一、后端环境

```
JDK 17.0.9
SpringBoot 3.2.0

IntelliJ IDEA 2023.3.1

ElasticSearch 8.8.1
Kibana 8.8.1（ES与Kibana版本号需相同）
```

在构建 `SpringBoot` 项目时需先启动 `ES` 服务，否则会报错

ElasticSearch下载地址

Kibana下载地址

`ES` 解压后在 `elasticsearch-8.8.1/config/elasticsearch.yml` 中添加一行：`xpack.security.enabled: false` 可关闭较为麻烦的安全认证。

`Kibana` 解压后在 `kibana-8.8.1/config/kibana.yml` 中添加一行：`i18n.locale: "zh-CN"` 可将可视化界面转为中文。

设置完成后分别点击 `elasticsearch-8.8.1/bin/elasticsearch.bat` 和 `kibana-8.8.1/bin/kibana.bat` 以启动服务。`ES` 的默认访问端口为 `localhost:9200`，`Kibana` 的默认访问端口为 `localhost:5601`。

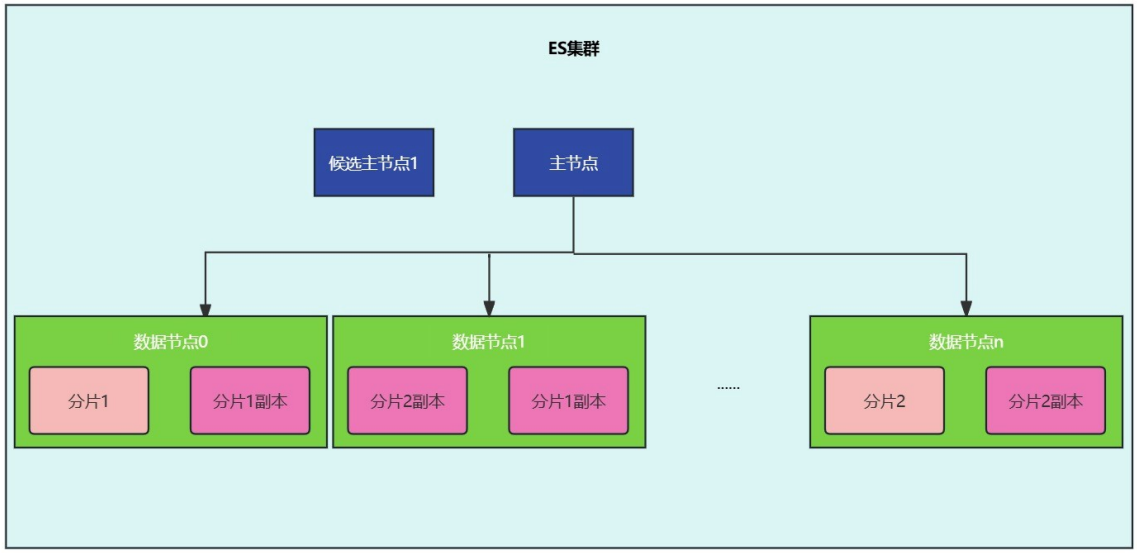
二、ES、Kibana与搜索方法介绍

2.1 Elasticsearch

`ES` 是一个基于 `Lucene` 的、开源的高扩展的分布式搜索引擎，它可以近乎实时地存储和检索数据，且提供了多语言搜索、全文搜索等特性，致力于让开发者以较低的开发成本开发搜索系统。

`ES` 是分布式的，它可将一个索引切分成多个分片存储，每个分片都可以拥有若干副本，以保证在一个分片损坏时数据能够快速恢复。分布式搜索可以降低服务器压力，提高搜索效率。

