Elasticsearch基础笔记+项目图例

一 Elasticsearch概念：

1.Elasticsearch是位于Elastic Stack核心的分布式搜索和分析引擎。

2.Elasticsearch为所有类型的数据提供近乎实时的搜索和分析。无论您拥有结构化或非结构化文本、数字数据或地理空间数据，Elasticsearch都可以有效地存储并以支持快速搜索的方式对其进行索引。

3.Elasticsearch是一个分布式文档存储。Elasticsearch不是将信息存储为列数据行，而是存储已序列化为JSON文档的复杂数据结构。当集群中有多个Elasticsearch节点时，存储的文档分布在整个集群中，可以从任何节点立即访问它们。当一个文档被存储时，它被索引并在1秒内几乎实时地完全搜索。Elasticsearch使用一种称为倒索引的数据结构，它支持非常快速的全文搜索。倒排索引列出出现在任何文档中的每个惟一单词，并标识每个单词出现在所有文档中的所有文档。

二 基本操作+结合项目

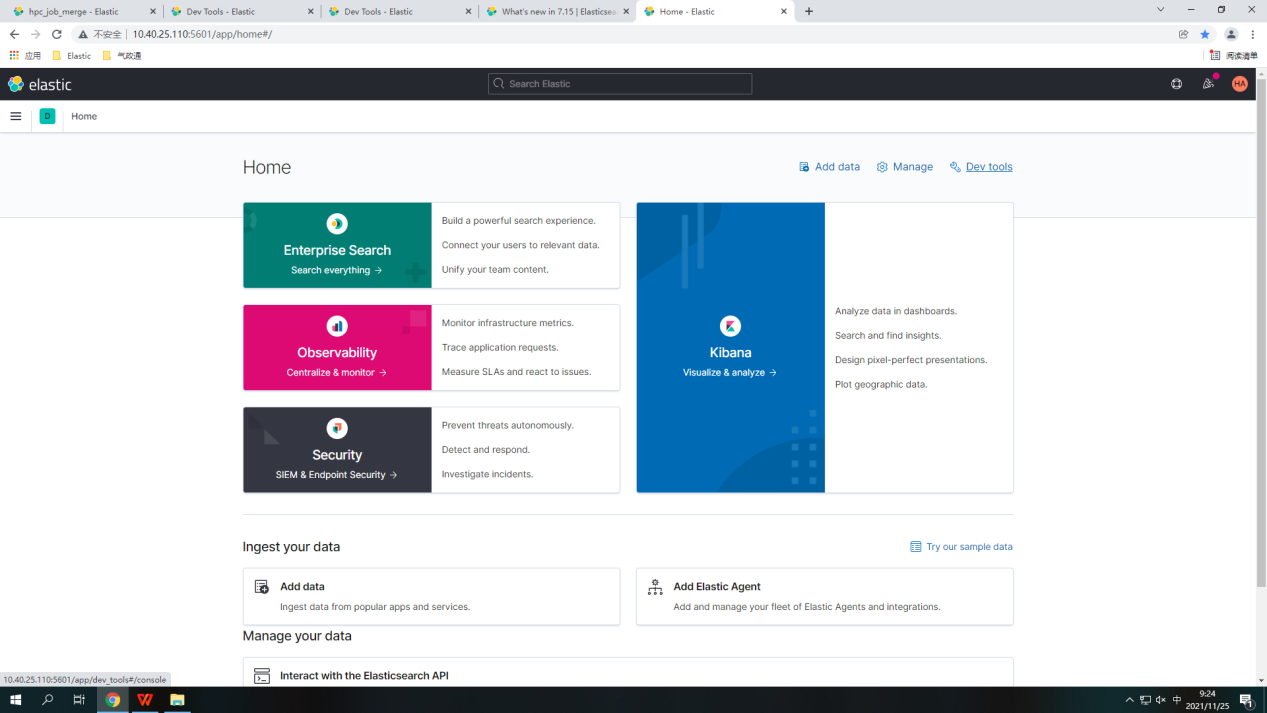
两种操作方式：

i:通过linux命令行直接操作数据。比如批量读取编辑好格式的文档，直接往引擎中录入数据。

ii：通过已经部署好的web Elasticsearch网站进行图形化界面操作（推荐\*\*\*）。http://10.40.25.110:5601/app/home#/

