**Elasticsearch**

**集群（附加）**

# **第一章 ElasticSearch集群**

## **集群的相关概念**

### **集群 cluster**

一个集群就是由一个或多个节点组织在一起，它们共同持有整个的数据，并一起提供索引和搜索功能。一个集群由一个唯一的名字标识，这个名字默认就是“elasticsearch”。这个名字是重要的，因为一个节点只能通过指定某个集群的名字，来加入这个集群

### **节点 node**

一个节点是集群中的一个服务器，作为集群的一部分，它存储数据，参与集群的索引和搜索功能。和集群类似，一个节点也是由一个名字来标识的，默认情况下，这个名字是一个随机的漫威漫画角色的名字，这个名字会在启动的时候赋予节点。这个名字对于管理工作来说挺重要的，因为在这个管理过程中，你会去确定网络中的哪些服务器对应于Elasticsearch集群中的哪些节点。

一个节点可以通过配置集群名称的方式来加入一个指定的集群。默认情况下，每个节点都会被安排加入到一个叫做“elasticsearch”的集群中，这意味着，如果你在你的网络中启动了若干个节点，并假定它们能够相互发现彼此，它们将会自动地形成并加入到一个叫做“elasticsearch”的集群中。

在一个集群里，只要你想，可以拥有任意多个节点。而且，如果当前你的网络中没有运行任何Elasticsearch节点，这时启动一个节点，会默认创建并加入一个叫做“elasticsearch”的集群。

### **分片和复制 shards&replicas**

一个索引可以存储超出单个结点硬件限制的大量数据。比如，一个具有10亿文档的索引占据1TB的磁盘空间，而任一节点都没有这样大的磁盘空间；或者单个节点处理搜索请求，响应太慢。为了解决这个问题，Elasticsearch提供了将索引划分成多份的能力，这些份就叫做分片。当你创建一个索引的时候，你可以指定你想要的分片的数量。每个分片本身也是一个功能完善并且独立的“索引”，这个“索引”可以被放置到集群中的任何节点上。分片很重要，主要有两方面的原因： 1）允许你水平分割/扩展你的内容容量。 2）允许你在分片（潜在地，位于多个节点上）之上进行分布式的、并行的操作，进而提高性能/吞吐量。

至于一个分片怎样分布，它的文档怎样聚合回搜索请求，是完全由Elasticsearch管理的，对于作为用户的你来说，这些都是透明的。

在一个网络/云的环境里，失败随时都可能发生，在某个分片/节点不知怎么的就处于离线状态，或者由于任何原因消失了，这种情况下，有一个故障转移机制是非常有用并且是强烈推荐的。为此目的，Elasticsearch允许你创建分片的一份或多份拷贝，这些拷贝叫做复制分片，或者直接叫复制。

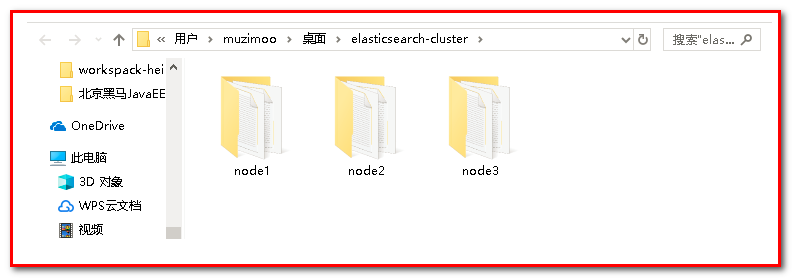
复制之所以重要，有两个主要原因： 在分片/节点失败的情况下，提供了高可用性。因为这个原因，注意到复制分片从不与原/主要（original/primary）分片置于同一节点上是非常重要的。扩展你的搜索量/吞吐量，因为搜索可以在所有的复制上并行运行。总之，每个索引可以被分成多个分片。一个索引也可以被复制0次（意思是没有复制）或多次。一旦复制了，每个索引就有了主分片（作为复制源的原来的分片）和复制分片（主分片的拷贝）之别。分片和复制的数量可以在索引创建的时候指定。在索引创建之后，你可以在任何时候动态地改变复制的数量，但是你事后不能改变分片的数量。

