云搜索服务

用户指南

文档版本 04

发布日期 2023-06-20

华为技术有限公司

商标声明


和其他华为商标均为华为技术有限公司的商标。
本文档提及的其他所有商标或注册商标，由各自的所有人拥有。

注意

您购买的产品、服务或特性等应受华为公司商业合同和条款的约束，本文档中描述的全部或部分产品、服务或特性可能不在您的购买或使用范围之内。除非合同另有约定，华为公司对本文档内容不做任何明示或暗示的声明或保证。

由于产品版本升级或其他原因，本文档内容会不定期进行更新。除非另有约定，本文档仅作为使用指导，本文档中的所有陈述、信息和建议不构成任何明示或暗示的担保。

目 录

1 产品介绍

1.1 什么是云搜索服务

什么是云搜索服务

云搜索服务（Cloud Search Service，简称CSS）是一个基于Elasticsearch且完全托管的在线分布式搜索服务，为用户提供结构化、非结构化文本、以及基于AI向量的多条件检索、统计、报表。云搜索服务是云ELK生态的一系列软件集合，为您全方位提供托管的ELK生态云服务，兼容Elasticsearch、Kibana、Cerebro等软件。

Elasticsearch是一个开源搜索引擎，可以实现单机和集群部署，并提供托管的分布式搜索引擎服务。在ELK整个生态中，Elasticsearch集群支持结构化、非结构化文本的多条件检索、统计、报表。Elasticsearch搜索引擎相关内容的深入介绍可参见**《Elasticsearch：权威指南》**。

云搜索服务支持自动部署，快速创建Elasticsearch集群，免运维，内置搜索调优实践；拥有完善的监控体系，提供一系列系统、集群以及查询性能等关键指标，让用户更专注于业务逻辑的实现。

产品功能

- 兼容Elasticsearch原生接口

  兼容开源Elasticsearch软件原生接口，支持Beats、Kibana等周边生态。
- 接入多种数据源

  无缝对接Ftp/Obs/Hbase/Kafka等多种数据源，仅需简单配置，无需编程。
- 一键化操作

  一键申请集群、一键扩容、一键重启，从小规模测试到大规模上线，所有主要操作都是一键可达。
- 自定义快照策略

  支持用户触发以及定时触发的快照备份能力，支持恢复到本集群以及其他集群的能力，随时恢复误删数据或者迁移数据到新的搜索集群。

1.2 产品优势

云搜索服务主要有以下特点与显著优势：

高效易用

TB级数据毫秒级返回检索结果，提供可视化平台方便数据展示和分析。

弹性灵活

按需申请，在线扩容，零业务中断，快速应对业务增长。

无忧运维

全托管服务，开箱即用，主要操作一键可达，专业团队贴身看护。

内核增强

提供导入性能增强、存算分离、读写分离、高性能向量检索引擎等高性价比特性。

高可靠性

支持用户手动触发以及定时触发的快照备份，支持恢复到本集群以及其他集群的能力，通过快照恢复支持集群的数据迁移。

- 自动备份（备份快照）

  云搜索服务提供备份功能，可以在控制台的备份恢复界面开启自动备份功能，并根据实际业务需要设置备份周期。

  自动备份是将集群的索引数据进行备份。索引的备份是通过创建集群快照实现，第一次备份时，建议将所有索引数据进行备份。

  云搜索服务支持将ES实例的快照数据保存到对象存储（OBS）服务中，借助OBS的跨region复制功能，可实现数据的跨region备份。

- 恢复数据（恢复快照）

  当数据发生丢失或者想找回某一时间段数据时，可以在“集群快照”界面上单击“恢复”功能，将已有的快照，通过恢复快照功能，将备份的索引数据恢复到指定的集群中，可以快速获得数据。

高安全性

云搜索服务主要从以下几个方面保障数据和业务运行安全：

- 网络隔离

  整个网络划分为2个平面，即业务平面和管理平面。两个平面采用物理隔离的方式进行部署，保证业务、管理各自网络的安全性。

  - 业务平面：主要是集群的网络平面，支持为用户提供业务通道，对外提供数据定义、索引、搜索能力。
  - 管理平面：主要是管理控制台，用于管理云搜索服务。
  - 通过VPC或安全组专有网络来确保主机的安全。

- 访问控制
  - 通过网络访问控制列表（ACL），可以允许或拒绝进入和退出各个子网的网络流量。
  - 内部安全基础设施（包括网络防火墙、入侵检测和防护系统）可以监视通过IPsec VPN连接进入或退出VPC的所有网络流量。
  - 支持用户认证与索引级别鉴权，支持对接第三方管理用户系统。
- 数据安全
  - 在云搜索服务中，通过多副本机制保证用户的数据安全。
  - 支持客户端与服务端通过SSL加密通信。
- 操作审计

  通过云审计服务支持对关键日志与操作进行审计。

高可用性

云搜索服务支持跨可用区部署方案。为了防止数据丢失并在服务中断时最大限度地减少集群停机时间，在创建集群时，可以选择部署在同一个区域中的两个或三个可用区，系统将在选择的可用区之间自动分配节点。当某一可用区出现故障时，剩余的可用区依然可以不间断地提供服务，显著增强了集群的可用性，提升了服务的稳定性。

1.3 产品组件

CSS服务支持Kibana和Cerebro组件。

Kibana

Kibana是一个开源的数据分析与可视化平台，与Elasticsearch搜索引擎一起使用。通过Kibana可以搜索、查看存放在Elasticsearch索引中的数据，也可以实现以图表、地图等方式展示数据。Kibana的官方文档请参见：**https://www.elastic.co/guide/en/kibana/current/index.html**

云搜索服务的Elasticsearch集群默认提供Kibana，无需安装部署，即可一键访问Kibana。云搜索服务兼容了开源Kibana可视化展现和Elasticsearch统计分析能力。

- 支持10余种数据呈现方式
- 支持近20种数据统计方式
- 支持时间、标签等各种维度分类

Cerebro

Cerebro是使用Scala、Play Framework、AngularJS和Bootstrap构建的基于Elasticsearch Web的开源可视化管理工具。通过Cerebro可以对集群进行Web可视化管理，如执行Rest请求、修改Elasticsearch配置、监控实时的磁盘、集群负载、内存使用率等。

云搜索服务的Elasticsearch集群默认提供Cerebro，无需安装部署，即可一键访问Cerebro。云搜索服务完全兼容开源Cerebro，适配最新0.8.4版本。

- 支持Elasticsearch可视化实时负载监控。
- 支持Elasticsearch可视化数据管理。

1.4 应用场景

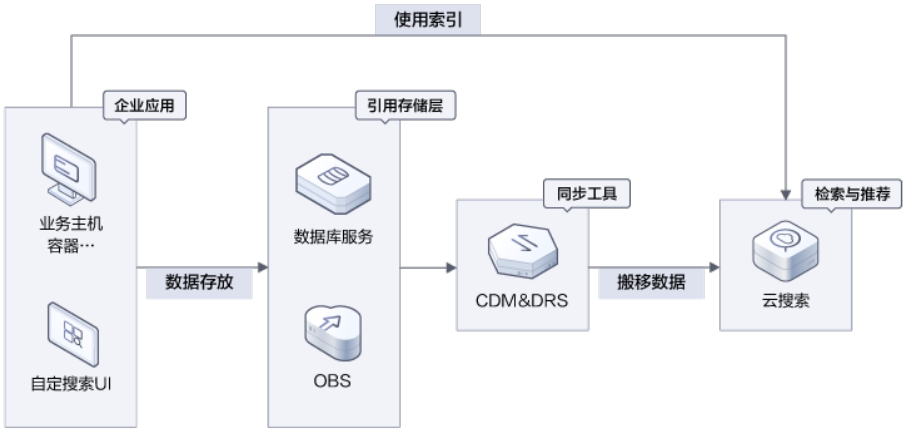
云搜索服务可以帮助网站和APP搭建搜索框，提升用户的搜索体验；也可以用于搭建日志分析平台，助力企业实现数据驱动运维，数据驱动运营；它的向量检索能力可以帮助客户快速构建基于AI的图搜、推荐、语义搜索等丰富的应用。

站内搜索

云搜索服务可用于对网站内容进行关键字检索、对电商网站商品进行检索与推荐。

- 实时检索：站内资料或商品信息更新数秒至数分钟内即可被检索。
- 分类统计：检索同时可以将符合条件的商品进行分类统计。
- 高亮提示：提供高亮能力，页面可自定义高亮显示方式。

图 1-1 站内搜索场景

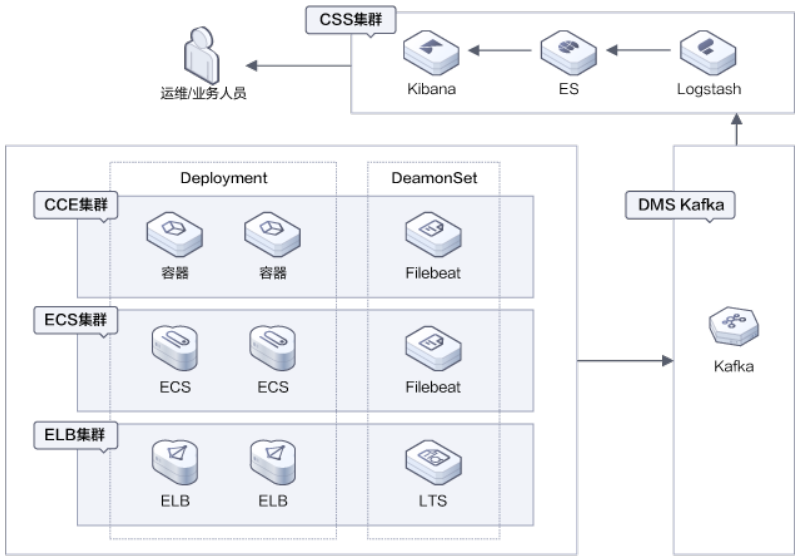


全场景日志分析

云搜索服务可用于全场景日志分析，包括ELB日志、服务器日志、容器和应用日志。其中Kafka作为消息缓冲队列，用于削峰填谷，Logstash负责数据ETL，Elasticsearch负责数据检索与分析，最后由Kibana以可视化的方式呈现给用户。

- 性价比高：采用鲲鹏算力、冷热分离、存算分离，成本同比降低30%+。
- 易用性好：支持丰富的可视化查询语句与拖拽式报表。
- 强大的处理能力：支持每天百TB级数量入库，提供PB级以上数据处理能力。

图 1-2 全场景日志分析场景

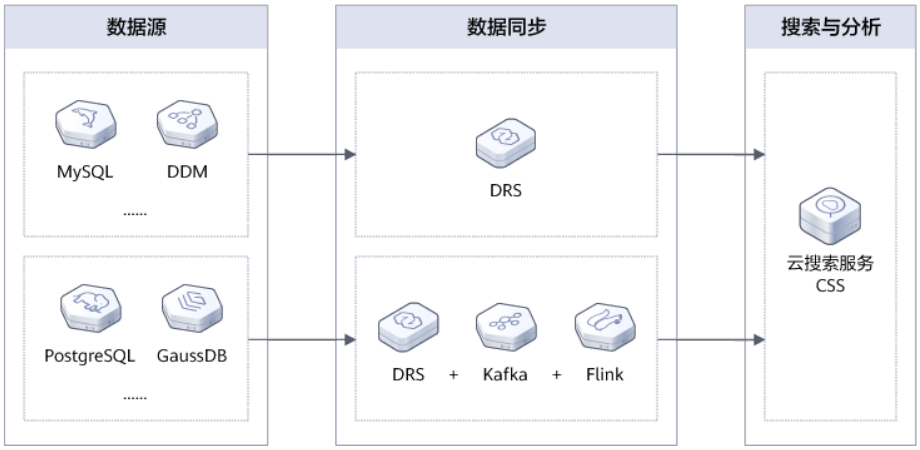


数据库查询加速

云搜索服务可用于加速数据库查询。在电商、物流企业等有订单查询的业务场景，存在数据量大、查询并发高、吞吐大、查询延迟低的要求，关系型数据库具备较好的事务性与原子性，但其TP与AP处理能力较弱，通过将CSS作为备数据库，可提升整个系统的TP与AP处理能力。

- 高性能：支持文本、时间、数字、空间等数据类型；亿级数据查询毫秒级响应。
- 高可扩展性：支持200+数据节点，支持1000+个数据字段。
- 业务“0”中断：规格变更、配置更新采用滚动重启，双副本场景下业务0中断。

图 1-3 数据库查询加速场景

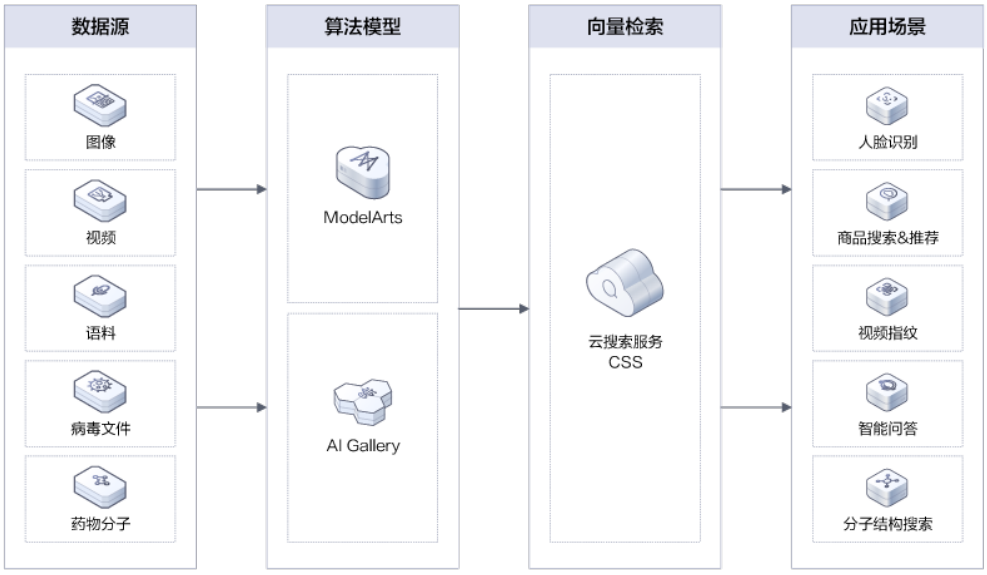


向量检索

云搜索服务支持对图像、视频、语料等非结构化数据提取的特征向量数据进行最近邻或近似近邻检索。

- 高效可靠：云向量检索引擎，提供高效的搜索性能以及分布式容灾能力。
- 索引丰富：支持多种索引算法及相似度度量方式，满足各类应用场景及需求。
- “0”学习成本：完全兼容开源ES语法与生态。

图 1-4 向量检索场景



1.5 约束与限制

集群和节点限制

下表显示了云搜索服务的集群和节点的限制。

表 1-1 Elasticsearch 类型集群和节点限制

集群和节点	限制
每个集群的最大节点数（节点数量）	默认值32，最大支持200个节点，如果需要更改默认值，请联系技术支持。
每个集群的最小节点数（节点数量）	1

浏览器限制

- 访问云搜索服务管理控制台，建议使用如下版本浏览器
  - Google Chrome：36.0及更高版本
  - Mozilla FireFox：35.0及更高版本
- 访问云搜索服务中Kibana和Cerebro，建议使用如下版本浏览器
  - Google Chrome：36.0及更高版本
  - Mozilla FireFox：35.0及更高版本

1.6 配额说明

本服务应用的资源类型如下：

- 实例数
- CPU数量
- 内存数量(GB)
- 磁盘数
- 磁盘容量(GB)