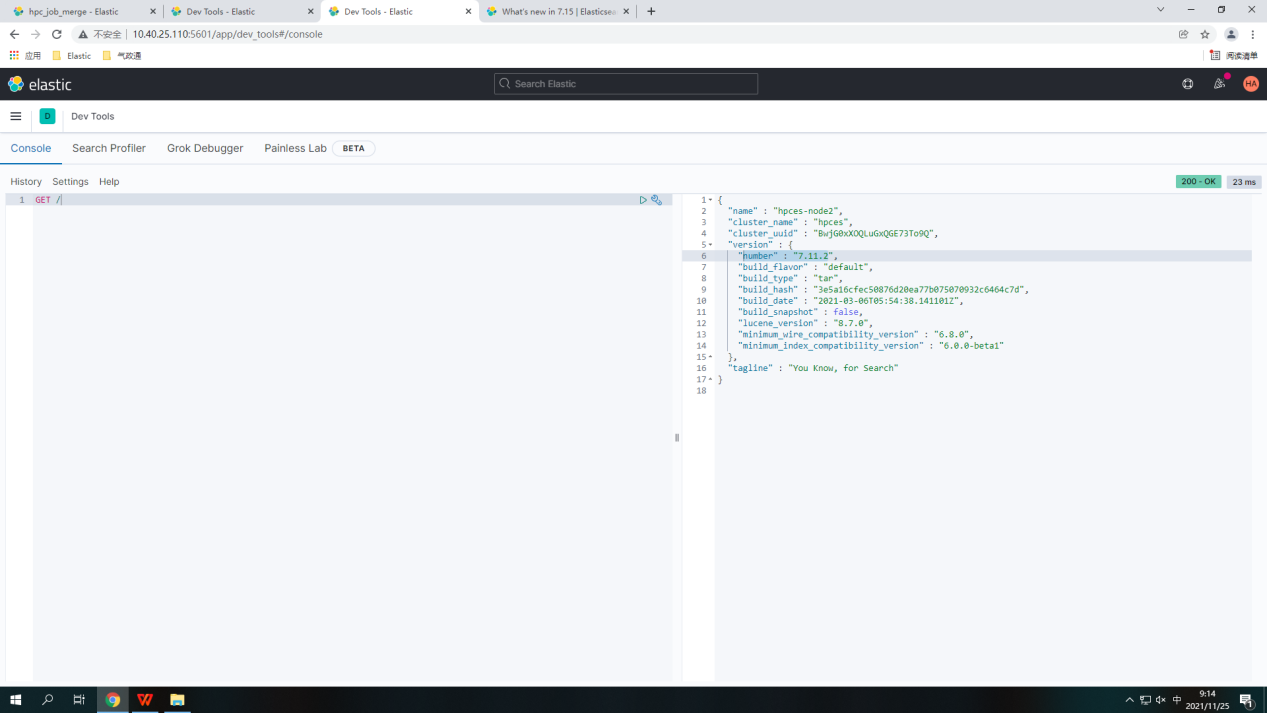
1 书写命令方法（以查看Elasticsearch版本为例）

首先打开图形化界面，认证登陆以后。在图中右上方找到“Dev Tools”工具。后续的增删改查操作也在“dev tools”中进行。



在dev tools中左边的console直接写入命令： GET /

这样就可以完成简单的版本查询。显示的版本结果如右边里面的version显示，为7.11.2.