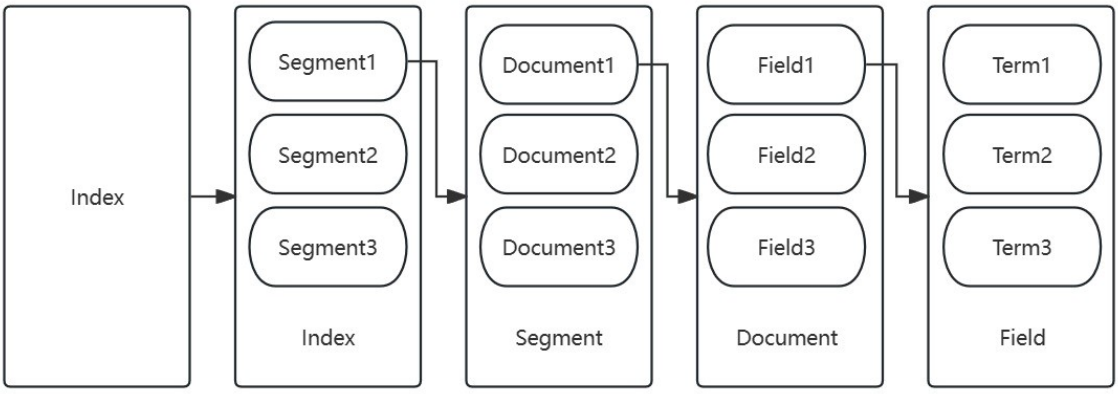
`ES` 的集群中一般有一个主节点和若干数据节点。主节点负责管理索引，如创建和删除；决定分片分配到哪个数据节点；跟踪数据节点的状态等。数据节点储存数据，执行具体的数据操作，如增删改查和数据的聚合等等。为防止主节点失效而出现“群龙无首”的情况，一般会有若干候选主节点，在主节点失效时选举出新的主节点接替工作。



ES集群示例

`Lucene` 是 `Apache` 下 `jakarta` 项目组的子项目，是一个全文搜索引擎架构。它利用文档倒排索引结构来实现在海量数据中做到近乎实时的索引。这样的结构可以使得给出一个词时，能够取得含有这个词的所有文档列表。

`Lucene` 的存储结构分为五级：索引、段、文档，字段，词。



Lucene存储结构

对于存储的文档，`ES` 对其先分词再利用倒排索引存储。`ES` 自带的分词器均为英文设计，在中文文本的表现为分割成单字，所以需使用第三方分词器（如 `IK` 分词器）进行分词。

2.2 Kibana

`Kibana` 是一个基于 `ElasticSearch` 的数据可视化和分析平台。`Kibana` 提供了可视化的控制台以便直接操作数据，其机器学习模块将被用来完成语义搜索。

2.3 关键词搜索和语义搜索

关键词搜索使用 TF-IDF 算法和BM25算法，`BM25` 现为默认算法。语义搜索使用文本向量化后求余弦相似度的方法。

三、代码说明

3.1 Common软件包

处理了跨域请求；定义了发送的 `Result` 格式

3.2 Entity层

主要分为 `Event`（事件）、`EventTheme`（事件专题）、`EventSchema`（事件类型模式）三个实体。`User` 类仅作为连接 `MySQL` 的配置尝试。

表1 Event类

属性	类型	描述
`id`	`String`	事件 `id`
`title`	`String`	事件标题
`description`	`String`	事件详情
`arguments`	`List<String>`	事件论元
`typeId`	`String`	事件类型 `id`
`typeName`	`String`	事件类型名称
`themeId`	`String`	事件专题 `id`
`themeName`	`String`	事件专题名称
`time`	`String`	事件发生时间
`image`	`String`	事件缩略图名

事件缩略图名开始时只要一张图片，后改为支持多张图片，目前以逗号隔开的 `String` 类型传输，可改为 `List<String>`

表2 EventTheme类

属性	类型	描述
`id`	`String`	事件专题 `id`
`themeName`	`String`	事件专题名称

表3 EventSchema类

属性	类型	描述
`id`	`String`	事件类型 `id`
`typeName`	`String`	事件类型名称
`trigger`	`List<String>`	事件触发词
`arguments`	`List<String>`	事件论元

3.3 Controller、Service、Mapper层

`Controller` 接收前端页面请求，`Service` 层对 `Mapper` 层封装从而对高层提供抽象的“服务”。

`EventSchema`、`EventTheme` 各对应一套三层架构。

`Event` 由于存在关键词搜索和语义搜索两种模式，与上面两者不同。对于关键词搜索，使用 `ElasticRepository` 接口，这个接口继承自 `Repository` 接口，能方便地进行增删改查操作。它可以将用户按照一定格式自定义的方法直接转化为特殊查询；详细文档链接。

对于语义搜索，这里使用 `elasticsearch-java` 包中的 `ElasticsearchClient`，即 `ES` 客户端中提供的方法进行查询。文档地址如下