1.7 与其他服务之间的关系

CSS与其他服务的关系如图1-5所示。

图 1-5 CSS 与其他服务的关系

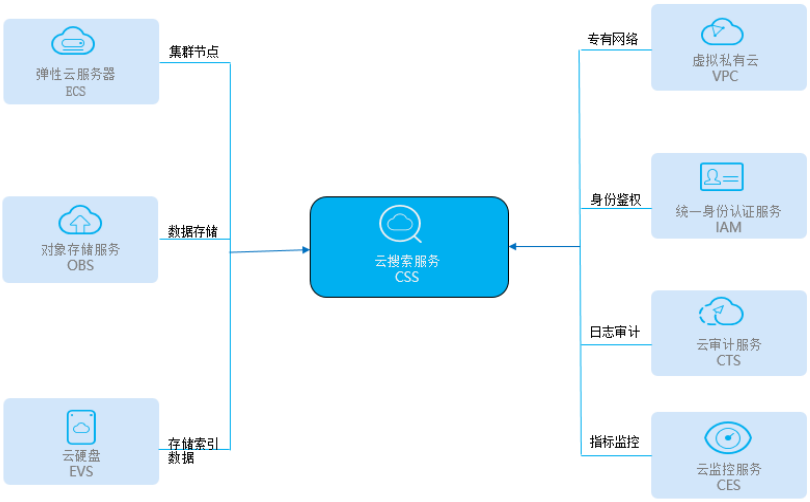


表 1-2 CSS 服务于其他服务的关系

相关服务	交互功能
虚拟私有云（Virtual Private Cloud，简称VPC）	云搜索服务CSS的集群创建在虚拟私有云（VPC）的子网内，VPC通过逻辑方式进行网络隔离，为用户的集群提供安全、隔离的网络环境。
弹性云服务器（Elastic Cloud Server，简称ECS）	云搜索服务CSS的集群中每个节点为一台弹性云服务器（ECS）。创建集群时将自动创建弹性云服务器作为节点。
云硬盘（Elastic Volume Service，简称EVS）	云搜索服务CSS使用云硬盘（EVS）存储索引数据。创建集群时，将自动创建云硬盘用于集群存储。
对象存储服务（Object Storage Service，简称OBS）	云搜索服务CSS的集群快照存储在对象存储服务（OBS）的桶中。
统一身份认证服务（Identity and Access Management，简称IAM）	云搜索服务CSS使用统一身份认证服务（IAM）进行鉴权。
云监控服务（Cloud Eye）	云搜索服务使用云监控服务实时监测集群的指标信息，保障服务正常运行。云搜索服务当前支持的监控指标为磁盘使用率和集群健康状态。用户通过磁盘使用率指标可以及时了解集群的磁盘使用情况。通过集群健康状态指标，用户可以了解集群的健康状态。
云审计服务（Cloud Trace Service，简称CTS）	云审计服务（CTS）可以记录与CSS云搜索服务相关的操作事件，便于日后的查询、审计和回溯。

1.8 基本概念

集群

云搜索服务是以集群为单位进行组织，一个集群代表一个独立运行的搜索服务，由多个节点构成。

索引

用于存储Elasticsearch的数据，是一个或多个分片分组在一起的逻辑空间。

Shard

索引可以存储数据量超过1个节点硬件限制的数据。为满足这样的需求，Elasticsearch提供了一个能力，将一个索引拆分为多个，称为Shard。当您创建一个索引时，您可以根据实际情况指定Shard的数量。每个Shard托管在集群中的任意一个节点中，且每个Shard本身是一个独立的、全功能的“索引”。

Shard的数量只能在创建索引前指定，且在索引创建成功后无法修改。

Replica（副本）

Shard下的实际存储索引的一个副本。可以理解为备份Shard。副本的存在可以预防单节点故障。使用过程中，您可以根据业务情况增加或减少Replica数量。

文档

Elasticsearch存储的实体，是可以被索引的基本单位，相当于关系型数据库中的行。

文档类型

类似关系型数据库中的表，用于区分不同的数据。

Elasticsearch 7.x以下版本中，1个索引里面可以包含若干个文档类型，每个文档必须设定它的文档类型。

Elasticsearch 7.x及以上版本中，文档类型只支持“_doc”。

映射

用来约束字段的类型，可以根据数据自动创建。相当于数据库中的Schema。

字段

组成文档的最小单位。相当于数据库中的Column。

2 快速入门

2.1 快速开始使用 Elasticsearch 搜索引擎

本章节提供了一个简单示例：使用Elasticsearch搜索引擎来为用户提供商品搜索功能。您可以参考此场景示例数据，使用云搜索服务的Elasticsearch搜索引擎搜索数据，基本操作流程如下所示：

- **步骤1：创建集群**
- **步骤2：导入数据**
- **步骤3：搜索数据**
- **步骤4：删除集群**

场景描述

某女装品牌在网上经营电商业务，其以前是使用传统数据库来为用户提供商品搜索功能，但随着用户数量和业务的增长，使用传统数据库的弊端愈来愈明显。主要问题表现为：响应速度慢、准确性低。为了改善用户体验从而避免用户流失，该电商网站开始使用Elasticsearch搜索引擎来为用户提供商品搜索功能，使用了一段时间后，不仅解决了之前使用传统数据库产生的问题，而且实现了用户数量的增长。

本章节将介绍如何使用Elasticsearch搜索引擎为用户提供搜索功能。

假设该电商网站经营商品的数据如下所示：

```
{
"products":[
{"productName":"2017秋装新款文艺衬衫女装","size":"L"}
{"productName":"2017秋装新款文艺衬衫女装","size":"M"}
{"productName":"2017秋装新款文艺衬衫女装","size":"S"}
{"productName":"2018春装新款牛仔裤女装","size":"M"}
{"productName":"2018春装新款牛仔裤女装","size":"S"}
{"productName":"2017春装新款休闲裤女装","size":"L"}
{"productName":"2017春装新款休闲裤女装","size":"S"}
]
}
```

步骤 1：创建集群

在开始搜索数据之前，您需要创建一个集群，其搜索引擎为Elasticsearch。例如，您可以创建一个名称为“Sample-ESCluster”的集群。此集群仅用于入门指导使用，建议

选用“节点规格”为“ess.spec-4u8g”，“节点存储”为“高I/O”，“节点存储容量”为“40GB”。详细操作步骤请参见**创建Elasticsearch类型集群（非安全模式）**。

在开始搜索数据之前，您需要创建一个集群，其搜索引擎为Elasticsearch。例如，您可以创建一个名称为“Sample-ESCluster”的集群。此集群仅用于入门指导使用，建议选用“节点规格”为“ess.spec-4u8g”，“节点存储”为“高I/O”，“节点存储容量”为“40GB”。详细操作步骤请参见**创建Elasticsearch类型集群（安全模式）**或**创建Elasticsearch类型集群（非安全模式）**。

集群创建完成后，在集群列表查看已创建的集群，集群状态为“可用”表示集群创建成功。如下图所示：

图 2-1 创建集群

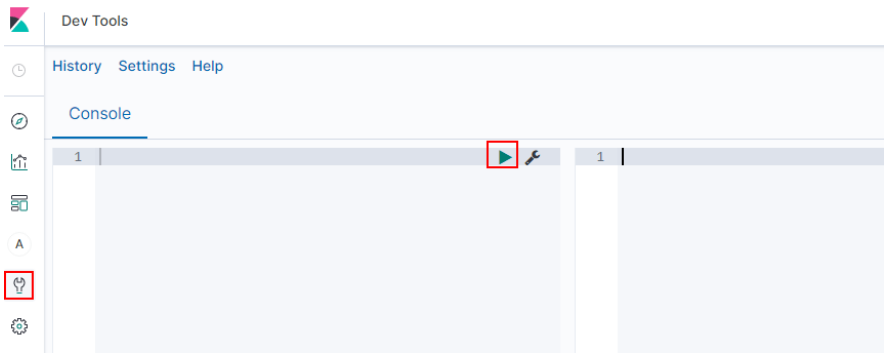
步骤 2：导入数据

云搜索服务支持通过Logstash、Kibana或API将数据导入到Elasticsearch。其中Kibana是Elasticsearch的图形化界面，便于交互验证，因此，这里以Kibana为例介绍将数据导入到Elasticsearch的操作流程。

1. 在“集群管理”页面选择需要登录的集群，单击“操作”列中的“Kibana”进入Kibana登录界面。
   - 非安全模式的集群：将直接进入Kibana操作界面。
   - 安全模式的集群：需要在登录页面输入用户名和密码，单击“Log In”进入Kibana操作界面。用户名默认为admin，密码为创建集群时设置的管理员密码。
2. 在Kibana的左侧导航中选择“Dev Tools”，进入Console界面，如**图2-2**所示。

   Console左侧区域为输入框，输入框右侧的三角形为执行命令按钮；Console右侧为结果输出区域。

   图 2-2 Console 界面

   说明

   不同版本的Kibana界面会有细微差别，请以实际环境为准。
3. 在Console界面，执行如下命令创建索引“my_store”。

   （7.x之后版本）


```
PUT /my_store
{
  "settings": {
    "number_of_shards": 1
  },
  "mappings": {
          "properties": {
        "productName": {
          "type": "text",
          "analyzer": "ik_smart"
        },
        "size": {
          "type": "keyword"
        }
      }
    }
  }
```

返回结果如下所示。

```
{
  "acknowledged" : true,
  "shards_acknowledged" : true,
  "index" : "my_store"
}
```

4. 在Console界面，执行如下命令，将数据导入到“my_store”索引中。

   （7.x之后版本）

```
POST /my_store/_doc/_bulk
{"index":{}}
{"productName":"2017秋装新款文艺衬衫女装","size":"L"}
{"index":{}}
{"productName":"2017秋装新款文艺衬衫女装","size":"M"}
{"index":{}}
{"productName":"2017秋装新款文艺衬衫女装","size":"S"}
{"index":{}}
{"productName":"2018春装新款牛仔裤女装","size":"M"}
{"index":{}}
{"productName":"2018春装新款牛仔裤女装","size":"S"}
{"index":{}}
{"productName":"2017春装新款休闲裤女装","size":"L"}
{"index":{}}
{"productName":"2017春装新款休闲裤女装","size":"S"}
```

   当返回结果信息中“errors”字段的值为“false”时，表示导入数据成功。

步骤 3：搜索数据

- **全文检索**

  假设用户进入该电商网站，她想要查找名称包含“春装牛仔裤”的商品信息，可以搜索“春装牛仔裤”。这里使用Kibana演示用户搜索数据在后台的执行命令和返回结果。

  执行命令如下所示。

  （7.x之后版本）

```
GET /my_store/_search
{
  "query": {"match": {
    "productName": "春装牛仔裤"
  }}
}
```

  返回结果如下所示。

```
{
  "took" : 3,
  "timed_out" : false,
  "_shards" : {
```



```
  "total" : 1,
  "successful" : 1,
  "skipped" : 0,
  "failed" : 0
 },
 "hits" : {
  "total" : {
   "value" : 4,
   "relation" : "eq"
  },
  "max_score" : 1.7965372,
  "hits" : [
   {
    "_index" : "my_store",
    "_type" : "_doc",
    "_id" : "9xf6VHIBfClt6SDjw7H5",
    "_score" : 1.7965372,
    "_source" : {
     "productName" : "2018春装新款牛仔裤女装",
     "size" : "M"
    }
   },
   {
    "_index" : "my_store",
    "_type" : "_doc",
    "_id" : "-Bf6VHIBfClt6SDjw7H5",
    "_score" : 1.7965372,
    "_source" : {
     "productName" : "2018春装新款牛仔裤女装",
     "size" : "S"
    }
   },
   {
    "_index" : "my_store",
    "_type" : "_doc",
    "_id" : "-Rf6VHIBfClt6SDjw7H5",
    "_score" : 0.5945667,
    "_source" : {
     "productName" : "2017春装新款休闲裤女装",
     "size" : "L"
    }
   },
   {
    "_index" : "my_store",
    "_type" : "_doc",
    "_id" : "-hf6VHIBfClt6SDjw7H5",
    "_score" : 0.5945667,
    "_source" : {
     "productName" : "2017春装新款休闲裤女装",
     "size" : "S"
    }
   }
  ]
 }
}
```


  - Elasticsearch支持分词，上面执行命令会将“春装牛仔裤”分词为“春装”和“牛仔裤”。
  - Elasticsearch支持全文检索，上面执行命令会在所有商品信息中搜索包含“春装”或“牛仔裤”的商品信息。
  - Elasticsearch与传统数据库不同，它能借助倒排索引在毫秒级返回结果。
  - Elasticsearch支持评分排序，在上面返回结果中，前两条商品信息中同时出现了“春装”和“牛仔裤”，后两条商品信息中只出现了“春装”，所以前两条比后两条与检索关键词的匹配度更高，分数更高，排序也更靠前。

- **聚合结果显示**

该电商网站可以提供聚合结果显示功能，例如： 对“春装”对应的产品按照尺码分类，统计不同尺码的数量。这里使用Kibana演示聚合结果显示功能在后台的执行命令和返回结果。

执行命令如下所示。

（7.x之后版本）

```
GET /my_store/_search
{
"query": {
"match": { "productName": "春装" }
},
"size": 0,
"aggs": {
"sizes": {
"terms": { "field": "size" }
}
}
}
```

返回结果如下所示。

（7.x之后版本 ）

```
{
  "took" : 3,
  "timed_out" : false,
  "_shards" : {
    "total" : 1,
    "successful" : 1,
    "skipped" : 0,
    "failed" : 0
  },
  "hits" : {
    "total" : {
      "value" : 4,
      "relation" : "eq"
    },
    "max_score" : null,
    "hits" : [ ]
  },
  "aggregations" : {
    "sizes" : {
      "doc_count_error_upper_bound" : 0,
      "sum_other_doc_count" : 0,
      "buckets" : [
        {
          "key" : "S",
          "doc_count" : 2
        },
        {
          "key" : "L",
          "doc_count" : 1
        },
        {
          "key" : "M",
          "doc_count" : 1
        }
      ]
    }
  }
}
```

步骤 4：删除集群

当您已完全了解Elasticsearch搜索引擎的使用流程和方法后，您可以参考如下步骤，删除示例集群以及示例数据，避免造成资源浪费。

说明

由于集群删除后，数据无法恢复，请谨慎操作。

1. 登录云搜索服务管理控制台。在左侧菜单栏选择“集群管理 > Elasticsearch”。
2. 进入集群管理页面，选中“Sample-ESCluster”集群所在行，在操作列单击“更多”>“删除”。
3. 在弹出的确认对话框中，确认要删除的集群名称，单击“确定”完成操作。

3 权限管理

3.1 创建用户并授权使用 CSS

本章节介绍创建CSS用户操作，将CSS服务的策略授予用户组，并将用户添加至用户组中（一个用户组下面的用户具有相同的权限），从而使用户拥有对应的CSS权限，操作流程如**图1 给用户授权CSS权限流程**所示。

CSS具有两种类型用户权限（CSS管理员权限和只读权限），在权限规划的时候请规划这两种类型的用户组。

前提条件

给用户组授权之前，请您了解用户组可以添加的CSS系统策略，请参见权限管理。

示例流程

图 3-1 给用户授权 CSS 权限流程

1. 创建用户组并授权
   在IAM控制台创建用户组，并授予云搜索服务权限。
2. 创建用户并加入用户组
   在IAM控制台创建用户，并将其加入**1.创建用户组并授权**中创建的用户组。
3. 用户登录并验证权限
   新创建的用户登录控制台，验证云搜索服务的权限。

4 创建并接入集群

4.1 部署跨 AZ 集群

为了防止数据丢失，或者在服务中断时能够最大限度地减少集群停机时间，CSS服务支持集群选择跨AZ部署，提升集群的高可用能力。在创建集群时选择同一个区域中的两个或三个可用区，系统自动在选择的可用区之间分配节点。

选择节点数

当创建集群时，选择了两个或者三个可用区时，CSS服务将自动为开启跨AZ高可用特性，节点将会被均衡的分布在不同的AZ，不同节点数量的AZ分布情况可以参见**表4-1**。

说明

- 创建集群时，选择的节点数量要大于等于AZ数量，否则不支持跨AZ部署。
- 部署跨AZ集群时，如果选择了“启用Master节点”，Master节点也会被均匀的分布在不同的AZ上。
- 系统分配的节点，满足各个AZ之间节点数量差小于等于1。

