
          "doc": { "name": "Jane Doe" }  
        }'   
    下面的例子展示了怎样将我们ID为1的文档的name字段改成“Jane Doe”的同时，给它加上age字段：  
        curl -XPOST 'localhost:9200/customer/external/1/\_update?pretty' -d '  
        {  
          "doc": { "name": "Jane Doe", "age": 20 }  
        }'  
    更新也可以通过使用简单的脚本来进行。这个例子使用一个脚本将age加5：  
        curl -XPOST 'localhost:9200/customer/external/1/\_update?pretty' -d '  
        {  
          "script" : "ctx.\_source.age += 5"  
        }'      
    在上面的例子中，ctx.\_source指向当前要被更新的文档。  
    注意，在写作本文时，更新操作只能一次应用在一个文档上。将来，Elasticsearch将提供同时更新符合指定查询条件的多个文档的功能（类似于SQL的UPDATE-WHERE语句）。  
4)删除文档  
    删除文档是相当直观的。以下的例子展示了我们怎样删除ID为2的文档：  
        curl -XDELETE 'localhost:9200/customer/external/2?pretty'  
    我们也能够一次删除符合某个查询条件的多个文档。以下的例子展示了如何删除名字中包含“John”的所有的客户：  
        curl -XDELETE 'localhost:9200/customer/external/\_query?pretty' -d '  
        {  
          "query": { "match": { "name": "John" } }  
        }'     
    注意，以上的URI变成了/\_query，以此来表明这是一个“查询删除”API，其中删除查询标准放在请求体中，但是我们仍然使用DELETE。现在先不要担心查询语法，我们将会在本教程后面的部分中涉及。  
5)批处理:  
    除了能够对单个的文档进行索引、更新和删除之外，Elasticsearch也提供了以上操作的批量处理功能，这是通过使用\_bulk API实现的。这个功能之所以重要，在于它提供了非常高效的机制来尽可能快的完成多个操作，与此同时使用尽可能少的网络往返。  
    作为一个快速的例子，以下调用在一次bulk操作中索引了两个文档（ID 1 - John Doe and ID 2 - Jane Doe）:  
        curl -XPOST 'localhost:9200/customer/external/\_bulk?pretty' -d '  
        {"index":{"\_id":"1"}}  
        {"name": "John Doe" }  
        {"index":{"\_id":"2"}}  
        {"name": "Jane Doe" }'  
    以下例子在一个bulk操作中，首先更新第一个文档（ID为1），然后删除第二个文档（ID为2）：  
        curl -XPOST 'localhost:9200/customer/external/\_bulk?pretty' -d '  
        {"update":{"\_id":"1"}}  
        {"doc": { "name": "John Doe becomes Jane Doe" } }  
        {"delete":{"\_id":"2"}}'  
    注意上面的delete动作，由于删除动作只需要被删除文档的ID，所以并没有对应的源文档.

PUT插入索引数据(幂等操作)

IMG_256

IMG_256IMG_256

POST插入或修改索引数据

IMG_256

IMG_256

如果想要确定我们创建的都是全新的数据

1：使用随机ID(post方式)

2：在url后面添加参数

curl -XPUT ['localhost:9200/crxy/emp/2?op\_type=create](http://localhost:9200/crxy/emp/2?op_type=create)' -d '{"name":"ls","age":25}'

curl -XPUT '[localhost:9200/crxy/emp/2/\_create](http://localhost:9200/crxy/emp/2/_create)' -d '{"name":"zs","age":25}'

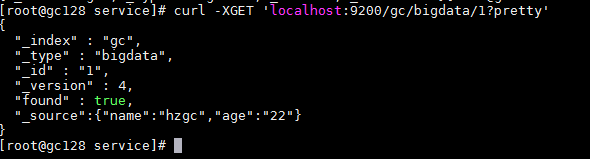
如果成功创建了新的文档，Elasticsearch将会返回常见的元数据以及201 Created的HTTP反馈码。而如果存在同名文件，Elasticsearch将会返回一个409 Conflict的HTTP反馈码

IMG_256

**局部更新**，可以添加新字段或者更新已有字段（必须使用POST）

curl -XPOST localhost:9200/gc/bigdata/1/\_update -d '{"doc":{"name":"hzgc"}}'

IMG_256

GET获取索引数据(格式如上)

