ElasticSearch系列文章：集群操作

rest 接口

现在我们已经有一个正常运行的节点（和集群），下一步就是要去理解怎样与其通信。幸运的是，Elasticsearch提供了非常全面和强大的REST API，利用这个REST API你可以同你的集群交互。下面是利用这个API，可以做的几件事情：

1、查你的集群、节点和索引的健康状态和各种统计信息
2、管理你的集群、节点、索引数据和元数据
3、对你的索引进行 CRUD（创建、读取、更新和删除）和搜索操作
4、执行高级的查询操作， 像是分页、排序、过滤、脚本编写（scripting）、小平面刻画（faceting）、聚合（aggregations）和许多其它操作

集群健康(cluster health)

让我们以基本的健康检查作为开始，我们可以利用它来查看我们集群的状态。我们使用curl，当然你也可以使用任何可以创建HTTP/REST调用的工具，来使用该功能。
我们假设我们还在我们启动Elasticsearch的节点上并打开另外一个shell窗口。

要检查集群健康，我们将使用_cat API。需要事先记住的是，我们的节点HTTP的端口是9200：

```
curl 'localhost:9200/_cat/health?v'
```

相应的响应是：

```
epoch      timestamp cluster       status node.total node.data shards pri relo init unassign
1394735289 14:28:09  elasticsearch green          1         1      0   0    0    0        0
```

可以看到，我们集群的名字是“elasticsearch”，正常运行，并且状态是绿色。

当我们查看集群状态的时候，我们可能得到绿色、黄色或红色三种状态。绿色代表一切正常（集群功能齐全）；黄色意味着所有的数据都是可用的，但是某些复制没有被分配（集群功能齐全）； 红色则代表因为某些原因，某些数据不可用。注意，即使是集群状态是红色的，集群仍然是部分可用的（它仍然会利用可用的分片来响应搜索请求），但是可能你需要尽快修复它，因为你有丢失的数据。

从上面的响应中，我们可以看到 一共有一个节点，由于里面没有数据，我们有0个分片。注意，由于我们使用默认的集群名字（elasticsearch），并且由于Elasticsearch默认使用网络多播（multicast）发现其它节点， 如果你在你的网络中启动了多个节点，
你就已经把它们加入到集群中了。在这种情形下，你可能在上面的响应中看到多个节点。

我们也可以获得节集群中的节点列表：

```
curl 'localhost:9200/_cat/nodes?v'
```

对应的响应是:

```
curl 'localhost:9200/_cat/nodes?v'
host        ip        heap.percent ram.percent load node.role master name
mwubuntu1   127.0.1.1            8          4 0.00 d         *      New Goblin
```

这儿， 我们可以看到叫“New Goblin”的节点，这个节点是我们集群中的唯一节点。

列出所有的索引

让我们看一下我们的索引：

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

对应的响应是:

```
curl 'localhost:9200/_cat/indices?v'
health index pri rep docs.count docs.deleted store.size pri.store.size
```

这个结果意味着，在我们的集群中没有任何索引。

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

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。

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。

需要注意的是，当你想将文档索引到某个索引的时候，Elasticsearch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found字段-（指明我们找到了一个ID为1的文档）和_source字段（返回我们前一步中索引的完整JSON文档）之外，没有什么特别之处。

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。

我们细看一下我们学过的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'
```

仔细研究以上的命令，我们可以发现访问Elasticsearch中数据的一个模式。这个模式可以被总结为：

```
curl -X<REST Verb> <Node>:<Port>/<Index>/<Type>/<ID>
```

这个REST访问模式普遍适用于所有的API命令，如果你能记住它，你就会为掌握Elasticsearch开一个好头。