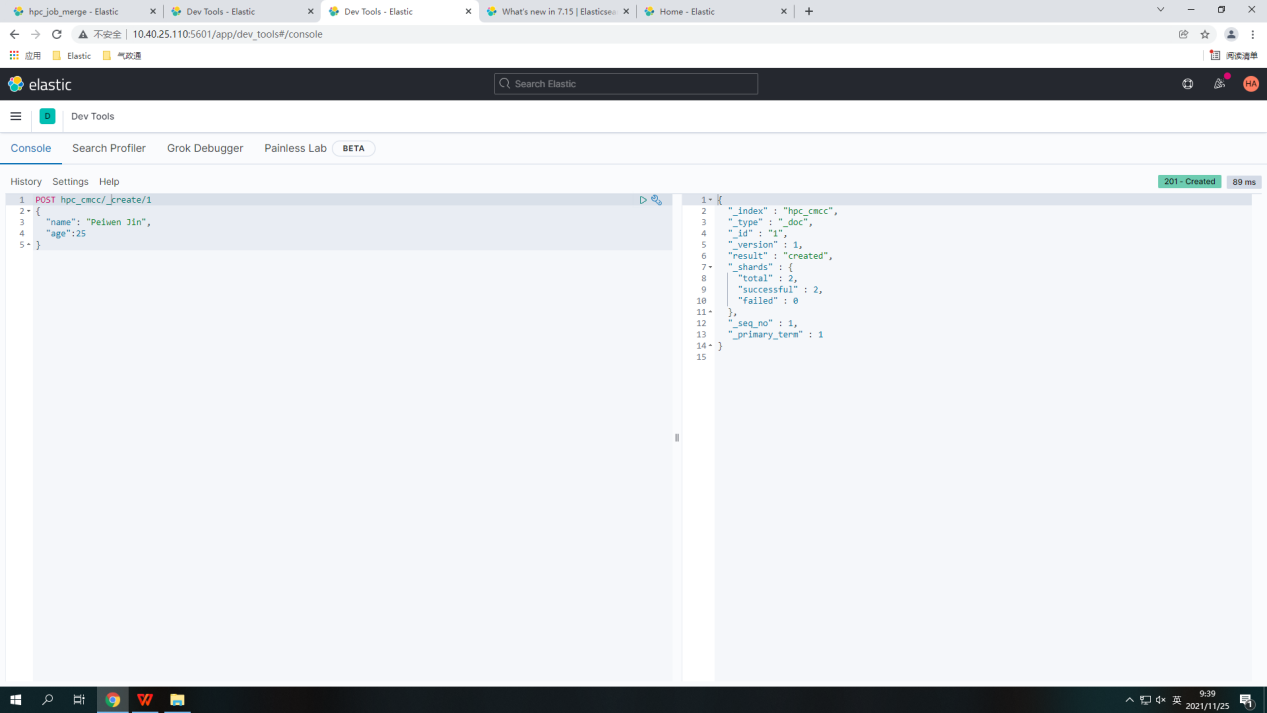


2 创建索引

|  |  |
| --- | --- |
| 输入 | 输出 |
| POST hpc\_cmcc/\_create/1  {  "name": "Peiwen Jin",  "age":25  } | {  "\_index" : "hpc\_cmcc",  "\_type" : "\_doc",  "\_id" : "1",  "\_version" : 1,  "result" : "created",  "\_shards" : {  "total" : 2,  "successful" : 2,  "failed" : 0  },  "\_seq\_no" : 1,  "\_primary\_term" : 1  } |