查询所有：

curl -XGET localhost:9200/gc/bigdata/\_search

根据条件查询：

curl -XGET localhost:9200/gc/bigdata/\_search?q=name:ls

DELETE删除索引数据

curl -XDELETE localhost:9200/gc/bigdata/1/

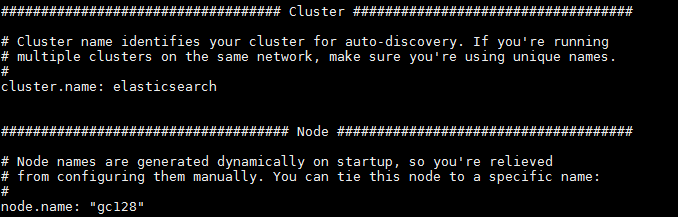
IMG_256

删除一个文档也不会立即生效，它只是被标记成已删除。Elasticsearch将会在你之后添加更多索引的时候才会在后台进行删除内容的清理

3、ES集群部署

对每个节点下ES\_HOME/config目录下的elasticsearch.yml进行修改

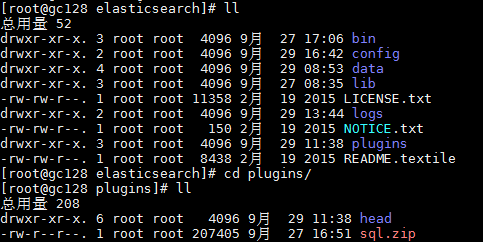
每个节点下的cluster.name保持一致(node.name可以不修改),每个节点的http.port设置不一样(我设的分别为9200/19200/29200)



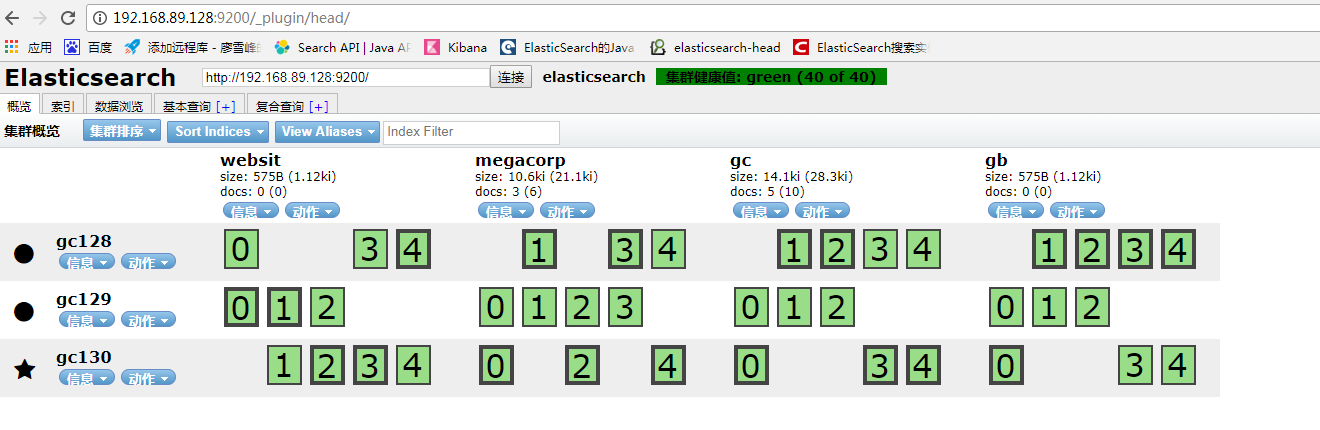
IMG_256

**ElasticSearch\_Head配置(配置后scp到每个节点)**

1. cd ES\_HOME/bin目录下
2. 执行命令 ./plugininstall mobz/elasticsearch-head
3. 下载安装包
4. 在准备下载包的时候我们已经将包下载到电脑本地了，所以讲下载下来 的包(elasticsearch-head-master)解压elasticsearch-head-master文件夹。
5. 在Elasticsearch的安装的plugin下创建目录head
6. cd ES\_HOME/plugins/
7. mkdir head
8. 跳转到head文件夹下，将刚才解压的elasticsearch-head-master文件夹下的所有文件拷贝到head目录下

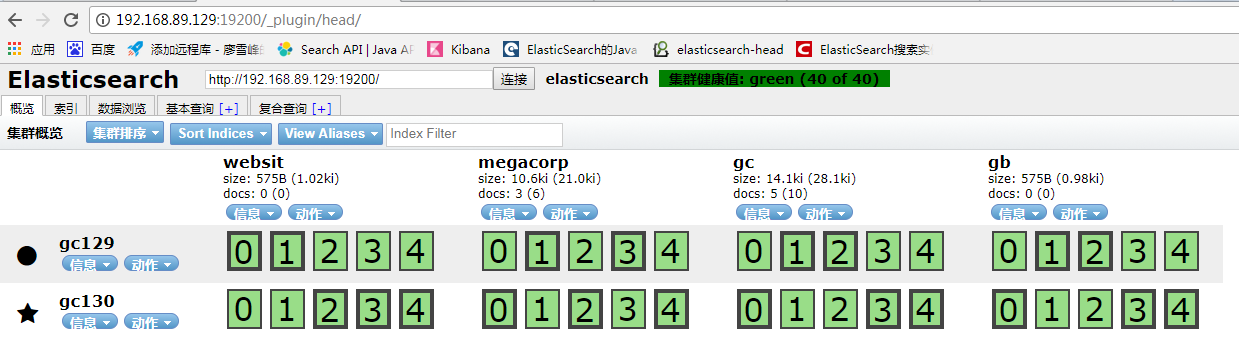


9)在每个节点下启动 ES\_HOME/bin/service/elasticsearch start



4、集群节点上下线情况

1)gc128节点下线



1. gc129节点下线



1. gc128、gc129节点重新上线(刚上线刷新网页可能没有分片，等一会刷新，分片重新分配)