默认情况下，Elasticsearch中的每个索引被分片5个主分片和1个复制，这意味着，如果你的集群中至少有两个节点，你的索引将会有5个主分片和另外5个复制分片（1个完全拷贝），这样的话每个索引总共就有10个分片。

## **集群的搭建**

### **准备三台elasticsearch服务器**

创建elasticsearch-cluster文件夹，在内部复制三个elasticsearch服务



### **修改每台服务器配置**

修改elasticsearch-cluster\node\*\config\elasticsearch.yml配置文件

#### **node1节点：**

http.cors.enabled: true

http.cors.allow-origin: "\*"

#节点1的配置信息：

#集群名称，保证唯一

cluster.name: my-elasticsearch

#节点名称，必须不一样

node.name: node-1

#必须为本机的ip地址

network.host: 127.0.0.1

#服务端口号，在同一机器下必须不一样

http.port: 9200

#集群间通信端口号，在同一机器下必须不一样

transport.tcp.port: 9300

#设置集群自动发现机器ip集合

discovery.zen.ping.unicast.hosts: ["127.0.0.1:9300","127.0.0.1:9301","127.0.0.1:9302"]

#### **node2节点：**

http.cors.enabled: true

http.cors.allow-origin: "\*"

#节点2的配置信息：

#集群名称，保证唯一

cluster.name: my-elasticsearch

#节点名称，必须不一样

node.name: node-2

#必须为本机的ip地址

network.host: 127.0.0.1

#服务端口号，在同一机器下必须不一样

http.port: 9201

#集群间通信端口号，在同一机器下必须不一样

transport.tcp.port: 9301

#设置集群自动发现机器ip集合

discovery.zen.ping.unicast.hosts: ["127.0.0.1:9300","127.0.0.1:9301","127.0.0.1:9302"]

#### **node3节点：**

http.cors.enabled: true

http.cors.allow-origin: "\*"

#节点3的配置信息：

#集群名称，保证唯一

cluster.name: my-elasticsearch

#节点名称，必须不一样

node.name: node-3

#必须为本机的ip地址

network.host: 127.0.0.1

#服务端口号，在同一机器下必须不一样

http.port: 9202

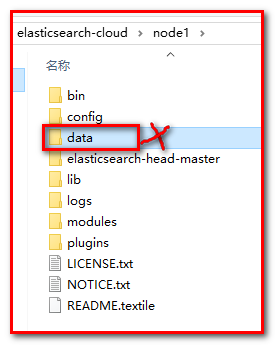
#集群间通信端口号，在同一机器下必须不一样

transport.tcp.port: 9302

#设置集群自动发现机器ip集合

discovery.zen.ping.unicast.hosts: ["127.0.0.1:9300","127.0.0.1:9301","127.0.0.1:9302"]

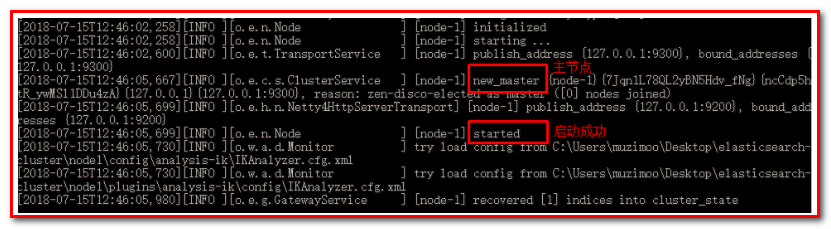
注意：这里需要删除node1，node2，node3的data文件，防止数据冲突