表 4-1 节点数量和 AZ 分布

集群节点个数	单AZ	两AZ		三AZ		
	AZ1	AZ1	AZ2	AZ1	AZ2	AZ3
1个节点	1	不支持		不支持		
2个节点	2	1	1	不支持		
3个节点	3	2	1	1	1	1
4个节点	4	2	2	2	1	1
…	…	…	…	…	…	…

设置副本

设置副本能最大程度的利用AZ的高可用能力。

- 在跨两个可用区的部署中，当其中一个AZ不可用时，剩下的AZ需要继续提供服务，因此索引的副本个数至少为1个。由于Elasticsearch默认副本数为1个，因此如果您对读性能没有特殊要求，可以直接使用默认值。
- 在跨三个可用区部署中，为了保证其中任意一个AZ不可用时，剩余的AZ需要继续提供服务，因此索引的副本数至少要为1个。为了提高集群的查询能力，也可以设置更多的副本。由于Elasticsearch默认的副本数为1个，因此需要用户修改setting配置来实现修改索引副本个数。

  可以通过如下命令修改索引的副本个数，如：

  curl -XPUT http://ip:9200/{index_name}/_settings -d '{"number_of_replicas":2}'

  也可以通过在模板中指定所有索引的副本个数，如：

  curl -XPUT http://ip:9200/ _template/templatename -d '{ "template": "*", "settings": {"number_of_replicas": 2}}'

  说明

  - ip：内网访问地址。
  - index_name：索引名称。
  - number_of_replicas：修改后的索引副本个数。命令中的取值表示修改为2个索引副本。

可用区中断的行为分析

当创建集群时，选择两个或三个AZ，如果一个AZ故障，业务故障行为分析如表1所示。

表 4-2 AZ 故障的业务故障行为分析

选择的AZ数量	开启主节点个数	业务中断行为
2	0	• 如果节点个数为2的倍数： – 一半的数据节点故障，需要替换故障可用区中的一个节点，才能继续选择主节点。 • 如果节点数为奇数： – 故障AZ含多一个节点，需要替换故障可用区中一个节点，才能继续选择主节点。相关替换请联系技术支持。 – 故障AZ含少一个节点，不中断业务，能够继续选主。

选择的AZ数量	开启主节点个数	业务中断行为
2	3	有50%机会的停机时间。当两个专用主节点分配到一个可用区中，一个主节点分配到另一个可用区中时： • 如果具有一个专用主节点的可用区遇到中断，则剩余可用区具有两个专用主节点，这两个专用主节点可以选择出主节点。 • 如果具有两个专用主节点的可用区遇到中断，剩余可用区只有一个专用主节点，无法选择出主节点，业务中断，需要联系技术支持。
3	0	当您选择3个可用区，节点个数为4，三个可用区的节点分布数为2，1，1，如果节点个数为2的可用区故障，那么此时业务中断，建议您选择三个可用区时避免选择4个节点。 一般不会出现业务中断时间。
3	3	无业务中断时间。

4.2 安全集群简介

CSS服务在创建Elasticsearch集群时，支持创建安全模式的集群，当集群开启安全模式后，访问集群时需要进行安全认证，且支持对集群进行授权、加密等功能。

背景信息

CSS服务支持创建多种安全模式的集群，不同安全模式的差异请参见表4-3。

表 4-3 集群安全模式对比

集群安全模式	适用场景	优点	缺点
非安全模式	适合内网业务，用于测试场景。	简单，接入集群容易。	安全性差，谁都可以访问集群。
安全模式+HTTP协议	可以实现用户权限隔离，适用于对集群性能敏感的场景。	访问集群需要安全认证，提升了集群安全性，且通过HTTP协议访问集群能保留集群的高性能。	无法公网访问集群。
安全模式+HTTPS协议	有非常高的安全要求，且需要公网访问集群的场景。	访问集群需要安全认证，提升了集群安全性，且HTTPS协议的通讯加密可以实现集群公网访问功能。	通过HTTPS协议访问集群，集群性能相对HTTP协议来说，会下降20%左右。

安全认证

当访问安全模式的集群时，需要输入用户名和密码才能访问。支持以下两类用户的安全认证：

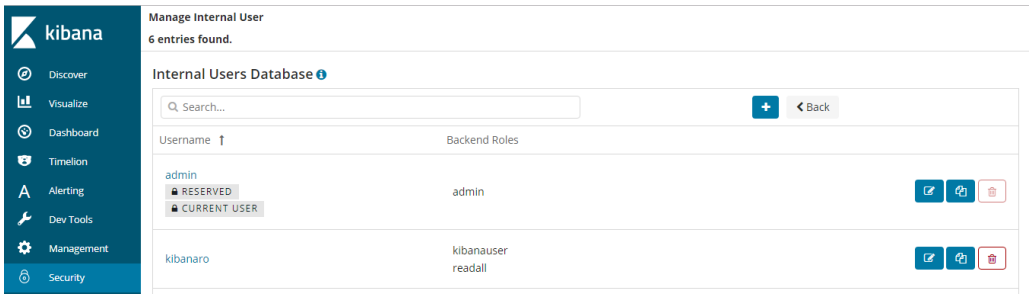
- 集群的管理员：管理员帐户名默认为**admin**，密码为创建集群时设置的管理员密码。
- 集群的User：通过kibana创建的集群用户和密码。

授权

在kibana使用界面您可以在Security菜单中控制用户在ES集群中的权限，并且可以针对集群、索引、文档和字段四个级别进行分层权限设置。详细操作请参见**使用Kibana创建用户并授权**。

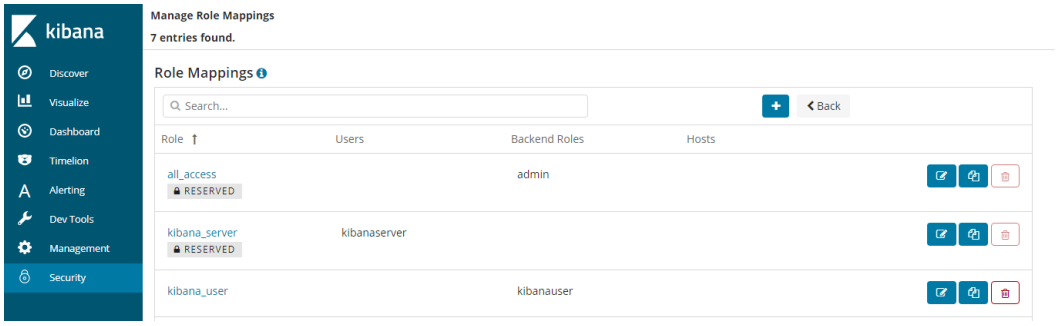
您可以增删用户，并将用户映射到角色类型设置权限。

图 4-1 用户设置



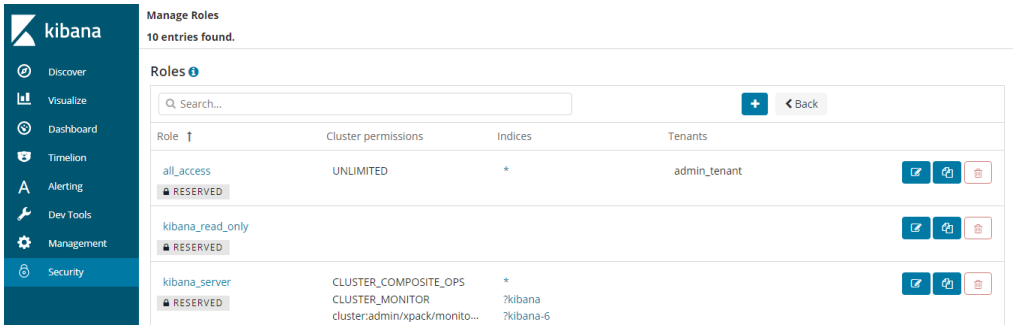
可以使用角色映射配置角色成员，可使用用户名、后端角色和主机名将用户分配给角色。

图 4-2 角色映射



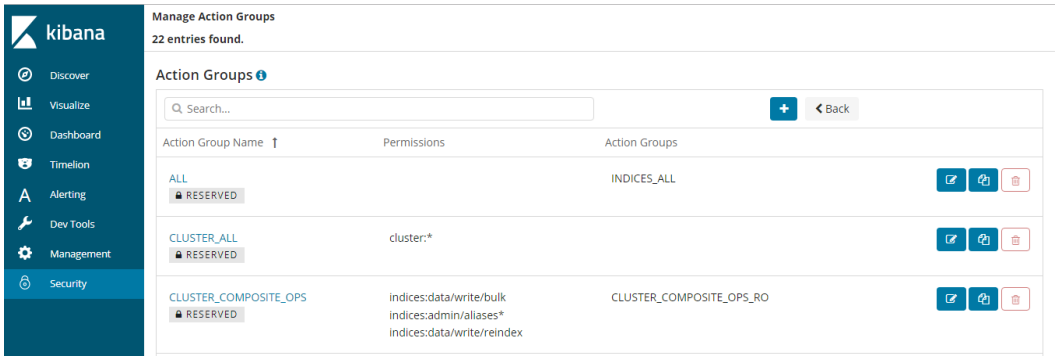
可以设置每种角色的集群访问权限、索引和文档访问权限以及kibana租户。

图 4-3 角色权限设置



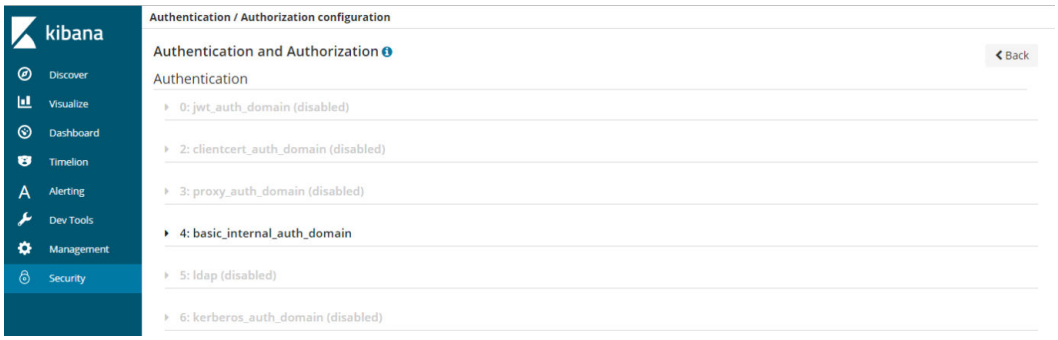
可以设置操作组，并将操作组分配给角色配置角色对索引和文档类型的访问权限。

图 4-4 操作组设置



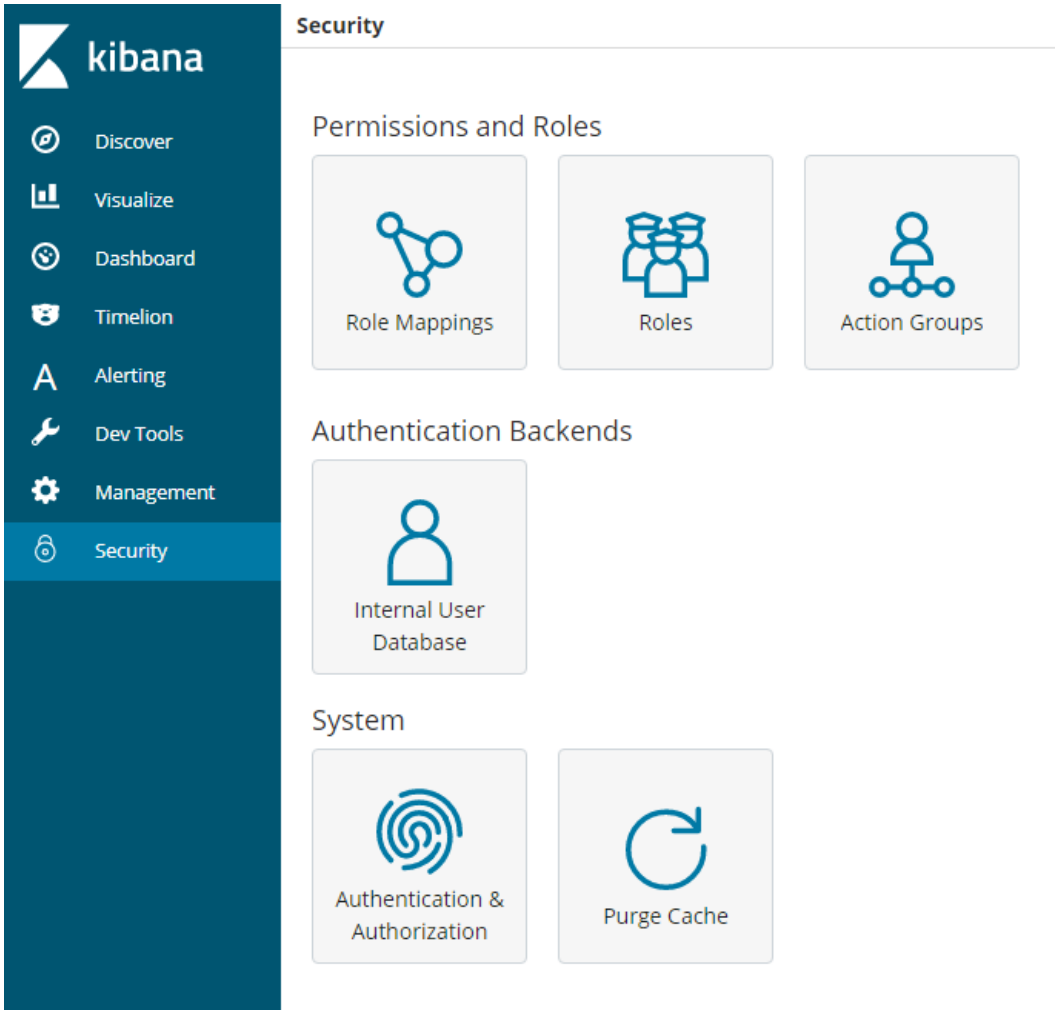
可以查询集群当前设置的身份验证及授权模块的参数。使用securityadmin命令行可修改相关配置。

图 4-5 集群参数查看



最后，安全模块还为您提供了清除所有安全缓存的功能。

图 4-6 安全缓存清除



加密

当您使用节点对节点传输或者HTTP传输方式传输关键数据时，可以借助SSL/TLS加密，对数据安全进行保护。

以上功能除了可以使用Kibana可视化界面操作，还可以使用.yml文件（不推荐）和REST API操作，更多安全模式相关内容可以查看**安全模式官方介绍**。

重置管理员密码

当您想要更换安全模式集群的管理员密码，或者忘记管理员密码时，可以对密码进行重置。

1. 在集群管理列表，选择需要重置密码的集群，单击集群名称，进入集群基本信息页面。
2. 在“配置信息”区域，单击“重置密码”后的“重置”，设置并确认新的管理员密码。

说明

- 可输入的字符串长度为8~32个字符。
- 密码至少包含大写字母、小写字母、数字和特殊字符四类中的三类。其中支持的特殊字符有：~!@#$%^&*()-_=+\|[{}];:,<.>/?
- 不能与管理员帐户名或倒序的管理员帐户名相同。
- 建议定期修改密码。

图 4-7 重置密码

4.3 创建 Elasticsearch 类型集群（安全模式）

Elasticsearch集群支持开启安全模式。参考本章可以完成安全模式的Elasticsearch集群的创建。

说明

- 公网访问和Kibana公网访问需要开启安全模式才能使用。

背景信息

- 新建集群时，当设置不同节点类型时支持的节点数量区间会有区别，具体情况请参考表4-4。

表 4-4 不同节点类型的节点数量说明

集群包含的节点类型	节点数量的取值范围
ess	ess：1~32

集群包含的节点类型	节点数量的取值范围
ess、ess-master	ess：1~200 ess-master：3~9的奇数
ess、ess-client	ess：1~32 ess-client：1~32
ess、ess-cold	ess：1~32 ess-cold：1~32
ess、ess-master、ess-client	ess：1~200 ess-master：3~9的奇数 ess-client：1~32
ess、ess-master、ess-cold	ess：1~200 ess-master：3~9的奇数 ess-cold：1~32
ess、ess-client、ess-cold	ess：1~32 ess-client：1~32 ess-cold：1~32
ess、ess-master、ess-client、ess-cold	ess：1~200 ess-master：3~9的奇数 ess-client：1~32 ess-cold：1~32
四种节点类型的说明： ● ess：默认节点类型，即创建集群时必选的数据节点类型，其他3种节点类型都是基于业务需要可选的类型。 ● ess-master：Master节点 ● ess-client：Client节点 ● ess-cold：冷数据节点	

