# Elasticsearch

## Elasticsearch概述

### Elasticsearch 是什么？

The Elastic Stack, 包括 Elasticsearch、Kibana、Beats 和 Logstash（也称为 ELK Stack）。能够安全可靠地获取任何来源、任何格式的数据，然后实时地对数据进行搜索、分析和可视化。Elaticsearch，简称为 ES，ES 是一个开源的高扩展的分布式全文搜索引擎，是整个 ElasticStack 技术栈的核心。它可以近乎实时的存储、检索数据；本身扩展性很好，可以扩展到上百台服务器，处理 PB 级别的数据。

### 全文搜索引擎

Google，百度类的网站搜索，它们都是根据网页中的关键字生成索引，我们在搜索的时候输入关键字，它们会将该关键字即索引匹配到的所有网页返回；还有常见的项目中应用日志的搜索等等。对于这些非结构化的数据文本，关系型数据库搜索不是能很好的支持。

一般传统数据库，全文检索都实现的很鸡肋，因为一般也没人用数据库存文本字段。进行全文检索需要扫描整个表，如果数据量大的话即使对 SQL 的语法优化，也收效甚微。建立了索引，但是维护起来也很麻烦，对于 insert 和 update 操作都会重新构建索引。基于以上原因可以分析得出，在一些生产环境中，使用常规的搜索方式，性能是非常差的：

* 搜索的数据对象是大量的非结构化的文本数据。
* 文件记录量达到数十万或数百万个甚至更多。
* 支持大量基于交互式文本的查询。
* 需求非常灵活的全文搜索查询。
* 对高度相关的搜索结果的有特殊需求，但是没有可用的关系数据库可以满足。
* 对不同记录类型、非文本数据操作或安全事务处理的需求相对较少的情况。

为了解决结构化数据搜索和非结构化数据搜索性能问题，我们就需要专业，健壮，强大的全文搜索引擎这里说到的全文搜索引擎指的是目前广泛应用的主流搜索引擎。它的工作原理是计算机索引程序通过扫描文章中的每一个词，对每一个词建立一个索引，指明该词在文章中出现的次数和位置，当用户查询时，检索程序就根据事先建立的索引进行查找，并将查找的结果反馈给用户的检索方式。这个过程类似于通过字典中的检索字表查字的过程。

### Elasticsearch 应用案例

* GitHub: 2013 年初，抛弃了 Solr，采取 Elasticsearch 来做 PB 级的搜索。“GitHub 使用Elasticsearch 搜索 20TB 的数据，包括 13 亿文件和 1300 亿行代码”。
* 百度：目前广泛使用 Elasticsearch 作为文本数据分析，采集百度所有服务器上的各类指标数据及用户自定义数据，通过对各种数据进行多维分析展示，辅助定位分析实例异常或业务层面异常。目前覆盖百度内部 20 多个业务线（包括云分析、网盟、预测、文库、直达号、钱包、风控等），单集群最大 100 台机器，200 个 ES 节点，每天导入 30TB+数据。
* 新浪：使用 Elasticsearch 分析处理 32 亿条实时日志。
* 阿里：使用 Elasticsearch 构建日志采集和分析体系。
* Stack Overflow：解决 Bug 问题的网站，全英文，编程人员交流的网站。

## Elasticsearch 入门

### 核心概念

#### 索引（Index）

一个索引就是一个拥有几分相似特征的文档的集合。比如说，你可以有一个客户数据的 索引，另一个产品目录的索引，还有一个订单数据的索引。一个索引由一个名字来标识（必 须全部是小写字母），并且当我们要对这个索引中的文档进行索引、搜索、更新和删除的时 候，都要使用到这个名字。在一个集群中，可以定义任意多的索引。 能搜索的数据必须索引，这样的好处是可以提高查询速度，比如：新华字典前面的目录 就是索引的意思，目录可以提高查询速度。

#### 类型（Type）

在一个索引中，你可以定义一种或多种类型。 一个类型是你的索引的一个逻辑上的分类/分区，其语义完全由你来定。通常，会为具 有一组共同字段的文档定义一个类型。不同的版本，类型发生了不同的变化

|  |  |
| --- | --- |
| 版本 | Type |
| 5.x | 支持多种 type |
| 6.x | 只能有一种 type |
| 7.x | 默认不再支持自定义索引类型（默认类型为：\_doc） |
| 8.x | 将要去掉type |

#### 文档（Document）

一个文档是一个可被索引的基础信息单元，也就是一条数据 比如：你可以拥有某一个客户的文档，某一个产品的一个文档，当然，也可以拥有某个 订单的一个文档。文档以 JSON（Javascript Object Notation）格式来表示，而 JSON 是一个 到处存在的互联网数据交互格式。

在一个 index/type 里面，你可以存储任意多的文档。

#### 字段（Field）

相当于是数据表的字段，对文档数据根据不同属性进行的分类标识。

#### 映射（Mapping）

mapping 是处理数据的方式和规则方面做一些限制，如：某个字段的数据类型、默认值、 分析器、是否被索引等等。这些都是映射里面可以设置的，其它就是处理 ES 里面数据的一 些使用规则设置也叫做映射，按着最优规则处理数据对性能提高很大，因此才需要建立映射， 并且需要思考如何建立映射才能对性能更好。

#### 分片（Shards）

一个索引可以存储超出单个节点硬件限制的大量数据。比如，一个具有 10 亿文档数据 的索引占据 1TB 的磁盘空间，而任一节点都可能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， 每一份就称之为分片。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，它的文档怎样聚合和搜索请求，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，无需过分关心。

#### 副本（Replicas）

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(副本)。

复制分片之所以重要，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副本上并行运行。

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 1 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 1 个主分片和另外 1 个复制分片（1 个完全拷贝），这样的话每个索引总共就有 2 个分片，我们需要根据索引需要确定分片个数。

### Elasticsearch 安装

Elasticsearch 的官方地址：https://www.elastic.co/cn/

我们选择 7.8.0 版本（最新版本半 年前的版本） 下载地址：

<http://dl.elasticsearch.cn/elasticsearch/>