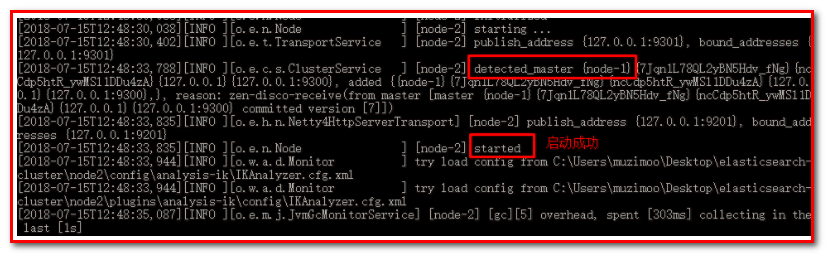
### **启动各个节点服务器**

双击elasticsearch-cluster\node\*\bin\elasticsearch.bat

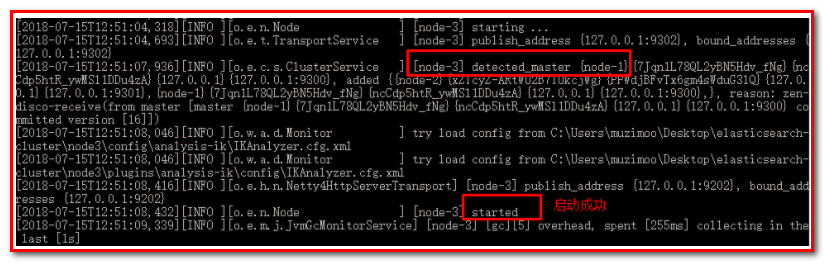
#### **启动节点1：**



#### **启动节点2：**



#### **启动节点3：**



### **集群测试（postman，restful风格）**

#### **添加索引和映射**

PUT localhost:9200/blog1

{

"mappings": {

"article": {

"properties": {

"id": {

"type": "long",

"store": true,

"index":"not\_analyzed"

},

"title": {

"type": "text",

"store": true,

"index":"analyzed",

"analyzer":"standard"

},

"content": {

"type": "text",

"store": true,

"index":"analyzed",

"analyzer":"standard"

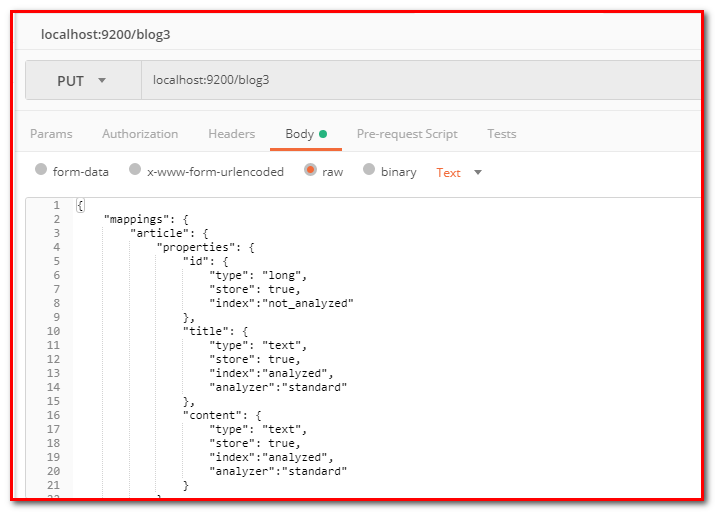
}

}

}

}

}



#### **添加文档**

POST localhost:9200/blog1/article/1

{

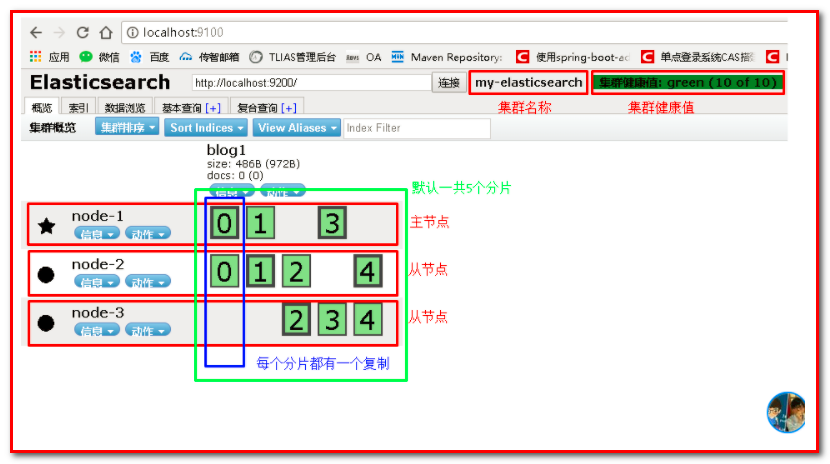
"id":1,

"title":"ElasticSearch是一个基于Lucene的搜索服务器",

"content":"它提供了一个分布式多用户能力的全文搜索引擎，基于RESTfulweb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，是当前流行的企业级搜索引擎。设计用于云计算中，能够时搜索，稳定，可靠，快速，安装使用方便。"