操作步骤

1. 登录云搜索服务管理控制台。
2. 单击右上角的“创建集群”，进入“创建集群”页面。
3. 选择“当前区域”和“可用区”。

表 4-5 区域和可用区参数说明

参数	说明
当前区域	集群工作区域在右侧下拉框中选择。

参数	说明
可用区	选择集群工作区域下关联的可用区。 云搜索服务最多支持配置3个“可用区”，详细请参考**部署跨AZ集群**。

4. 配置集群基本信息。

表 4-6 基本参数说明

参数	说明
集群版本	选择所需的集群版本，支持的版本以界面可选项为准。
集群名称	自定义集群名称，可输入的字符范围为4～32个字符，只能包含数字、字母、中划线和下划线，且必须以字母开头。 **说明** 当集群创建成功后，您可以根据需求修改集群名称。单击需要修改的集群名称，进入集群基本信息页面，单击“集群名称”后面的  ，修改完成后，单击  ，进行保存。如果需要取消修改，可单击  进行取消。

5. 配置集群的规格信息。

表 4-7 规格参数说明

参数	说明
节点数量	集群中的节点个数。 • 如果未启用Master节点和Client节点时，此参数指定的节点将被作为Master节点和Client节点，同时具备集群管理、存储数据、提供接入集群和分析数据的服务。此时，为保证集群中数据的稳定性，建议设置节点数量大于等于3个。 • 如果启用Master节点，且未启用Client节点，此参数指定的节点将用于存储数据并提供Client节点功能。 • 如果已启用Master节点和Client节点，此参数指定的节点将仅用于存储数据。 • 如果启用Client节点，且未启用Master节点，此参数指定的节点将用于存储数据并提供Master节点功能。
CPU架构	目前支持“X86计算”和“鲲鹏计算”两种类型。具体支持的类型由实际区域环境决定。
节点规格	集群中的节点规格。您可以根据需求，选择对应的规格。每个集群只能选择一个规格。
节点存储	当前支持三种存储类型，普通I/O、高I/O、超高I/O。

参数	说明
节点存储容量	存储空间大小，其取值范围与节点规格关联，不同的规格允许的取值范围不同。 节点存储容量只支持配置为20的倍数。
启用Master节点	Master节点用于管理集群中的所有节点。当需要存储和分析的数据量大，所需节点数量大于20个节点时，建议启用Master节点，保证集群的稳定性。反之，建议仅设置集群的“节点数量”参数，同时作为Master和Client节点即可。 启用Master节点后，在下方选择对应的“节点规格”、“节点数量”和“节点存储”。“节点数量”必须是不小于3的奇数，最多设置9个节点。“节点存储”的存储容量为固定值，存储类型可以根据实际情况选择。
启用Client节点	Client节点用于提供客户端接入集群和分析数据的服务。当需要存储和分析的数据量大，所需节点数量大于20个节点时，建议启用Client节点，保证集群的稳定性。反之，建议仅设置集群的“节点数量”参数，同时作为Master和Client节点即可。 启用Client节点后，在下方选择对应的“节点规格”、“节点数量”和“节点存储”。“节点数量”可设置为1~32任意数值。“节点存储”的存储容量为固定值，存储类型可以根据实际情况选择。
启用冷数据节点	冷数据节点用于存放对于历史数据要求分钟级别的返回。当用户对历史数据返回时间要求不是很高的话，可以将这部分数据存储在冷数据节点上，从而降低成本。 启用冷数据节点后，在下方选择对应的“节点规格”、“节点数量”和“节点存储”。“节点数量”可设置为1~32任意数值。“节点存储”的存储类型和存储容量可以根据实际情况选择。 开启冷数据节点之后，云搜索服务将会自动的给相关节点打上冷热标签。

6. 设置集群的企业项目。

   如果您开通了“企业项目”，在创建集群时，可以给集群绑定一个企业项目。您可以在右侧下拉框中选择当前用户下已创建的企业项目，也可以通过单击“查看项目管理”按钮，前往“企业项目管理”管理控制台，新建企业项目和查看已有的企业项目。

7. 单击“下一步：网络配置”，设置集群的网络配置。

表 4-8 网络配置参数说明

参数	说明
虚拟私有云	VPC即虚拟私有云，是通过逻辑方式进行网络隔离，提供安全、隔离的网络环境。 选择创建集群需要的VPC，单击“查看虚拟私有云”进入VPC服务查看已创建的VPC名称和ID。如果没有VPC，需要创建一个新的VPC。 **说明** 此处您选择的VPC必须包含网段（CIDR），否则集群将无法创建成功。新建的VPC默认包含网段（CIDR）。
子网	通过子网提供与其他网络隔离的、可以独享的网络资源，以提高网络安全。 选择创建集群需要的子网，可进入VPC服务查看VPC下已创建的子网名称和ID。
安全组	安全组是一个逻辑上的分组，为同一个VPC内具有相同安全保护需求并相互信任的弹性云服务器提供访问策略。单击“查看安全组”可了解安全组详情。 **说明** • 为了确保您能够正常访问集群，需要放通安全组9200规则。 • 如果创建的集群为7.6.2及以上版本，则需要确保同安全组内节点之间的端口全放通。如果无法放通同安全组内节点之间的全部端口，请至少确保9300端口的通信。
安全模式	集群支持选择是否开启安全模式，开启之后将对集群进行通讯加密和安全认证。 • **管理员帐户名**默认为admin。 • 设置并确认**管理员密码**。要记住设置的密码，后续访问集群需要输入密码。
HTTPS访问	只有开启集群的安全模式才可以启用HTTPS访问，开启HTTPS访问后，访问集群将进行通讯加密。 **说明** 安全集群使用HTTPS通信，相比非安全集群使用HTTP通信在读取性能上会慢很多。如果想要读取性能快，又想要使用安全集群所提供的用户权限隔离资源（索引、文档、字段等）的功能，则可以关闭HTTPS访问。关闭HTTPS访问后，会使用HTTP协议与集群通信，无法保证数据安全性，并且无法开启公网访问功能。
公网访问	开启“HTTPS访问”后，可以选择是否配置“公网访问”，配置公网访问后，用户可以获得一个公网访问的IP，通过这个IP可以在公网上访问该安全集群，详细配置请参考**公网访问**。

8. 单击“下一步：高级配置”，设置集群自动快照和其他高级功能。
   a. 设置集群自动快照开关、基础配置和快照配置。
      系统默认打开集群快照开关，如果您不需要启用自动快照，可以在“集群快照开关”右侧关闭。自动快照会创建委托访问对象存储服务OBS，快照存储在标准存储中需额外计费。

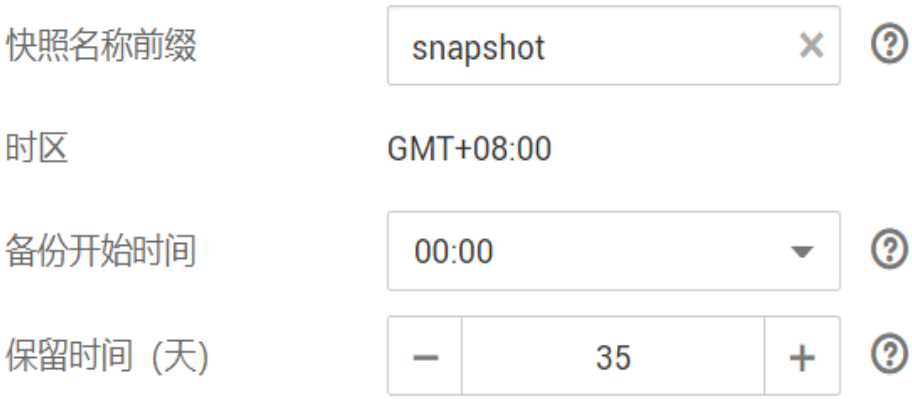
表 4-9 集群快照基础配置的参数说明

参数	说明
OBS桶	在下拉框中选择存储快照的OBS桶。也可以单击右侧的“创建桶”新建OBS。 创建或者已存在的OBS桶需满足如下条件： ● “存储类别”为“标准存储”。
备份路径	快照在OBS桶中的存放路径。 备份路径配置规则： ● 备份路径不能包括下列符号：\:*?"<>| ● 备份路径不能以“/”开头。 ● 备份路径不能以“.”开头或结尾。 ● 备份路径的总长度不能超过1023个字符。
IAM委托	指当前帐号授权云搜索服务访问或维护存储在OBS中数据。也可以单击右侧的“创建委托”新建IAM委托。 创建或者已存在的IAM委托需满足如下条件： ● “委托类型”选择“云服务”。 ● “云服务”选择“Elasticsearch”或者“云搜索服务 CSS”。 ● 设置当前委托具备“全局服务”中“对象存储服务”项目的“Tenant Administrator”权限。

表 4-10 集群快照自动创建快照的参数说明

参数	说明
快照名称前缀	快照名称前缀的长度为1～32个字符，只能包含小写字母、数字、中划线和下划线，且必须以小写字母开头。快照名称由快照名称前缀加上时间戳组成，例如自动生成的快照名称为“snapshot-1566921603720”。
时区	指备份时间对应的时区，不支持修改。基于此时区选择备份开始时间。
备份开始时间	指每天自动开始备份的时间，只能指定整点时间，如00:00、01:00，取值范围为00:00～23:00。请在下拉框中选择时间。
保留时间（天）	指备份的快照在OBS的保留时间，以“天”为单位，取值范围为1～90，您可以根据实际需求进行设置。系统在半点时刻会自动删除超过保留时间的快照。

图 4-8 设置自动创建快照的参数



b. 配置集群高级功能。

- **默认配置**：默认关闭“终端节点服务”、“Kibana公网访问”和“标签”功能，在集群创建完成后，若有需要也可以人工启用这些功能。
- **自定义**：根据需要选择开启“终端节点服务”、“Kibana公网访问”和“标签”功能。

表 4-11 高级配置参数

参数	说明
终端节点服务	开启终端节点服务后，用户可以获得一个内网访问的域名，通过这个域名，可以在同一个vpc内访问该集群。详细配置请参考**终端节点服务**。
Kibana公网访问	只有开启“安全模式”的集群，才能配置Kibana公网访问。开启Kibana公网访问后，用户可以获得一个Kibana公网访问地址，通过这个地址，可以在公网上面访问该集群。详细配置请参考**Kibana公网访问**。
标签	为集群添加标签，可以方便用户识别和管理拥有的集群资源。此处您可以选择“标签管理服务”中已定义好的“预定义标签”，也可以自己定义标签。详细标签使用请参考**标签管理**。

9. 单击“下一步：确认配置”，确认完成后单击“立即创建”开始创建集群。
10. 单击“返回集群列表”，系统将跳转到“集群管理”页面。您创建的集群将展现在集群列表中，且集群状态为“创建中”，创建成功后集群状态会变为“可用”。

    如果集群创建失败，请根据界面提示，重新创建集群。

4.4 创建 Elasticsearch 类型集群（非安全模式）

参考本章节可以完成非安全模式的Elasticsearch集群的创建。

背景信息

- 新建集群时，当设置不同节点类型时支持的节点数量区间会有区别，具体情况请参考表4-12。

表 4-12 不同节点类型的节点数量说明

集群包含的节点类型	节点数量的取值范围
ess	ess：1~32
ess、ess-master	ess：1~200 ess-master：3~9的奇数
ess、ess-client	ess：1~32 ess-client：1~32
ess、ess-cold	ess：1~32 ess-cold：1~32
ess、ess-master、ess-client	ess：1~200 ess-master：3~9的奇数 ess-client：1~32
ess、ess-master、ess-cold	ess：1~200 ess-master：3~9的奇数 ess-cold：1~32
ess、ess-client、ess-cold	ess：1~32 ess-client：1~32 ess-cold：1~32
ess、ess-master、ess-client、ess-cold	ess：1~200 ess-master：3~9的奇数 ess-client：1~32 ess-cold：1~32
四种节点类型的说明： • ess：默认节点类型，即创建集群时必选的数据节点类型，其他3种节点类型都是基于业务需要可选的类型。 • ess-master：Master节点 • ess-client：Client节点 • ess-cold：冷数据节点	

操作步骤

1. 登录云搜索服务管理控制台。
2. 单击右上角的“创建集群”，进入“创建集群”页面。

3. 选择“当前区域”和“可用区”。

表 4-13 区域和可用区参数说明

参数	说明
当前区域	集群工作区域在右侧下拉框中选择。
可用区	选择集群工作区域下关联的可用区。 云搜索服务最多支持配置3个“可用区”，详细请参考**部署跨AZ集群**。

4. 配置集群基本信息。

表 4-14 基本参数说明

参数	说明
集群版本	选择所需的集群版本，支持的版本以界面可选项为准。
集群名称	自定义集群名称，可输入的字符范围为4～32个字符，只能包含数字、字母、中划线和下划线，且必须以字母开头。 **说明** 当集群创建成功后，您可以根据需求修改集群名称。单击需要修改的集群名称，进入集群基本信息页面，单击“集群名称”后面的  ，修改完成后，单击，进行保存。如果需要取消修改，可单击  进行取消。

5. 配置集群的规格信息。

表 4-15 规格参数说明

参数	说明
节点数量	集群中的节点个数。 • 如果未启用Master节点和Client节点时，此参数指定的节点将被作为Master节点和Client节点，同时具备集群管理、存储数据、提供接入集群和分析数据的服务。此时，为保证集群中数据的稳定性，建议设置节点数量大于等于3个。 • 如果启用Master节点，且未启用Client节点，此参数指定的节点将用于存储数据并提供Client节点功能。 • 如果已启用Master节点和Client节点，此参数指定的节点将仅用于存储数据。 • 如果启用Client节点，且未启用Master节点，此参数指定的节点将用于存储数据并提供Master节点功能。
CPU架构	目前支持“X86计算”和“鲲鹏计算”两种类型。具体支持的类型由实际区域环境决定。

参数	说明
节点规格	集群中的节点规格。您可以根据需求，选择对应的规格。每个集群只能选择一个规格。
节点存储	当前支持三种存储类型，普通I/O、高I/O、超高I/O。
节点存储容量	存储空间大小，其取值范围与节点规格关联，不同的规格允许的取值范围不同。 节点存储容量只支持配置为20的倍数。
启用Master节点	Master节点用于管理集群中的所有节点。当需要存储和分析的数据量大，所需节点数量大于20个节点时，建议启用Master节点，保证集群的稳定性。反之，建议仅设置集群的“节点数量”参数，同时作为Master和Client节点即可。 启用Master节点后，在下方选择对应的“节点规格”、“节点数量”和“节点存储”。“节点数量”必须是不小于3的奇数，最多设置9个节点。“节点存储”的存储容量为固定值，存储类型可以根据实际情况选择。
启用Client节点	Client节点用于提供客户端接入集群和分析数据的服务。当需要存储和分析的数据量大，所需节点数量大于20个节点时，建议启用Client节点，保证集群的稳定性。反之，建议仅设置集群的“节点数量”参数，同时作为Master和Client节点即可。 启用Client节点后，在下方选择对应的“节点规格”、“节点数量”和“节点存储”。“节点数量”可设置为1~32任意数值。“节点存储”的存储容量为固定值，存储类型可以根据实际情况选择。
启用冷数据节点	冷数据节点用于存放对于历史数据要求分钟级别的返回。当用户对历史数据返回时间要求不是很高的话，可以将这部分数据存储在冷数据节点上，从而降低成本。 启用冷数据节点后，在下方选择对应的“节点规格”、“节点数量”和“节点存储”。“节点数量”可设置为1~32任意数值。“节点存储”的存储类型和存储容量可以根据实际情况选择。 开启冷数据节点之后，云搜索服务将会自动的给相关节点打上冷热标签。