创建索引的格式为：

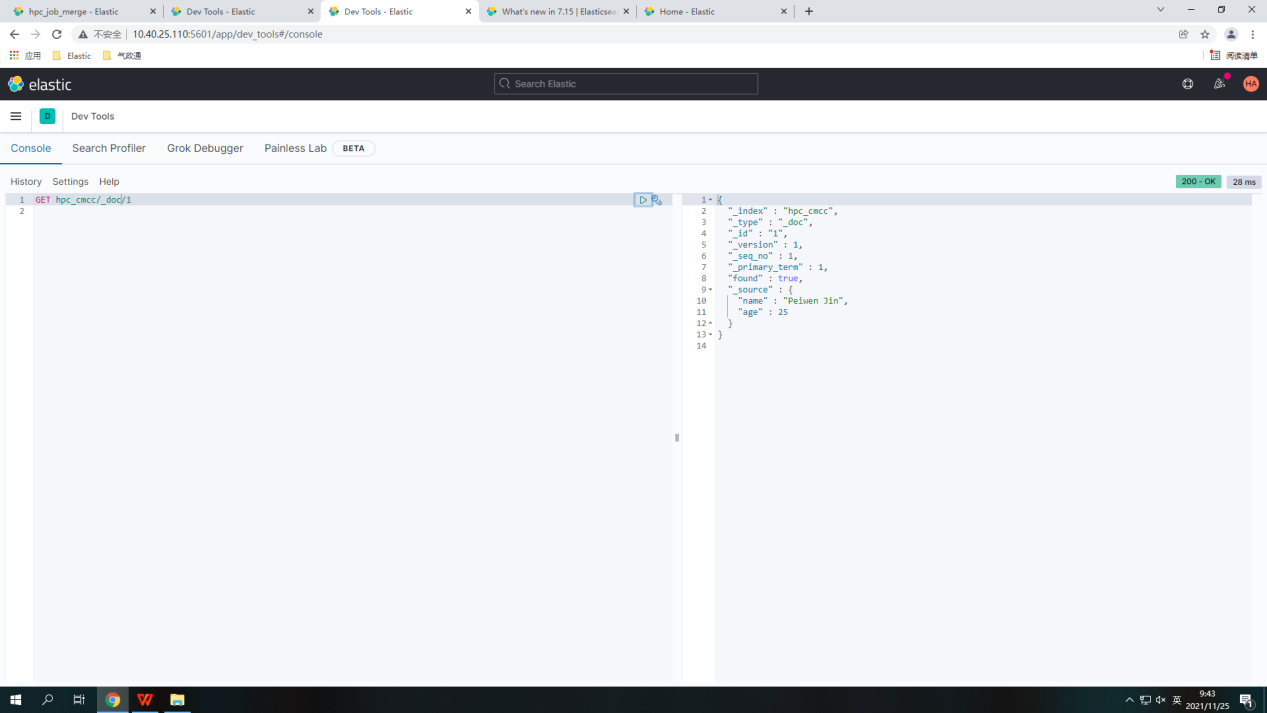
POST {index}/\_create/{id}



3 查看刚才创建的索引

|  |  |
| --- | --- |
| 输入 | 输出 |
| GET hpc\_cmcc/\_doc/1 | {  "\_index" : "hpc\_cmcc",  "\_type" : "\_doc",  "\_id" : "1",  "\_version" : 1,  "\_seq\_no" : 1,  "\_primary\_term" : 1,  "found" : true,  "\_source" : {  "name" : "Peiwen Jin",  "age" : 25  }  } |

查看索引格式为GET关键字+{index}+\_doc+{id}，返回的结果中注意“\_source”中就是我们刚才定义好的字段“name”和“age”。



4 删除索引

DELETE hpc\_cmcc/\_doc/1 仅删除指定索引的指定文档

DELETE hpc\_cmcc 删除整个索引

5 修改数据

PUT hpc\_cmcc/\_doc/1

{

“name”:”Peiwen Jin”,

“age”:25,

“sex”:”male”

}

这个修改操作是在上个创建索引实例中新增加了性别“sex”字段。同理，只要是针对指定索引下的指定文档，均可以按照需求修改数据。

6 bulk方法批量插入数据

PUT {index}/\_bulk

{ "create": { } }

{ "@timestamp": "2099-05-07T16:24:32.000Z", "event": { "original": "192.0.2.242 - - [07/May/2020:16:24:32 -0500] \"GET /images/hm\_nbg.jpg HTTP/1.0\" 304 0" } }

{ "create": { } }

{ "@timestamp": "2099-05-08T16:25:42.000Z", "event": { "original": "192.0.2.255 - - [08/May/2099:16:25:42 +0000] \"GET /favicon.ico HTTP/1.0\" 200 3638" } }