ES官方API地址 | Java官方文档

四、部署时的操作

由于语义搜索的引入，事件在ES中最终的存储方式为基本属性+与模型维数相同的密集向量，如384维的向量存储在 `"text_embedding: predicted_value"` 中，这意味着在文档存储时就需要经过模型的向量化。


```
{
  "_index": "event",
  "_id": "EVnokY8BJ9OxB54nauMO",
  "_score": 11.678823,
  "_source": {
    "image": "",
    "text_embedding": {
      "predicted_value": [...],
      "model_id": "all-minilm-l12-v2"
    },
    "typeName": "边境冲突",
    "description":
      """据乌克兰内务部官网当地时间24日消息，为应对北部邻国白俄罗斯的难民危机，乌执法部门23日启动加强乌白边境管控的联合行动。
乌克兰内务部说，国家边防局、国民警卫队、警察和武装部队将共同参加此次联合执法行动，使用无人机和其他装备对可能发生的非法越境行为实施监控。此外，执法部门还将在乌白边境附近的口岸、高速公路、铁路、公交车站和居民区加强巡逻和安全检查。
乌克兰北部与白俄罗斯接壤，边境长达1000公里左右。
近期，数以千计难民试图从白俄罗斯进入波兰、立陶宛等欧盟国家，最终前往西欧，其中，多数人来自中东地区。波兰等国强化边境管控，大量难民滞留边境。

""",
    "arguments": [
      "'关键词':['']",
      " '时间':['']",
      " '地点':['']",
      " '冲突方':['']"
    ],
    "typeId": "XVBAeo8BDW9AcecZAZHx",
    "time": "2021-11-25",
    "title": "乌克兰加强乌白边境管控应对难民危机"
  }
},
```


事件在ES中的格式

为完成语义搜索需要提前准备好模型，上传模型命令为（需下载安装eland ）

```
eland_import_hub_model --url {url(如localhost:5601)} --hub-model-id {模型文件夹名(路径)} --task-type {task_type}
```

文本向量化的 `task-type` 为 `text_embedding`

目前使用的模型为all-MiniLM-L12-v2 ，是一个384维的轻量级模型。

敲完命令后在Kibana的机器学习模块（需开启试用）便可以导入模型。

首先对事件索引的定义需要发生一些改变，**索引必须有一个用于存储密集向量的字段**，举例如下：

```
PUT test
{
  "mappings": {
    "properties": {
      "text_embedding.predicted_value": {
          "type": "dense_vector",           //knn的要求
          "dims": 384,                      //模型向量维数
          "index": true,
          "similarity": "cosine"            //求相似度的方法
        },
      "description": {                      //文本字段的额外定义
          "type": "text",
          "analyzer": "ik_max_word",        //存储时使用ik_max_word分词方式
          "search_analyzer": "ik_smart"     //查询时使用ik_smart分词方式
      }
    }
  }
}
```

这时我们需要一个**管道**使得模型处理后的向量能直接输入进事件中：

```
PUT _ingest/pipeline/text_embedding                 //管道名称,每个模型只能有一个同名称管道
{
  "description": "Text embedding pipeline",
  "processors": [
    {
      "inference": {
        "model_id": "all-minilm-l12-v2",            //模型名称，唯一
        "target_field": "text_embedding",           //与索引中字段对应
        "field_map": {
          "description": "text_field"               //冒号前内容与需要向量化的字段名称一致
       }
      }
    }
  ]
}
```

此时在插入文档时填写管道字段就能得到含有向量的文档。

另外可以通过重索引方法将没有向量的索引转化为有向量的索引，具体操作为先创建好带有密集向量的新索引，然后输入

```
POST _reindex?wait_for_completion=false
{
  "source": {
    "index": "test_old"             //旧索引
  },
  "dest": {
    "index": "test_new",            //新索引
    "pipeline": "text_embedding"    //指定管道
  }
}
```

五、如何查询

关键词搜索和基础操作可查看此链接

语义搜索可参考以下格式：

```
GET /test/_search
{
    "size": 10,                                     //返回结果数量
    "profile": true,
    "knn": {
      "field": "text_embedding.predicted_value", //与定义时包含密集向量的字段名相同
      "k": 15,
      "num_candidates": 100,                        //近似knn候选者数量
      "query_vector_builder": {
        "text_embedding": {
          "model_id": "all-minilm-l12-v2",
          "model_text": "阿塞拜疆和亚美尼亚"         //关键词
        }
      }
    },
    "_source": [                                    //返回的内容，如只显示description
      "description"
    ]
}
```