6. 设置集群的企业项目。

   如果您开通了“企业项目”，在创建集群时，可以给集群绑定一个企业项目。您可以在右侧下拉框中选择当前用户下已创建的企业项目，也可以通过单击“查看项目管理”按钮，前往“企业项目管理”管理控制台，新建企业项目和查看已有的企业项目。

7. 指定集群的网络规格相关参数。

表 4-16 参数说明

参数	说明
虚拟私有云	VPC即虚拟私有云，是通过逻辑方式进行网络隔离，提供安全、隔离的网络环境。 选择创建集群需要的VPC，单击“查看虚拟私有云”进入VPC服务查看已创建的VPC名称和ID。如果没有VPC，需要创建一个新的VPC。 **说明** 此处您选择的VPC必须包含网段（CIDR），否则集群将无法创建成功。新建的VPC默认包含网段（CIDR）。
子网	通过子网提供与其他网络隔离的、可以独享的网络资源，以提高网络安全。 选择创建集群需要的子网，可进入VPC服务查看VPC下已创建的子网名称和ID。
安全组	安全组是一个逻辑上的分组，为同一个VPC内具有相同安全保护需求并相互信任的弹性云服务器提供访问策略。单击“查看安全组”可了解安全组详情。 **说明** • 为了确保您能够正常访问集群，需要放通安全组9200规则。 • 如果创建的集群为7.6.2及以上版本，则需要确保同安全组内节点之间的端口全放通。如果无法放通同安全组内节点之间的全部端口，请至少确保9300端口的通信。
安全模式	关闭安全模式。

8. 单击“下一步：高级配置”，设置集群自动快照和其他高级功能。
   a. 设置集群自动快照开关、基础配置和快照配置。

      系统默认打开集群快照开关，如果您不需要启用自动快照，可以在“集群快照开关”右侧关闭。自动快照会创建委托访问对象存储服务OBS，快照存储在标准存储中需额外计费。

      表 4-17 集群快照基础配置的参数说明

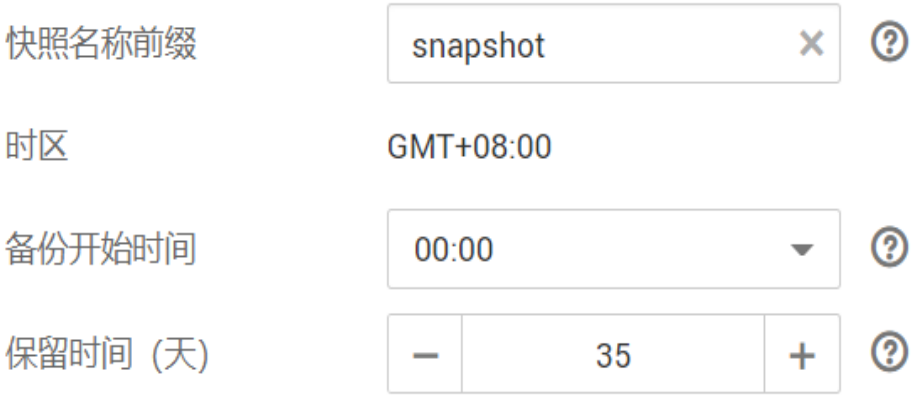
参数	说明
OBS桶	在下拉框中选择存储快照的OBS桶。也可以单击右侧的“创建桶”新建OBS。 创建或者已存在的OBS桶需满足如下条件： • “存储类别”为“标准存储”。
备份路径	快照在OBS桶中的存放路径。 备份路径配置规则： • 备份路径不能包括下列符号：\:*?"<>| • 备份路径不能以“/”开头。 • 备份路径不能以“.”开头或结尾。 • 备份路径的总长度不能超过1023个字符。

参数	说明
IAM委托	指当前帐号授权云搜索服务访问或维护存储在OBS中数据。也可以单击右侧的“创建委托”新建IAM委托。 创建或者已存在的IAM委托需满足如下条件： ● “委托类型”选择“云服务”。 ● “云服务”选择“Elasticsearch”或者“云搜索服务CSS”。 ● 设置当前委托具备“全局服务”中“对象存储服务”项目的“Tenant Administrator”权限。

表 4-18 集群快照自动创建快照的参数说明

参数	说明
快照名称前缀	快照名称前缀的长度为1～32个字符，只能包含小写字母、数字、中划线和下划线，且必须以小写字母开头。快照名称由快照名称前缀加上时间戳组成，例如自动生成的快照名称为“snapshot-1566921603720”。
时区	指备份时间对应的时区，不支持修改。基于此时区选择备份开始时间。
备份开始时间	指每天自动开始备份的时间，只能指定整点时间，如00:00、01:00，取值范围为00:00～23:00。请在下拉框中选择时间。
保留时间（天）	指备份的快照在OBS的保留时间，以“天”为单位，取值范围为1～90，您可以根据实际需求进行设置。系统在半点时刻会自动删除超过保留时间的快照。

图 4-9 设置自动创建快照的参数



b. 配置集群高级配置功能。

- **默认配置**：默认关闭“终端节点服务”、“Kibana公网访问”和“标签”功能，在集群创建完成后，若有需要也可以人工启用这些功能。

- **自定义**：根据需要选择开启“终端节点服务”和“标签”功能。

表 4-19 高级配置参数

参数	说明
终端节点服务	开启终端节点服务后，用户可以获得一个内网访问的域名，通过这个域名，可以在同一个vpc内访问该集群。详细配置请参考**终端节点服务**。
Kibana公网访问	非安全模式的集群，不支持启用“Kibana公网访问”。
标签	为集群添加标签，可以方便用户识别和管理拥有的集群资源。此处您可以选择“标签管理服务”中已定义好的“预定义标签”，也可以自己定义标签。详细标签使用请参考**标签管理**。

9. 单击“下一步：确认配置”，确认完成后单击“立即创建”开始创建集群。
10. 单击“返回集群列表”，系统将跳转到“集群管理”页面。您创建的集群将展现在集群列表中，且集群状态为“创建中”，创建成功后集群状态会变为“可用”。

    如果集群创建失败，请根据界面提示，重新创建集群。

4.5 快速访问 Elasticsearch 集群

CSS服务创建的Elasticsearch集群自带Kibana和Cerebro组件，支持一键打开Kibana和Cerebro，快速访问Elasticsearch集群。

通过 Kibana 访问集群

1. 登录云搜索服务管理控制台。
2. 在“集群管理”页面选择需要登录的集群，单击“操作”列中的“Kibana”进入Kibana登录界面。
   - 非安全模式的集群：将直接进入Kibana操作界面。
   - 安全模式的集群：需要在登录页面输入用户名和密码，单击“Log In”进入Kibana操作界面。用户名默认为admin，密码为创建集群时设置的管理员密码。
3. 登录成功后，可在Kibana界面进行相关操作访问Elasticsearch集群。

通过 Cerebro 访问集群

1. 登录云搜索服务管理控制台。
2. 在“集群管理”页面选择需要登录的集群，单击“操作”列中的“更多 > Cerebro”进入Cerebro登录页面。
   - 非安全模式的集群：单击Cerebro登录页面的集群名称即可进入Cerebro操作界面。
   - 安全模式的集群：单击Cerebro登录页面的集群名称，再输入用户名和密码，单击“Authenticate”进入Cerebro操作界面。用户名默认为admin，密码为创建集群时设置的管理员密码。

3. 登录成功后，可在Cerebro界面进行相关操作访问Elasticsearch集群。

4.6 查看集群的基本信息

在集群的基本信息页面，可以获取集群的内网访问地址、公网访问地址、版本、节点等信息。

1. 登录云搜索服务管理控制台。
2. 选择“集群管理 > Elasticsearch”，进入集群列表页面。
3. 单击集群名称进入集群“基本信息”页面，查看集群的基本信息。

表 4-20 基本信息的参数说明

类别	参数	描述
基本信息	集群名称	集群名称。支持自定义名称。 单击右侧  可以修改集群名称。
	ID	集群的唯一标识，是系统自动生成的。 同一个区域下，集群ID是唯一的。
	集群版本	集群的版本信息。
	集群状态	集群当前的状态。
	任务状态	集群当前的任务状态，若没有进行中的任务则显示“--”。
	创建时间	集群创建的时间。
	集群存储容量（GB）	集群设置的存储容量。
	集群存储使用量（GB）	集群已使用的存储容量。
配置信息	区域	集群所在区域。
	可用区	集群所在的可用区。
	虚拟私有云	集群所属的虚拟私有云。
	子网	集群所属的子网。
	安全组	集群所属的安全组。
	安全模式	集群的安全模式。 • 启用：表示当前集群是安全模式的集群。 • 未开启：表示当前集群是非安全集群。

类别	参数	描述
	重置密码	仅安全模式的集群显示。 单击“重置”可以修改安全集群的管理员帐户**admin**的密码。 **说明** 管理员密码的规则： ● 可输入的字符串长度为8~32个字符。 ● 密码至少包含大写字母、小写字母、数字和特殊字符四类中的三类。其中支持的特殊字符有：~!@#$%^&*()-_=+|[{}];:,<.>/? ● 不能与管理员帐户名或倒序的管理员帐户名相同。 ● 建议定期修改密码。
	企业项目	集群所属的企业项目。 单击项目名称可以跳转到项目管理页面查看企业项目的基本信息。
	访问控制	启用公网访问的集群是否设置访问控制，仅启用公网访问的集群显示。 ● 已开启：启用访问控制开关，只允许白名单列表中的IP地址通过公网访问集群。 ● 未开启：为启用访问控制开关，允许任何IP地址通过公网访问集群。 单击“设置”，可以更新访问控制开关和白名单。
	带宽	公网访问的带宽，仅启用公网访问的集群显示。 单击“修改”，可以更新带宽大小。
	HTTPS访问	集群是否启用HTTPS访问协议。 ● 关闭：表示集群未启用HTTPS访问，集群使用HTTP访问协议。 ● 开启：表示集群启用了HTTPS访问协议，仅安全模式的集群支持开启HTTPS访问。启用HTTPS访问的安全集群可以单击“下载证书”获取CER安全证书，用于接入安全模式的集群。

类别	参数	描述
	内网访问地址	集群的内网IP地址和端口号，使用此参数可以接入集群。如果集群只有一个节点，此处仅显示1个节点的IP地址和端口号，例如“10.62.179.32:9200”；如果集群有多个节点，此处显示所有节点的IP地址和端口号，例如“10.62.179.32:9200,10.62.179.33:9200”。
节点信息	节点规格	集群中节点的规格信息。
	节点存储	集群中节点的存储容量和存储类型。
	节点数量	集群中节点的个数。

5 集群形态变更

5.1 形态变更概述

CSS集群支持形态变更，包括集群扩容、集群规格变更、集群缩容等。当创建的集群规格不能满足业务需求时，可以通过形态变更，提高集群的使用效率，降低运维成本。

扩容

- 当**集群数据节点**（ess）的写入与查询压力大、响应时间过长时，可以通过扩容数据节点的“节点存储容量”保证数据的持久性。若因数据量过大或操作不当导致数据节点状态异常时，可以扩容“节点数量”保证集群的可用性。
- **冷数据节点**（ess-cold）主要用于分担ess数据节点的压力，当发现冷数据有丢失的风险时，可以扩容冷数据节点的“节点存储容量”保证冷数据的持久性，同时也支持扩容节点个数保证集群的可用性。

变更规格

- 当新增索引或分片分配的处理时间过长，或管理集群各个节点的协调、调度不足时，可以变更**Master节点**（ess-master）的“节点规格”保证集群的正常使用。
- 当数据节点任务分发量、结果汇聚量过大时，需要变更**Client节点**（ess-client）的“节点规格”。
- 当数据的写入与查询的突然变得缓慢时，可以变更**数据节点**（ess）的“节点规格”提高数据节点的查询与写入效率。
- 当存在冷数据查询缓慢时，可以变更**冷数据节点**（ess-cold）的“节点规格”提高对数据查询的效率。

缩容

当集群有充足的能力处理当前数据时，为节省资源可以随机减小集群占用的资源。

缩容指定节点

当集群有充足的能力处理当前数据时，为节省资源可以指定一个或多个节点进行缩容。

5.2 扩容

当集群数据面业务变化，需要动态调整集群节点的数量和容量时，可以执行“扩容”任务。扩容集群时，业务不会中断。

前提条件

- 集群处于“可用”状态，且无正在进行的任务。
- 有足够的配额支持集群扩容。

约束限制

- 扩容不支持修改“节点规格”。
- 扩容什么节点类型的“节点数量”和“节点存储容量”，扩容完成后只生效该节点类型的“节点数量”和“节点存储容量”，其他节点类型的“节点数量”和“节点存储容量”保持不变。
- 当集群包含的节点类型不同时，扩容的节点数量区间会有区别，具体情况请参考表5-1。

表 5-1 不同节点类型的节点数量说明

集群包含的节点类型	节点数量的取值范围
ess	ess：1~32
ess、ess-master	ess：1~200 ess-master：3~9的奇数
ess、ess-client	ess：1~32 ess-client：1~32
ess、ess-cold	ess：1~32 ess-cold：1~32
ess、ess-master、ess-client	ess：1~200 ess-master：3~9的奇数 ess-client：1~32
ess、ess-master、ess-cold	ess：1~200 ess-master：3~9的奇数 ess-cold：1~32
ess、ess-client、ess-cold	ess：1~32 ess-client：1~32 ess-cold：1~32

集群包含的节点类型	节点数量的取值范围
ess、ess-master、ess-client、ess-cold	ess：1~200 ess-master：3~9的奇数 ess-client：1~32 ess-cold：1~32
四种节点类型的说明： • ess：默认节点类型，即创建集群时必选的数据节点类型，其他3种节点类型都是基于业务需要可选的类型。 • ess-master：Master节点 • ess-client：Client节点 • ess-cold：冷数据节点	

操作步骤

1. 登录云搜索服务管理控制台。
2. 在左侧菜单栏选择“集群管理 > Elasticsearch”，进入集群列表页面，选择目标集群，单击操作列的“更多>形态变更”进入更改集群规格页面。
3. 在更改集群规格页面，设置扩容参数。
   - “修改后的节点数量”：此处修改的是默认数据节点类型的节点数量，支持修改的取值范围请参考表5-1。
   - “节点存储容量”：此处修改的是默认数据节点类型的存储容量，该取值范围由“节点规格”决定。只支持配置为20的倍数。

   说明

   如果集群启用了Master节点、Client节点或冷数据节点，还可以更改Master节点、Client节点与冷数据节点的“节点数量”，冷数据节点还支持“节点存储容量”扩容。其中，Master节点、Client节点与冷数据节点的节点数量取值范围请参考表5-1。

图 5-1 集群扩容

4. 单击“下一步：立即变更”。
5. 确认变更信息后，单击“提交申请”。
6. 单击“返回集群列表”跳转到集群管理页面。集群的“任务状态”列显示为“扩容”，表示集群正在扩容。当集群状态变为“可用”，则表示扩容成功。

5.3 变更规格

当集群数据面业务变化，需要动态调整集群节点的规格时，可以执行“变更规格”任务。

前提条件

- 集群处于“可用”状态，且无正在进行的任务。
- 有足够的配额支持集群变更规格。
- 变更规格时，为了不中断业务，请确认业务数据都有副本。

  在Kibana中执行命令**GET _cat/indices?v**，若回显的“rep”值大于“0”，则表示有副本；若“rep”值等于“0”，则表示没有副本，请先为集群**手动创建快照**再变更规格。
- 如果数据量比较大的情况下，更改节点规格耗时会比较长，因此，建议在业务低峰期更改节点规格，利于更快完成规格更改。

约束限制

- 变更规格不支持修改“节点数量”和“节点存储容量”。
- 若将大规格更改为小规格，集群的处理性能将会降低，将会影响业务能力，请谨慎操作。
- 当集群包含多种节点类型时，一次只支持变更一种类型的节点规格，且变更完成后只生效所选类型的节点规格。
- 变更规格过程中，Kibana不可用。
- 变更规格过程中，会依次对节点进行关机，完成更改后依次开机。是一个滚动的变更过程。