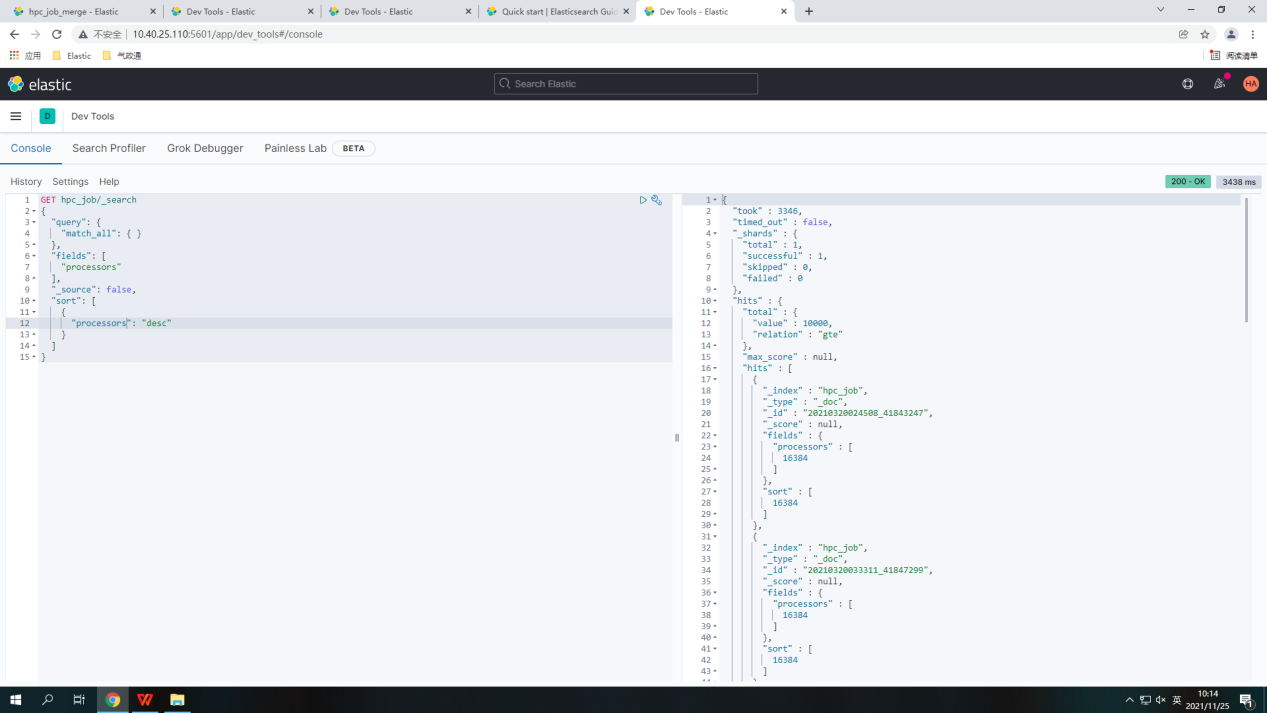
执行完毕后，再次获取数据查看，

GET {index}/\_search

7 获取特定field的信息

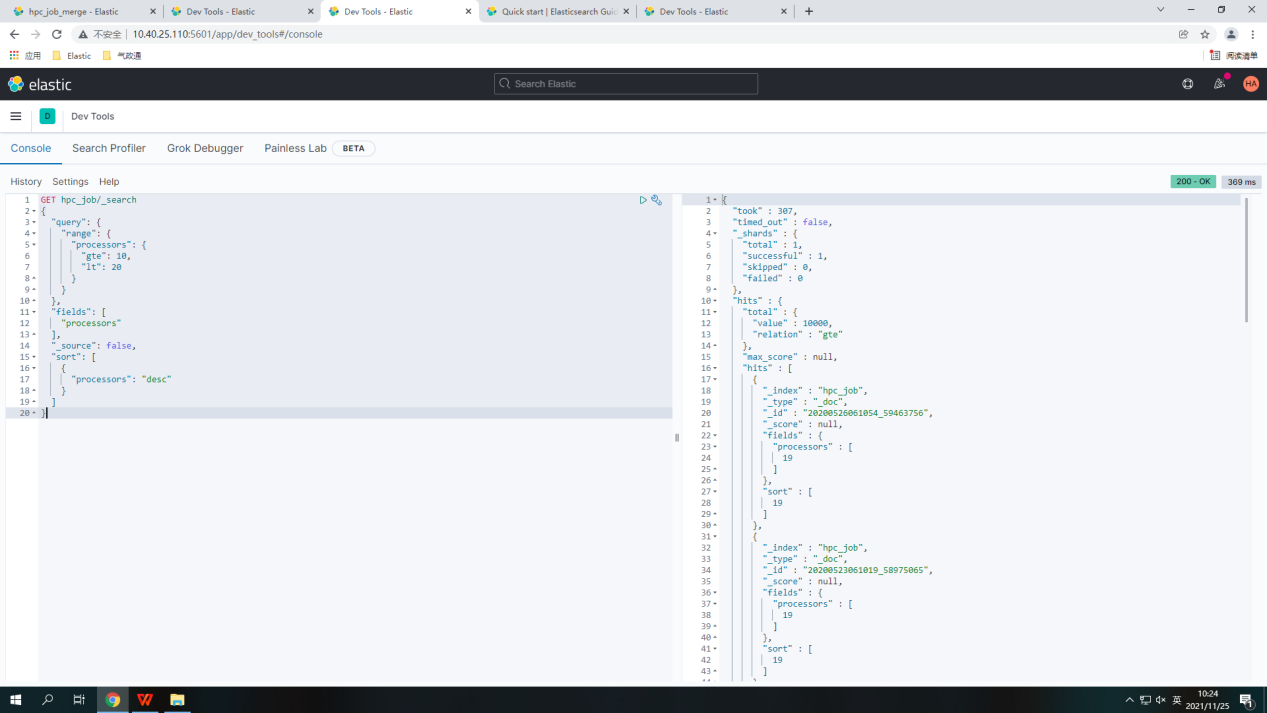
比如我想要获取hpc\_job中的特定的processors字段,并使它降序排列：