}

#### **使用elasticsearch-header查看集群情况**



### 使用java测试集群

#### 添加索引和映射

@Test  
 *//创建映射* **public void** test1() **throws** Exception{  
  
 Map<String, String> map = **new** HashMap();  
 map.put(**"cluster.name"**, **"my-elasticsearch"**);  
 Settings settings = Settings.*builder*().put(map).build();  
 *// 创建Client连接对象* TransportClient client = **new** PreBuiltTransportClient(settings)  
 .addTransportAddress(**new** InetSocketTransportAddress(InetAddress.*getByName*(**"127.0.0.1"**), 9300));  
  
 *//创建名称为blog2的索引* client.admin().indices().prepareCreate(**"blog2"**).get();  
 *// 添加映射  
 /\*\*  
  \* 格式：  
  \* "mappings" : {  
      "article" : {  
           "dynamic" : "false",  
           "properties" : {  
              "id" : { "type" : "string" },  
              "content" : { "type" : "string" },  
              "author" : { "type" : "string" }  
           }  
       }  
    }  
  \*/* XContentBuilder builder = XContentFactory.*jsonBuilder*()  
 .startObject()  
 .startObject(**"article"**)  
 .startObject(**"properties"**)  
 .startObject(**"id"**)  
 .field(**"type"**,**"long"**)  
 .endObject()  
 .startObject(**"title"**)  
 .field(**"type"**,**"string"**)  
 .field(**"store"**,**"true"**)  
 .field(**"analyzer"**,**"ik\_smart"**)  
 .endObject()  
 .startObject(**"content"**)  
 .field(**"type"**,**"string"**)  
 .field(**"store"**,**"true"**)  
 .field(**"analyzer"**,**"ik\_smart"**)  
 .endObject()  
 .endObject()  
 .endObject()  
 .endObject();  
 *// 创建映射* PutMappingRequest mapping = Requests.*putMappingRequest*(**"blog2"**).type(**"article"**).source(builder);  
 client.admin().indices().putMapping(mapping).get();  
 *//释放资源* client.close();  
 }

#### 添加文档

@Test  
 *//创建文档* **public void** test2() **throws** Exception {  
 Map<String, String> map = **new** HashMap();  
 map.put(**"cluster.name"**, **"my-elasticsearch"**);  
 Settings settings = Settings.*builder*().put(map).build();  
 *//创建客户端访问对象* TransportClient client = **new** PreBuiltTransportClient(settings).addTransportAddress(  
 **new** InetSocketTransportAddress(InetAddress.*getByName*(**"127.0.0.1"**), 9300));  
  
  
 *//创建文档* XContentBuilder builder = XContentFactory.*jsonBuilder*()  
 .startObject()  
 .field(**"id"**,1)  
 .field(**"title"**,**"ElasticSearch是一个基于Lucene的搜索服务器123"**)  
 .field(**"content"**,**"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\n"** +  
 **"用Java开发的，并作为Apache许可条款下的开放源码发布，是当前流行的企业级搜索引擎。设计用于云计算中，能\n"** +  
 **"够达到实时搜索，稳定，可靠，快速，安装使用方便。"**)  
 .endObject();  
  
 *//添加文档到指定索引库  
 /\*\*  
 \* 参数一blog1：表示索引对象  
 \* 参数二article：类型  
 \* 参数三1：建立id  
 \*/* client.prepareIndex(**"blog2"**,**"article"**,**"1"**).setSource(builder).get();  
  
 *//关闭资源* client.close();  
 }