操作步骤

1. 登录云搜索服务管理控制台。
2. 在左侧菜单栏选择“集群管理 > Elasticsearch”，进入集群列表页面，选择目标集群，单击操作列的“更多>形态变更”进入更改集群规格页面。
3. 在更改集群规格页面，设置变更规格的参数。
   - “变更后节点规格”：此处修改的是默认数据节点类型的节点规格，在下拉框中选择所需的规格。
   - 如果集群启用了Master节点、Client节点或冷数据节点，还可以更改Master节点、Client节点与冷数据节点的“节点规格”。

   图 5-2 变更集群规格

4. 单击“下一步：立即变更”。

5. 确认变更信息后，单击“提交申请”。
6. 在弹出的“索引副本校验”窗口确认是否勾选“进行索引副本校验”，单击“确认”启动集群规格变更。
   - 没有Master节点的集群更改节点规格时，若选择进行索引副本校验，则要求所有索引至少有1个副本，且“节点数量”总和不小于3。
   - 有Master节点的集群更改节点规格时，若选择进行索引副本校验，则要求所有索引至少有1个副本。
7. 单击“返回集群列表”跳转到集群管理页面。集群的“任务状态”列中显示为“规格修改”，表示集群正在更改规格。当集群状态变为“可用”，则表示规格变更成功。

5.4 缩容

当集群有充足的能力处理当前数据时，为节省资源可以执行“缩容”任务，随机减小集群占用的资源。缩容集群时，业务不会中断。

前提条件

集群处于“可用”状态，且无正在进行的任务。

约束限制

- 缩容不支持修改“节点规格”和“存储容量”。
- 缩容什么节点类型的“节点数量”，缩容完成后只生效新该节点类型的“节点数量”，其他节点类型的“节点数量”保持不变。
- 要确保缩容之后的磁盘使用量小于80%，且集群每个节点类型中每个AZ的节点数至少为1。
- 缩容过程会涉及数据迁移，将要下线的节点数据迁移到其他节点上，数据迁移的超时阈值为5小时。当超过5小时数据还未迁移完成，那么缩容会失败。建议在集群数据量较大的情况下，分多次进行缩容。
- 如果集群没有启用Master节点，缩容后剩余的数据节点个数（包含冷数据节点和其他类型节点）须大于之前的一半，并大于索引的最大副本个数。
- 当集群包含的节点类型不同时，缩容的节点数量区间会有区别，具体情况请参考表5-2。

表 5-2 不同节点类型的节点数量说明

集群包含的节点类型	节点数量的取值范围
ess	ess：1~32
ess、ess-master	ess：1~200 ess-master：3~9的奇数
ess、ess-client	ess：1~32 ess-client：1~32
ess、ess-cold	ess：1~32 ess-cold：1~32

集群包含的节点类型	节点数量的取值范围
ess、ess-master、ess-client	ess：1~200 ess-master：3~9的奇数 ess-client：1~32
ess、ess-master、ess-cold	ess：1~200 ess-master：3~9的奇数 ess-cold：1~32
ess、ess-client、ess-cold	ess：1~32 ess-client：1~32 ess-cold：1~32
ess、ess-master、ess-client、ess-cold	ess：1~200 ess-master：3~9的奇数 ess-client：1~32 ess-cold：1~32
四种节点类型的说明： ● ess：默认节点类型，即创建集群时必选的数据节点类型，其他3种节点类型都是基于业务需要可选的类型。 ● ess-master：Master节点 ● ess-client：Client节点 ● ess-cold：冷数据节点	

操作步骤

1. 登录云搜索服务管理控制台。
2. 在左侧菜单栏选择“集群管理 > Elasticsearch”，进入集群列表页面，选择目标集群，单击操作列的“更多>形态变更”进入更改集群规格页面。
3. 在更改集群规格页面，设置缩容参数。
   - “修改后的节点数量”：此处修改的是默认数据节点类型的节点数量，支持修改的取值范围请参考**表5-2**。
   - 如果集群启用了Master节点、Client节点或冷数据节点，还可以更改Master节点、Client节点与冷数据节点的“节点数量”。其中，Master节点、Client节点与冷数据节点的节点数量取值范围请参考**表5-2**。

图 5-3 集群缩容

图 5-4 缩容集群

4. 单击“下一步：立即变更”。
5. 确认变更信息后，单击“提交申请”。
6. 单击“返回集群列表”跳转到集群管理页面。集群的“任务状态”列中显示为“缩容中”，表示集群正在缩容。当集群状态变为“可用”，则表示缩容成功。

5.5 缩容指定节点

当集群有充足的能力处理当前数据时，为节省资源可以执行“指定节点缩容”任务，指定一个或多个节点进行缩容。对集群进行指定节点缩容时，业务不会中断。

前提条件

集群处于“可用”状态，且无正在进行的任务。

约束限制

- 要确保缩容之后的磁盘使用量小于80%，且集群每个节点类型中每个AZ的节点数至少为1。
- 关于跨AZ的集群，在不同AZ中同类型节点个数的差值要小于等于1。
- 关于没有Master节点的集群，每次缩容的数据节点和冷数据节点个数之和要小于缩容前数据节点和冷数据节点个数之和的一半，缩容后的数据节点和冷数据节点个数之和要大于索引的最大副本个数。
- 关于有Master节点的集群，每次缩容的Master节点个数要小于当前Master节点总数的一半，缩容后的Master节点个数必须是奇数且不小于3。

操作步骤

1. 登录云搜索服务管理控制台。
2. 在左侧菜单栏选择“集群管理 > Elasticsearch”，进入集群列表页面，选择目标集群，单击操作列的“更多>形态变更”进入更改集群规格页面。
3. 选择“缩容指定节点”页签。
4. 在缩容指定节点页面，勾选需要缩容的节点。

   图 5-5 指定节点缩容

5. 单击“下一步：立即变更”。
6. 确认变更信息后，单击“提交申请”。
7. 单击“返回集群列表”跳转到集群管理页面。集群的“任务状态”列中显示为“缩容中”，表示集群正在缩容。当集群状态变为“可用”，则表示缩容成功。

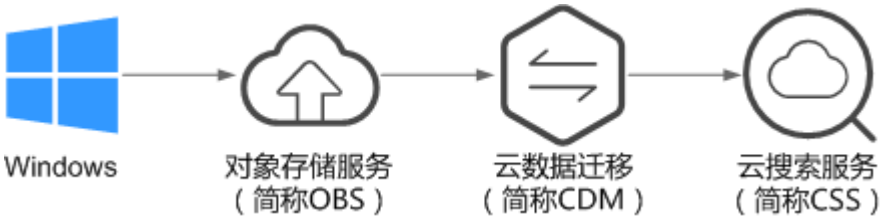
6 导入数据到 Elasticsearch

6.1 使用 CDM 从 OBS 导入数据到 Elasticsearch

云搜索服务支持通过CDM的向导式界面，将存储在对象存储服务（简称OBS）中的数据导入到Elasticsearch中。数据文件支持JSON、CSV等格式。

数据传输流程如图6-1所示。

图 6-1 使用 CDM 从 OBS 导入数据到 Elasticsearch 时的数据传输流程



操作步骤

1. 登录OBS管理控制台。
2. 创建待存储数据的OBS桶。
   具体操作请参见《对象存储服务控制台指南》中的创建桶。
   创建的OBS桶需满足“区域”必须跟创建集群的区域保持一致。
3. 将数据文件上传到OBS桶中。
   具体操作请参见《对象存储服务控制台指南》中的上传文件。
   例如：将如下数据保存为json格式的文件，上传到创建的OBS桶中。

```
{"productName":"2017秋装新款文艺衬衫女装","size":"L"}
{"productName":"2017秋装新款文艺衬衫女装","size":"M"}
{"productName":"2017秋装新款文艺衬衫女装","size":"S"}
{"productName":"2018春装新款牛仔裤女装","size":"M"}
{"productName":"2018春装新款牛仔裤女装","size":"S"}
{"productName":"2017春装新款休闲裤女装","size":"L"}
{"productName":"2017春装新款休闲裤女装","size":"S"}
```

4. 登录云搜索服务管理控制台。
5. 在左侧导航栏中，选择“集群管理 > Elasticsearch”，进入集群管理列表页面。

6. 在集群列表页面中，单击待导入数据的集群“操作”列的“Kibana”。
7. 在Kibana的左侧导航中选择“Dev Tools”，进入Console界面。
8. 在Console界面，执行命令创建待存储数据的索引，并指定自定义映射来定义数据类型。

   如果待导入数据的集群已存在可用的索引，则不需要再创建索引；如果待导入数据的集群不存在可用的索引，则需要参考如下示例创建索引。

   例如：在Console界面，执行如下命令，创建索引“demo”，并指定自定义映射来定义数据类型。

   7.x之前版本

   ```
   PUT /demo
   {
     "settings": {
       "number_of_shards": 1
     },
     "mappings": {
       "products": {
         "properties": {
           "productName": {
             "type": "text",
             "analyzer": "ik_smart"
           },
           "size": {
             "type": "keyword"
           }
         }
       }
     }
   }
   ```

   7.x之后版本

   ```
   PUT /demo
   {
     "settings": {
       "number_of_shards": 1
     },
     "mappings": {
         "properties": {
           "productName": {
             "type": "text",
             "analyzer": "ik_smart"
           },
           "size": {
             "type": "keyword"
           }
         }
       }
     }
   ```

   执行成功后显示如下：

   ```
   {
     "acknowledged" : true,
     "shards_acknowledged" : true,
     "index" : "demo"
   }
   ```

9. 登录CDM管理控制台。
10. 购买云数据迁移集群。

    具体操作请参见《 云数据迁移用户指南 》中的创建集群。

11. 新建CDM和云搜索服务的连接。

    具体操作请参见《 云数据迁移用户指南 》中的新建连接。

12. 新建CDM和OBS的连接。

具体操作请参见《 云数据迁移用户指南 》中的新建连接。

13. 在已购买的云数据迁移集群上新建作业，将OBS桶中的数据迁移到云搜索服务的待导入数据的集群中。

    具体操作请参见《 云数据迁移用户指南 》中的表/文件迁移。

14. 在已打开的Kibana的Console界面，通过搜索获取已导入的数据。

    在Kibana控制台，执行如下命令，搜索数据。查看搜索结果，如果数据与导入数据一致，表示数据文件的数据已导入成功。

    ```
    GET demo/_search
    ```

    执行成功后显示如下：

    ```
    {
      "took": 18,
      "timed_out": false,
      "_shards": {
        "total": 1,
        "successful": 1,
        "skipped": 0,
        "failed": 0
      },
      "hits": {
        "total": 7,
        "max_score": 1,
        "hits": [
          {
            "_index": "demo",
            "_type": "products",
            "_id": "g6UepnEBuvdFwWkRmn4V",
            "_score": 1,
            "_source": {
              "size": """"size":"L"}""",
              "productName": """{"productName":"2017秋装新款文艺衬衫女装""""
            }
          },
          {
            "_index": "demo",
            "_type": "products",
            "_id": "hKUepnEBuvdFwWkRmn4V",
            "_score": 1,
            "_source": {
              "size": """"size":"M"}""",
              "productName": """{"productName":"2017秋装新款文艺衬衫女装""""
            }
          },
          {
            "_index": "demo",
            "_type": "products",
            "_id": "haUepnEBuvdFwWkRmn4V",
            "_score": 1,
            "_source": {
              "size": """"size":"S"}""",
              "productName": """{"productName":"2017秋装新款文艺衬衫女装""""
            }
          },
          {
            "_index": "demo",
            "_type": "products",
            "_id": "hqUepnEBuvdFwWkRmn4V",
            "_score": 1,
            "_source": {
              "size": """"size":"M"}""",
              "productName": """{"productName":"2018春装新款牛仔裤女装""""
            }
          },
          {
            "_index": "demo",
            "_type": "products",
    ```



```
      "_id": "h6UepnEBuvdFwWkRmn4V",
      "_score": 1,
      "_source": {
        "size": """"size":"S"}""",
        "productName": """{"productName":"2018春装新款牛仔裤女装""""
      }
    },
    {
      "_index": "demo",
      "_type": "products",
      "_id": "iKUepnEBuvdFwWkRmn4V",
      "_score": 1,
      "_source": {
        "size": """"size":"L"}""",
        "productName": """{"productName":"2017春装新款休闲裤女装""""
      }
    },
    {
      "_index": "demo",
      "_type": "products",
      "_id": "iaUepnEBuvdFwWkRmn4V",
      "_score": 1,
      "_source": {
        "size": """"size":"S"}""",
        "productName": """{"productName":"2017春装新款休闲裤女装""""
      }
    }
  ]
 }
}
```

说明

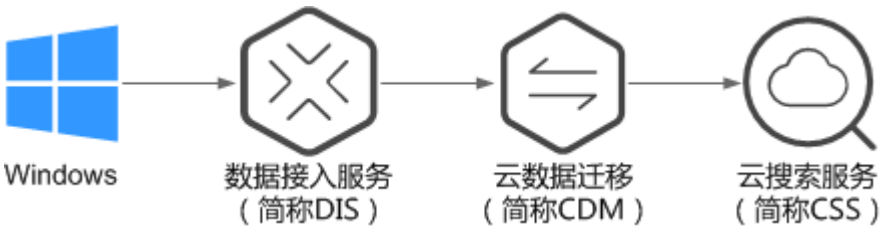
demo为创建的索引名称，需根据实际情况填写。

6.2 使用 DIS 导入本地数据到 Elasticsearch

通过DIS可以将本地windows系统上的日志数据上传到DIS队列中，然后通过CDM可以将DIS队列中的数据迁移到云搜索服务的Elasticsearch中，从而方便用户使用Elasticsearch搜索引擎高效管理和获取日志。数据文件支持JSON、CSV等格式。

数据传输流程如图6-2所示。

图 6-2 使用 DIS 导入本地数据到 Elasticsearch 时的数据传输流程



操作步骤

1. 登录DIS管理控制台。
2. 购买接入通道。

   具体操作请参见《数据接入服务用户指南》中的开通DIS通道。
3. 安装并配置DIS Agent。

具体操作请参见《数据接入服务用户指南》中的安装DIS Agent和配置DIS Agent。

4. 启动DIS Agent，将采集的本地数据上传到DIS队列中。

   具体操作请参见《数据接入服务用户指南》中的启动DIS Agent。

   例如：将如下数据通过DIS Agent上传到DIS队列中。

   ```
   {"logName":"aaa","date":"bbb"}
   {"logName":"ccc","date":"ddd"}
   {"logName":"eee","date":"fff"}
   {"logName":"ggg","date":"hhh"}
   {"logName":"mmm","date":"nnn"}
   ```

5. 登录云搜索服务管理控制台。
6. 在左侧导航栏中，选择“集群管理 > Elasticsearch”，进入集群管理列表页面。
7. 在集群列表页面中，单击待导入数据的集群“操作”列的“Kibana”。
8. 在Kibana的左侧导航中选择“Dev Tools”，进入Console界面。
9. 在Console界面，执行命令创建待存储数据的索引，并指定自定义映射来定义数据类型。

   如果待导入数据的集群已存在可用的索引，则不需要再创建索引；如果待导入数据的集群不存在可用的索引，则需要参考如下示例创建索引。

   例如：在Console界面，执行如下命令，创建索引“apache”，并指定自定义映射来定义数据类型。

   7.x之前版本

   ```
   PUT /apache
   {
      "settings": {
         "number_of_shards": 1
      },
      "mappings": {
         "logs": {
            "properties": {
               "logName": {
                  "type": "text",
                  "analyzer": "ik_smart"
               },
               "date": {
                  "type": "keyword"
               }
            }
         }
      }
   }
   ```

   7.x之后版本

   ```
   PUT /apache
   {
      "settings": {
         "number_of_shards": 1
      },
      "mappings": {
                  "properties": {
               "logName": {
                  "type": "text",
                  "analyzer": "ik_smart"
               },
               "date": {
                  "type": "keyword"
               }
            }
         }
   }
   ```


执行成功后显示如下：

```
{
  "acknowledged" : true,
  "shards_acknowledged" : true,
  "index" : "apache"
}
```