|  |  |
| --- | --- |
| 输入 | 输出 |
| GET hpc\_job/\_search  {  "query": {  "match\_all": { }  },  "fields": [  "processors"  ],  "\_source": false,  "sort": [  {  "processors": "desc"  }  ]  } | {  "took" : 3346,  "timed\_out" : false,  "\_shards" : {  "total" : 1,  "successful" : 1,  "skipped" : 0,  "failed" : 0  },  "hits" : {  "total" : {  "value" : 10000,  "relation" : "gte"  },  "max\_score" : null,  "hits" : [  {  "\_index" : "hpc\_job",  "\_type" : "\_doc",  "\_id" : "20210320024508\_41843247",  "\_score" : null,  "fields" : {  "processors" : [  16384  ]  },  "sort" : [  16384  ]  }, ... ... |



8 限定field的范围并选择性获取

|  |  |
| --- | --- |
| 输入 | 输出 |
| GET hpc\_job/\_search  {  "query": {  "range": {  "processors": {  "gte": 10,  "lt": 20  }  }  },  "fields": [  "processors"  ],  "\_source": false,  "sort": [  {  "processors": "desc"  }  ]  } | {  "took" : 307,  "timed\_out" : false,  "\_shards" : {  "total" : 1,  "successful" : 1,  "skipped" : 0,  "failed" : 0  },  "hits" : {  "total" : {  "value" : 10000,  "relation" : "gte"  },  "max\_score" : null,  "hits" : [  {  "\_index" : "hpc\_job",  "\_type" : "\_doc",  "\_id" : "20200526061054\_59463756",  "\_score" : null,  "fields" : {  "processors" : [  19  ]  },  "sort" : [  19  ]  },  { ... ... |



注：“gte” “>=”

“gt” “>”

“lt” “<”

“eq” “=”

“lte” “<=”

三 复杂操作

1 按照条件查询

GET cmcc/\_search

{

“query”:{

“match”:{

“province”:”广州”

}

}

}

GET cmcc/\_search表示我对cmcc这个索引进行查询，query是查询的关键字，match后面的省份，广州表示在cmcc索引中我需要所有满足省份这个字段等于广州的数据，并返回。

第二个例子是添加了逻辑与或非。

GET cmcc/\_search

{

“query”:{

“bool”:{

“should”[

{

“match”:{

“age”:21

}

},

“match”:{

“age”:23

}

]

}

}

}

在这个例子中，我们使用bool+should表示或逻辑，即我需要的数据应该噢满足年龄为21或者为23.

bool 查询可以接受 must（逻辑与） 、 must\_not（逻辑或） 和 should（逻辑非） 参数下的多个查询语句。

比如：

GET /my\_index/my\_type/\_search

{

"query": {

"bool": {

"must": { "match": { "title": "quick" }},

"must\_not": { "match": { "title": "lazy" }},

"should": [

{ "match": { "title": "brown" }},

{ "match": { "title": "dog" }}

]

}

}

}

以上的查询结果返回 title 字段包含词项 quick 但不包含 lazy 的任意文档。同时，这个返回的文档包含 brown 或 dog 这两个词项，当然也可能都包含。