ES介绍

# 1.ES是什么

ES是一个开源的高性能全文检索和分析引擎。

# 2.ES有什么用

ES可以用来快速的近实时的存储，搜索和分析大体积的的数据。

# 3.ES的使用场景，例子：

1.网店的产品搜索。

2.日志分析。

3.价格提醒。

# 4.ES的基础观念点：

1.近实时：也就是说es的文档从被建立到它可以被搜索到是有一个小延迟的，大概在1秒左右。

2.集群：一个或多个节点的集合。

3.节点：一个节点就是一个服务。

4.索引：文档的集合。

5.类型：一个索引可以有多种类型（6.0以上已废弃，未来会移除，所以现在的索引都是单类型的）。

6.文档：一个文档就是一个最基础的可以被索引的信息单元。

7.碎片&副本：碎片是为了解决一个节点内数据过大，通过水平分割数据将数据分散到不同服务器上。副本是为了容灾。

# 5.ES的安装

1.安装环境：jdk8及以上

2.操作系统：windows，linux，macos

3.默认端口：9200

4.启动文件位于安装目录的bin目录下

5.配置文件位于安装目录的config目录下

# 6.ES的配置

1.elasticsearch.yml ES的配置文件

2.jvm.options ES的jvm配置文件

3.log4j2.properties ES的日志记录配置文件

较常用的配置项有以下：

jvm.options中的-Xms、-Xmx用于设置jvm启动时申请的内存大小，内存小的机器上有用。

elasticsearch.yml中的cluster.name集群名字、node.name节点名字、path.data索引文件路径、path.logs日志路径、network.host绑定指定ip地址、http.port api通信端口。

# 7.ES的启动（以linux为例）

1.默认启动（适合开发时使用）

bin/elasticsearch

2.以守护进程的方式启动（适合线上使用）

bin/elasticsearch -s -d

3.指定集群名字和节点名字启动（适合集群时使用）

bin/elasticsearch -Ecluster.name=my\_cluster\_name -Enode.name=my\_node\_name

# 8.ES的基础操作

ES的增删改和搜索操作均通过REST API完成。同时API还提供集群、节点、索引的状态检测，数据管理等等。

1.索引的创建：

PUT test

{

"settings" : {

"number\_of\_shards" : 1,//碎片的数量（静态）

"number\_of\_replicas" : 2//副本的数量（动态）

},

"mappings" : {

"type1" : {//类型

"properties" : {

"field1" : { //列名

"type" : "text" ,//列类型

"analyzer": "standard”//分词器

}

}

}

}

}

2.索引的删除

DELETE /test

3.获取索引

GET /test

4.文档的创建

POST twitter/tweet/

{

"user" : "kimchy",

"post\_date" : "2009-11-15T14:12:12",

"message" : "trying out Elasticsearch"

}

5.文档的获取

GET twitter/tweet/0

6.文档的删除

DELETE /twitter/tweet/1

7.文档的更新

PUT test/type1/1

{

"counter" : 1,

"tags" : ["red"]

}

# 9.PHPSDK的常用操作

1.实例化es的操作类：Elasticsearch\ClientBuilder::fromConfig($config) 或者 Elasticsearch\ClientBuilder::create()→build()

2.创建新文档操作：$client→create($params)

$params[“index”] = 索引的名字（必须）

[“type”] = 索引的类型（必须）

[“body”] = 文档内容

3.通过查询语句更新文档内容：$client→updateByQuery($params)

$params[“index”] = 索引的名字（必须）

[“type”] = 索引的类型（必须）

[“body”] = 查询语句

4.通过查询语句删除文档：$client→deleteByQuery($params)

$params[“index”] = 索引的名字（必须）

[“type”] = 索引的类型（必须）

[“body”] = 查询语句

5.查询：$client→search($params)

$params[“index”] = 索引的名字（必须）

[“type”] = 索引的类型（必须）

[“body”] = 查询语句

# 10.查询语句DSL简介

1.DSL查询语句由两种类型组成：

1.叶查询子句：是指通过查询特定的字段内的特定值，比如match，term和range查询。（可以独立用于查询）

2.复合查询子句：是指包含其他叶查询或复合查询并进行组合条件的查询,例如bool和dis\_max。（它的内部必定要包含至少一个的叶查询才能正常工作，也就是不能独立工作）

2.查询语句的依赖上下文环境：

1.query上下文：在这种环境下除了决定文档是否匹配还会对文档进行得分计算。（没有缓存）

2.filter上下文：在这种环境下只计算文档是否匹配，不进行分数计算。（筛选条件会缓存）

3.复合查询的几种类型：

1.boosting查询：在匹配条件中你可以设置减分条件和减去的分数，当匹配到的文档中包含有你设定的减分条件时会减去一个你设定的分数。

2.constant score查询：分值由你决定，所有匹配到的文档返回分值均为此值。

3.function score查询：类似constant score查询，不同之处在于分值是一个函数，这个函数可以是你自定义的也可以使用它内置提供的。

4.dis max查询：类似bool查询，区别在于计算一个文档的得分方式。dis max的计算方式为加重每个子查询中得分最高的分数的权重来计算得分。

5.bool查询：组合查询，遵循匹配越多越好的原则，会累计每个匹配的must和should子句的分数作为匹配文档最后的得分。每个子句都有一种事件类型，事件类型有4种分别是：

must：子句必须出现在匹配的文档中并且会将其分数纳入计算。

filter：子句必须出现在匹配的文档中，但是忽略其分数。

should：在query上下文环境中并且至少有一个must或filter子句那么should子句不起作用。在filter上下文环境中或者子句中均不存在must和filter子句，那么至少匹配一个should子句。最小匹配子句数量可以通过minimum\_should\_match进行设置。

must\_not：子句必须不能出现在匹配的文档中，不计算其分数。