10. 登录CDM管理控制台。
11. 购买云数据迁移集群。

    具体操作请参见《 云数据迁移用户指南 》中的创建集群。
12. 新建CDM和云搜索服务的连接。

    具体操作请参见《 云数据迁移用户指南 》中的新建连接。
13. 新建CDM和DIS的连接。

    具体操作请参见《 云数据迁移用户指南 》中的新建连接。
14. 在已购买的云数据迁移集群上新建作业，将DIS队列中的数据迁移到云搜索服务的待导入数据的集群中。

    具体操作请参见《 云数据迁移用户指南 》中的表/文件迁移。
15. 在已打开的Kibana的Console界面，通过搜索获取已导入的数据。

    在Kibana控制台，输入如下命令，搜索数据。查看搜索结果，如果数据与导入数据一致，表示数据文件的数据已导入成功。

    ```
    GET apache/_search
    ```

    执行成功后显示如下：

    ```
    {
      "took": 81,
      "timed_out": false,
      "_shards": {
        "total": 1,
        "successful": 1,
        "skipped": 0,
        "failed": 0
      },
      "hits": {
        "total": 5,
        "max_score": 1,
        "hits": [
          {
            "_index": "apache",
            "_type": "logs",
            "_id": "txfbqnEBPuwwWJWL-qvP",
            "_score": 1,
            "_source": {
              "date": """{"logName":"aaa""",
              "logName": """date":"bbb"}"""
            }
          },
          {
            "_index": "apache",
            "_type": "logs",
            "_id": "uBfbqnEBPuwwWJWL-qvP",
            "_score": 1,
            "_source": {
              "date": """{"logName":"ccc""",
              "logName": """date":"ddd"}"""
            }
          },
          {
            "_index": "apache",
            "_type": "logs",
            "_id": "uRfbqnEBPuwwWJWL-qvP",
            "_score": 1,
    ```

```
      "_source": {
        "date": """{"logName":"eee"""",
        "logName": """"date":"fff"}""""
      }
    },
    {
      "_index": "apache",
      "_type": "logs",
      "_id": "uhfbqnEBPuwwWJWL-qvP",
      "_score": 1,
      "_source": {
        "date": """{"logName":"ggg"""",
        "logName": """"date":"hhh"}""""
      }
    },
    {
      "_index": "apache",
      "_type": "logs",
      "_id": "uxfbqnEBPuwwWJWL-qvP",
      "_score": 1,
      "_source": {
        "date": """{"logName":"mmm"""",
        "logName": """"date":"nnn"}""""
      }
    }
  ]
 }
}
```

说明

apache为创建的索引名称，需根据实际情况填写。

6.3 使用 Logstash 导入数据到 Elasticsearch

云搜索服务支持使用Logstash将其收集的数据迁移到Elasticsearch中，方便用户通过Elasticsearch搜索引擎高效管理和获取数据。数据文件支持JSON、CSV等格式。

Logstash 是开源的服务器端数据处理管道，能够同时从多个来源采集数据、转换数据，然后将数据发送到Elasticsearch中。Logstash的官方文档请参见：**https://www.elastic.co/guide/en/logstash/current/getting-started-with-logstash.html**。

数据导入分为如下2种场景：

- **Logstash部署在外网时导入数据**
- **Logstash部署在弹性云服务器上时导入数据**

前提条件

- 为方便操作，建议采用Linux操作系统的机器部署Logstash。
- Logstash的下载路径为：**https://www.elastic.co/cn/downloads/logstash-oss**

  说明

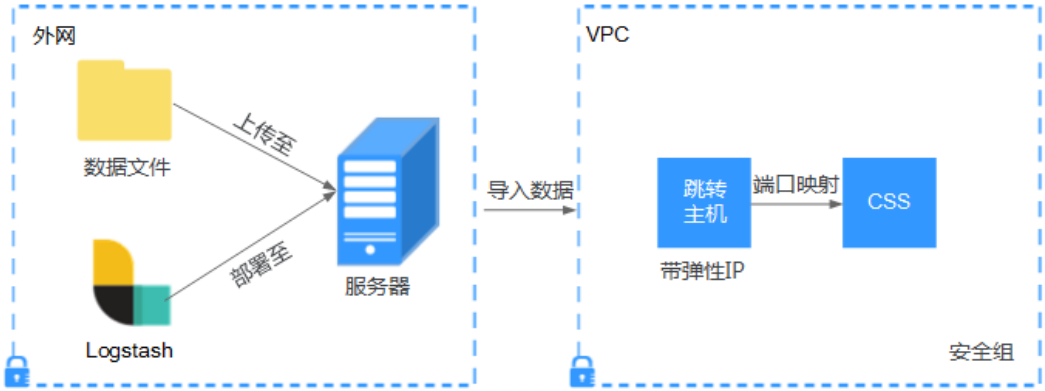
  Logstash要求使用OSS版本，选择和CSS一致版本。
- 安装完Logstash后，再根据如下步骤导入数据。安装Logstash的操作指导，请参见：**https://www.elastic.co/guide/en/logstash/current/installing-logstash.html**

- 安装Logstash之前，需要先安装JDK。在Linux操作系统中，您可以执行**yum -y install java-1.8.0**命令直接安装1.8.0版本JDK。在Windows操作系统中，您可以访问**JDK官网**，下载符合操作系统版本的JDK，并根据指导安装。
- 在“**Logstash部署在弹性云服务器上时导入数据**”场景中，请确保此弹性云服务器与接入的Elasticsearch集群在同一个VPC下。

Logstash 部署在外网时导入数据

当Logstash部署在外网时，导入数据的流程说明如**图6-3**所示。

图 6-3 Logstash 部署在外网时导入数据示意图



1. 创建一个跳转主机，并按如下要求进行配置。
   - 跳转主机为一台Linux操作系统的弹性云服务器，且已绑定弹性IP。
   - 跳转主机与CSS集群在同一虚拟私有云下。
   - 已开放跳转主机的本地端口，用于SSH转发，能够从本地端口转发至CSS集群某一节点的9200端口。
   - 关于跳转主机的本地端口转发配置，请参见**SSH官方文档**。
2. 使用PuTTY，通过弹性IP登录已创建的跳转主机。
3. 执行如下命令进行端口映射，将发往跳转主机对外开放端口的请求转发到待导入数据的集群中。

   ```
   ssh -g -L <跳转主机的本地端口:节点的内网访问地址和端口号> -N -f root@<跳转主机的私网IP地址>
   ```

   说明

   - *<跳转主机的本地端口>*：为步骤**1**中的端口。
   - *<节点的内网访问地址和端口号>*：为集群中某一节点的内网访问地址和端口号。当该节点出现故障时，将导致命令执行失败。如果集群包含多个节点，可以将*<节点的内网访问地址和端口号>*替换为集群中另一节点的内网访问地址和端口号；如果集群只包含一个节点，则需要将该节点修复之后再次执行命令进行端口映射。
   - *<跳转主机的私网IP地址>*：打开弹性云服务器管理控制台，从“IP地址”列中获取标有“私网”对应的IP地址。

   例如：跳转主机对外开放的端口号为9200，节点的内网访问地址和端口号为192.168.0.81:9200，跳转主机的私网IP地址为192.168.0.227，需要执行如下命令进行端口映射。

   ```
   ssh -g -L 9200:192.168.0.81:9200 -N -f root@192.168.0.227
   ```

4. 登录部署了Logstash的服务器，将需要进行操作的数据文件存储至此服务器中。

例如，需要导入的数据文件“access_20181029_log”，文件存储路径为“/tmp/access_log/”，此数据文件中包含的数据如下所示：

说明

文件存储路径中的access_log文件夹如果不存在，用户可以自建。

```
|   All |                 Heap used for segments |                       |    18.6403 |      MB |
|   All |              Heap used for doc values |                       |   0.119289 |      MB |
|   All |                   Heap used for terms |                       |    17.4095 |      MB |
|   All |                   Heap used for norms |                       |  0.0767822 |      MB |
|   All |                  Heap used for points |                       |   0.225246 |      MB |
|   All |           Heap used for stored fields |                       |   0.809448 |      MB |
|   All |                         Segment count |                       |        101 |         |
|   All |                      Min Throughput |         index-append |    66232.6 |  docs/s |
|   All |                   Median Throughput |         index-append |    66735.3 |  docs/s |
|   All |                      Max Throughput |         index-append |    67745.6 |  docs/s |
|   All |            50th percentile latency |         index-append |    510.261 |      ms |
```

5. 在部署Logstash的服务器中，执行如下命令在Logstash的安装目录下新建配置文件logstash-simple.conf。

```
cd /<Logstash的安装目录>/
vi logstash-simple.conf
```

6. 在配置文件logstash-simple.conf中输入如下内容。

```
input {
数据所在的位置
}
filter {
数据的相关处理
}
output {
    elasticsearch {
        hosts => "<跳转主机的公网IP地址>:<跳转主机对外开放的端口号>"
    }
}
```

- input：指明了数据的来源。实际请根据用户的具体情况来设置。input参数的详细解释和使用介绍请参见**https://www.elastic.co/guide/en/logstash/current/input-plugins.html**。
- filter：指定对数据进行处理的方式。例如，对日志进行了提取和处理，将非结构化信息转换为结构化信息。filter参数的详细解释和使用介绍请参见**https://www.elastic.co/guide/en/logstash/current/filter-plugins.html**。
- output：指明了数据的目的地址。output参数的详细解释和使用介绍请参见**https://www.elastic.co/guide/en/logstash/current/output-plugins.html**。*<跳转主机的公网IP地址>*请从弹性云服务器管理控制台的“IP地址”列中获取标有“弹性公网”对应的IP地址。*<跳转主机对外开放的端口号>*即为步骤**1**中的端口，例如：9200。

以步骤**4**中“/tmp/access_log/”的数据文件为例，输入数据文件从首行开始，且过滤条件保持为空，即不做任何数据处理操作。跳转主机的公网IP和端口号为“192.168.0.227:9200”。导入数据的索引名称为“myindex”。配置文件的示例如下所示，配置文件按实际数据情况修改完成后，输入“:wq”保存。

```
input {
    file{
      path => "/tmp/access_log/*"
      start_position => "beginning"
    }
}
filter {
}
output {
    elasticsearch {
      hosts => "192.168.0.227:9200"
```



```
    index => "myindex"

  }
}
```

说明

如果在使用中出现license相关的报错，可以尝试设置ilm_enabled => false。

如果集群开启了安全模式，则需要先下载证书。

a. 在集群基本信息页面下载证书。

图 6-4 下载证书

b. 将下载的证书存放到部署logstash服务器中。

c. 修改配置文件logstash-simple.conf。

以步骤4中“/tmp/access_log/”的数据文件为例，输入数据文件从首行开始，且过滤条件保持为空，即不做任何数据处理操作。跳转主机的公网IP和端口号为“192.168.0.227:9200”。导入数据的索引名称为“myindex”，证书存放路径为“/logstash/logstash6.8/config/CloudSearchService.cer”。配置文件的示例如下所示，配置文件按实际数据情况修改完成后，输入“:wq”保存。

```
input{
  file {
    path => "/tmp/access_log/*"
    start_position => "beginning"
  }
}
filter {
  }
output{
  elasticsearch{
    hosts => ["https://192.168.0.227:9200"]
    index => "myindex"
    user => "admin"
    password => "******"
    cacert => "/logstash/logstash6.8/config/CloudSearchService.cer"
  }
}
```

说明

password：登录安全集群的密码。

7. 执行如下命令将Logstash收集的数据导入到集群中。
```
./bin/logstash -f logstash-simple.conf
```

说明

此命令需要在存放logstash-simple.conf文件的目录下执行。例如，logstash-simple.conf文件存放在/root/logstash-7.1.1/，则需要先进入该路径，再执行此命令。

8. 登录云搜索服务管理控制台。
9. 在左侧导航栏中，选择“集群管理 > Elasticsearch”，进入集群管理列表页面。
10. 在集群列表页面中，单击待导入数据的集群“操作”列的“Kibana”。
11. 在Kibana的左侧导航中选择“Dev Tools”，进入Console界面。
12. 在已打开的Kibana的Console界面，通过搜索获取已导入的数据。

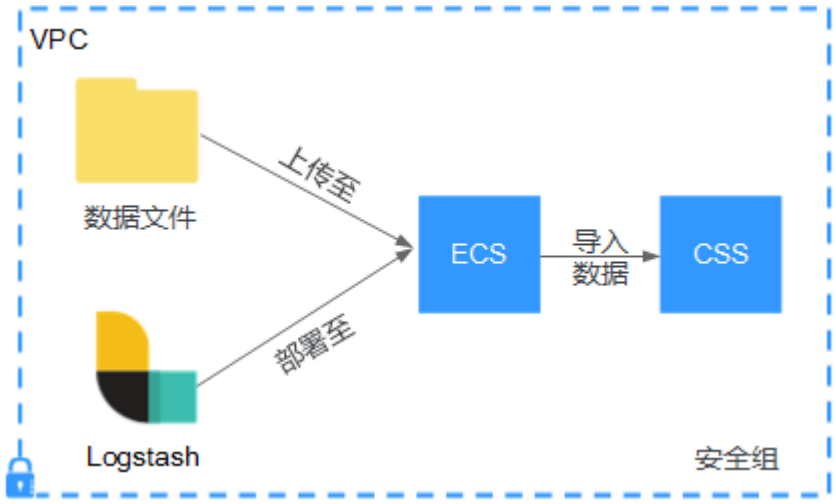
    在Kibana控制台，输入如下命令，搜索数据。查看搜索结果，如果数据与导入数据一致，表示数据文件的数据已导入成功。

```
GET myindex/_search
```

Logstash 部署在弹性云服务器上时导入数据

当Logstash部署在同一VPC的弹性云服务时，导入数据的流程说明如图6-5所示。

图 6-5 Logstash 部署在弹性云服务器上时导入数据示意图



1. 确保已部署Logstash的弹性云服务器与待导入数据的集群在同一虚拟私有云下，已开放安全组的9200端口的外网访问权限，且弹性云服务器已绑定弹性IP。
2. 使用PuTTY登录弹性云服务器。

   例如此服务器中存储了需要导入的数据文件“access_20181029_log”，文件存储路径为“/tmp/access_log/”，此数据文件中包含的数据如下所示：

```
|   All |                    Heap used for segments |                     |     18.6403 |      MB |
|   All |                  Heap used for doc values |                     |     0.119289 |      MB |
|   All |                       Heap used for terms |                     |     17.4095 |      MB |
|   All |                       Heap used for norms |                     |     0.0767822 |      MB |
|   All |                      Heap used for points |                     |     0.225246 |      MB |
|   All |            Heap used for stored fields |                     |     0.809448 |      MB |
|   All |                          Segment count |                     |        101 |         |
|   All |                         Min Throughput |         index-append |      66232.6 |  docs/s |
|   All |                      Median Throughput |         index-append |      66735.3 |  docs/s |
|   All |                         Max Throughput |         index-append |      67745.6 |  docs/s |
|   All |               50th percentile latency |         index-append |      510.261 |      ms |
```

3. 执行如下命令在Logstash的安装目录下新建配置文件logstash-simple.conf。

```
cd /<Logstash的安装目录>/
vi logstash-simple.conf
```

在配置文件logstash-simple.conf中输入如下内容。

```
input {
数据所在的位置
}
filter {
数据的相关处理
}
output {
    elasticsearch{
        hosts => "<节点的内网访问地址和端口号>"}
}
```

- input：指明了数据的来源。实际请根据用户的具体情况来设置。input参数的详细解释和使用介绍请参见**https://www.elastic.co/guide/en/logstash/current/input-plugins.html**。
- filter：对日志进行了提取和处理，将非结构化信息转换为结构化信息。filter参数的详细解释和使用介绍请参见**https://www.elastic.co/guide/en/logstash/current/filter-plugins.html**。
- output：指明了数据的目的地址。output参数的详细解释和使用介绍请参见**https://www.elastic.co/guide/en/logstash/current/output-plugins.html**。*<节点的内网访问地址和端口号>*为集群中节点的内网访问地址和端口号。

  当集群包含多个节点时，为了避免节点故障，建议将上述命令中*<节点的内网访问地址和端口号>*替换为该集群中多个节点的内网访问地址和端口号，多个节点的内网访问地址和端口号之间用英文逗号隔开，填写格式请参见如下示例。

  ```
  hosts => ["192.168.0.81:9200","192.168.0.24:9200"]
  ```

  当集群只包含一个节点时，填写格式请参见如下示例。

  ```
  hosts => "192.168.0.81:9200"
  ```

以步骤**2**中“/tmp/access_log/”的数据文件为例，输入数据文件从首行开始，且过滤条件保持为空，即不做任何数据处理操作。需导入数据的集群，其节点内网访问地址和端口号为“192.168.0.81:9200”。导入数据的索引名称为“myindex”。配置文件的示例如下所示，配置文件按实际数据情况修改完成后，输入“:wq”保存。

```
input {
    file{
      path => "/tmp/access_log/*"
      start_position => "beginning"
    }
}
filter {
}
output {
    elasticsearch {
      hosts => "192.168.0.81:9200"
      index => "myindex"

    }
}
```

如果集群开启了安全模式，则需要先下载证书。

a. 在集群基本信息页面下载证书。

图 6-6 下载证书

b. 将下载的证书存放到部署logstash服务器中。

c. 修改配置文件logstash-simple.conf。

以步骤2中“/tmp/access_log/”的数据文件为例，输入数据文件从首行开始，且过滤条件保持为空，即不做任何数据处理操作。跳转主机的公网IP和端口号为“192.168.0.227:9200”。导入数据的索引名称为“myindex”，证书存放路径为“/logstash/logstash6.8/config/CloudSearchService.cer”。配置文件的示例如下所示，配置文件按实际数据情况修改完成后，输入“:wq”保存。

```
input{
    file {
        path => "/tmp/access_log/*"
        start_position => "beginning"
    }
}
filter {
    }
output{
    elasticsearch{
        hosts => ["https://192.168.0.227:9200"]
        index => "myindex"
        user => "admin"
        password => "******"
        cacert => "/logstash/logstash6.8/config/CloudSearchService.cer"
    }
}
```

说明

password：登录安全集群的密码。

4. 执行如下命令将Logstash收集的弹性云服务器的数据导入到集群中。

```
./bin/logstash -f logstash-simple.conf
```

5. 登录云搜索服务管理控制台。
6. 在左侧导航栏中，选择“集群管理 > Elasticsearch”，进入集群管理列表页面。
7. 在集群列表页面中，单击待导入数据的集群“操作”列的“Kibana”。
8. 在Kibana的左侧导航中选择“Dev Tools”，进入Console界面。
9. 在已打开的Kibana的Console界面，通过搜索获取已导入的数据。

在Kibana控制台，输入如下命令，搜索数据。查看搜索结果，如果数据与导入数据一致，表示数据文件的数据已导入成功。


```
GET myindex/_search
```

6.4 使用 Kibana 或 API 导入数据到 Elasticsearch

云搜索服务支持使用Kibana或者API将数据导入到Elasticsearch中，数据文件支持JSON、CSV等格式。

使用 Kibana 导入数据

在导入数据之前，您可以使用Kibana接入集群。如下操作步骤介绍如何使用POST命令导入数据。

1. 登录云搜索服务管理控制台。
2. 在左侧导航栏中，选择“集群管理 > Elasticsearch”，进入集群管理列表页面。
3. 选择已创建的集群，单击操作列“Kibana”，登录Kibana。
4. 单击左侧导航栏的“Dev Tools”进入操作页面。
5. （可选）执行命令创建待存储数据的索引，并指定自定义映射来定义数据类型。

   如果待导入数据的集群已存在可用的索引，则不需要再创建索引；如果待导入数据的集群不存在可用的索引，则需要参考如下示例创建索引。

   例如：在Console界面，执行如下命令，创建索引“my_store”，并指定自定义映射来定义数据类型。

   7.x之前版本

   ```
   PUT /my_store
   {
     "settings": {
       "number_of_shards": 1
     },
     "mappings": {
       "products": {
         "properties": {
           "productName": {
             "type": "text"
           },
           "size": {
             "type": "keyword"
           }
         }
       }
     }
   }
   ```

   7.x之后版本

   ```
   PUT /my_store
   {
     "settings": {
       "number_of_shards": 1
     },
     "mappings": {
       "properties": {
         "productName": {
           "type": "text"
         },
         "size": {
           "type": "keyword"
         }
       }
     }
   }
   ```

6. 执行命令导入数据，以导入一条数据为例，执行如下命令。

7.x之前版本

```
POST /my_store/products/_bulk
{"index":{}}
{"productName":"Latest art shirts for women in 2017 autumn","size":"L"}
```

7.x之后版本

```
POST /my_store/_bulk
{"index":{}}
{"productName":"Latest art shirts for women in 2017 autumn","size":"L"}
```

返回结果如图6-7所示，当返回结果信息中“errors”字段的值为“false”时，表示导入数据成功。

图 6-7 返回消息

使用 API 导入数据

使用bulk API通过curl命令导入数据文件，如下操作以JSON数据文件为例。

说明

使用API导入数据文件时，建议导入的数据文件大小不能超过50MB。

1. 登录即将接入集群的弹性云服务器。
2. 执行如下命令，导入JSON数据。

   其中，**{*Private network address and port number of the node*}**需替换为集群中节点的内网访问地址和端口号，当该节点出现故障时，将导致命令执行失败。如果集群包含多个节点，可以将**{*Private network address and port number of the node*}**替换为集群中另一节点的内网访问地址和端口号；如果集群只包含一个节点，则需要将该节点修复之后再次执行命令进行导入数据。
   test.json为导入数据的json文件。

   ```
   curl -X PUT "http://{Private network address and port number of the node} /_bulk" -H 'Content-Type: application/json' --data-binary @test.json
   ```


说明

其中，-X参数的参数值为命令，如“-X PUT”，-H参数的参数值为消息头，如“-H 'Content-Type: application/json' --data-binary @test.json”。添加的-k参数时，请勿将-k参数放置在参数与参数值之间。

示例：将“testdata.json”数据文件中的数据导入至Elasticsearch集群，此集群未进行通信加密，其中一个节点内网访问地址为“192.168.0.90”，端口号为“9200”。其中testdata.json文件中的数据如下所示：

7.x之前版本

```
{"index": {"_index":"my_store","_type":"products"}}
{"productName": "2019秋装新款文艺衬衫女装","size": "M"}
{"index": {"_index":"my_store","_type":"products"}}
{"productName": "2019秋装新款文艺衬衫女装","size": "L"}
```

7.x之后版本

```
{"index": {"_index":"my_store"}}
{"productName": "2019秋装新款文艺衬衫女装","size": "M"}
{"index": {"_index":"my_store"}}
{"productName": "2019秋装新款文艺衬衫女装","size": "L"}
```

导入数据的操作步骤如下所示：

a. 可执行以下命令，创建my_store索引。

7.x之前版本

```
curl -X PUT http://192.168.0.90:9200/my_store -H 'Content-Type: application/json' -d '
 {
   "settings": {
     "number_of_shards": 1
   },
   "mappings": {
     "products": {
       "properties": {
         "productName": {
           "type": "text"
         },
         "size": {
           "type": "keyword"
         }
       }
     }
   }
 }'
```

7.x之后版本

```
curl -X PUT http://192.168.0.90:9200/my_store -H 'Content-Type: application/json' -d '
{
    "settings": {
        "number_of_shards": 1
    },
    "mappings": {
        "properties": {
            "productName": {
                "type": "text"
            },
            "size": {
                "type": "keyword"
            }
        }
    }
}'
```

b. 执行以下命令，导入testdata.json文件中的数据。

```
curl -X PUT "http://192.168.0.90:9200/_bulk" -H 'Content-Type: application/json' --data-binary @testdata.json
```


7 管理 Elasticsearch 类型集群

7.1 集群状态和存储容量状态说明

在云搜索服务管理控制台，直接展现当前云搜索服务中已有集群的状态以及集群存储容量状态。

表 7-1 集群状态说明

状态	说明
可用	表示集群服务正常运行中，并为用户提供服务。
异常	表示集群创建失败或不可用。 如果此集群处于“不可用”状态，支持删除集群操作或将集群正常状态时创建的快照恢复至其他集群。无法执行扩容集群、访问Kibana、创建快照或将快照恢复至此集群的操作。建议不要执行导入数据的操作，避免数据丢失。您可以查看监控或重启集群，但根据集群故障情况不同这些操作可能执行失败，当执行失败时，请及时联系技术支持。
处理中	表示集群正处在重启中、扩容中、备份中或恢复中。
创建中	表示集群正处在创建过程中。

表 7-2 集群存储容量状态

状态	说明
空闲	表示集群中所有节点存储容量使用率小于50%。
警告	表示集群中任一节点存储容量使用率大于 等于50%，小于80%。
危险	表示集群中任一节点存储容量使用率大于等于80%。建议增加集群的存储容量，以便能够正常使用集群进行数据搜索或分析。

状态	说明
异常	表示未能查询到集群的存储容量信息。例如，集群运行故障，状态为“异常”时，此集群的存储容量状态为“异常”。

7.2 集群列表简介

集群列表显示云搜索服务所有的集群，集群数量较多时，可采用翻页显示，您可以查看任何状态下的集群。

集群列表默认按时间顺序排列，时间最近的集群显示在最前端。集群列表参数说明如表7-3所示。

在集群列表右上角，您可以指定集群名称或集群ID，然后单击 进行查找。也可以单击右上角的，刷新集群列表。

表 7-3 集群列表说明

参数	描述
名称/ID	表示集群的名称和ID。单击集群名称可进入集群“基本信息”页面，展现了集群的基本信息。集群ID是系统自动生成的，是集群在服务中的唯一标示。
集群状态	展示集群当前的状态。集群状态说明请参见**集群状态和存储容量状态说明**。
任务状态	展示重启集群、扩容集群、备份集群、恢复集群等任务的状态。
版本	表示此集群中Elasticsearch的版本号。
创建时间	表示集群的创建时间。
企业项目	表示集群所归属的企业项目。
内网访问地址	集群的内网访问地址和端口号，您可以使用此参数接入集群。集群有多个节点时，此处显示多个节点的内网访问地址和端口号。
计费模式	呈现集群的计费模式。
操作	展示集群可执行的操作入口，包含Kibana、监控信息、重启、删除等其他更多操作。当某一操作无法执行时，显示为灰色链接。

7.3 备份与恢复索引

为避免数据丢失，您可以将集群的索引数据进行备份，当数据发生丢失或者想找回某一时间段数据时，您可以通过恢复索引操作快速获得数据。索引的备份是通过创建集群快照实现。第一次备份时，建议将所有索引数据进行备份。

- **管理自动创建快照**：自动创建快照指按照设置的规则，每天在指定时间自动创建快照。您可以开启自动创建功能、设置自动创建的策略、和关闭自动创建功能。
- **手动创建快照**：在任意时间，您通过手动创建快照的方式，针对当时的数据或某几个索引创建快照进行备份。
- **恢复数据**：将已有的快照，通过恢复快照功能，将备份的索引数据恢复到指定的集群中。
- **删除快照**：对于已失效的快照，建议删除以释放存储资源。

说明

- 创建快照之前，您需要进行基础配置，包含存储快照的OBS桶、快照的备份路径及安全认证使用的IAM委托。
- 集群快照存储的OBS桶，在首次设置后，不管自动创建快照还是手动创建快照，如果快照列表中已有可用的快照，则OBS桶将无法再变更，请谨慎选择存储OBS桶。
- 如果OBS桶已经存储了快照，OBS无法变更，您可以使用这个方法修改：首先关闭快照功能，然后再开启快照功能，指定新的OBS桶。一旦关闭快照功能，之前创建的快照将无法用于恢复集群。
- 当集群处于“不可用”状态时，快照功能中，除了恢复快照功能外，其他快照信息或功能只能查看，无法进行编辑。
- 备份与恢复过程中，支持集群扩容、访问Kibana、查看监控、删除其他快照的操作。不支持重启此集群、删除此集群、删除正在创建或恢复的快照、再次创建或恢复快照的操作。补充说明，当此集群正在进行创建快照或者恢复快照时，此时，自动创建快照任务将被取消。
- CSS集群第一次快照是全量，后面再备份快照是在之前的快照基础上增量，CSS是增量快照逻辑，快照之间的文件会相互依赖。
- CSS集群快照恢复到另一个集群会覆盖标集群中的同名索引，不同名的索引不会覆盖。如果两个集群的shard不一样，则同名的索引不会被覆盖。

前提条件

登录云搜索服务管理控制台的帐号或IAM用户必须同时具备如下权限才能使用创建或恢复快照功能。

- “全局服务”中“对象存储服务”项目的“Tenant Administrator”权限。
- 当前所属区域的“Elasticsearch Administrator”权限。

管理自动创建快照

1. 在云搜索服务管理控制台，单击左侧导航栏的“集群管理”。
2. 在“集群管理”页面，单击需要进行备份的集群名称，进入集群基本信息页面。在左侧导航栏选择“集群快照”，进入“集群快照”管理页面。
3. 在“集群快照”管理页面，在“集群快照开关”右侧单击开关，打开集群快照功能。
4. 打开集群快照功能后，云搜索服务会自动为客户创建OBS桶和IAM委托，用于存储快照。自动创建的OBS桶和IAM委托将直接展示在界面中。如果您不希望使用自动创建的OBS桶和IAM委托，您可以在“基础配置”右侧单击 进行配置。

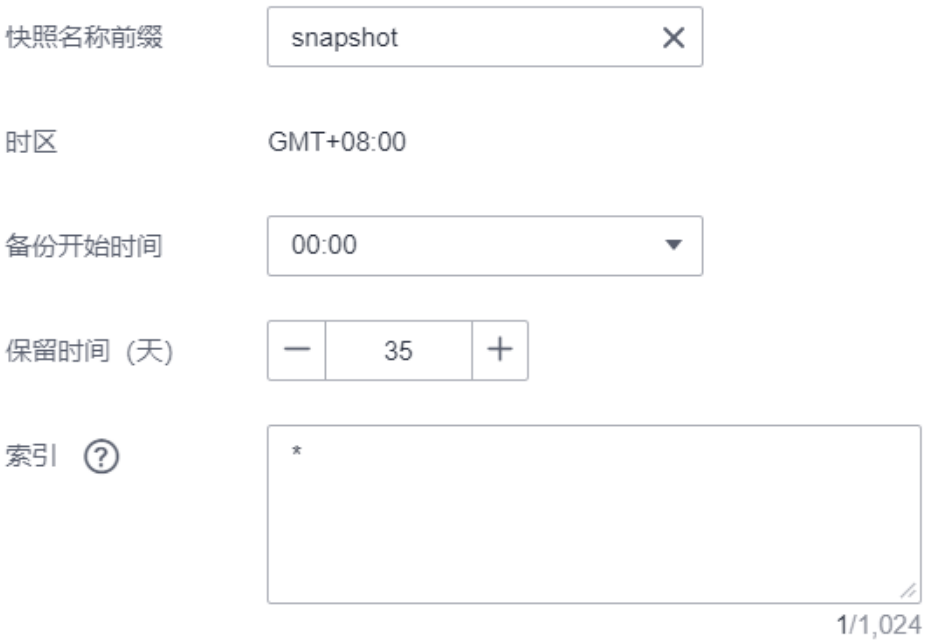
表 7-4 集群快照基础配置的参数说明

参数	说明
OBS桶	在下拉框中选择存储快照的OBS桶。也可以单击右侧的“创建桶”新建OBS。 创建或者已存在的OBS桶需满足如下条件： ● “存储类别”为“标准存储”。 ● CSS不支持加密的OBS桶，选择快照存储的OBS桶时，确保此OBS桶的加密功能关闭。
备份路径	快照在OBS桶中的存放路径。 备份路径配置规则： ● 备份路径不能包括下列符号：\:*?"<>| ● 备份路径不能以“/”开头。 ● 备份路径不能以“.”开头或结尾。 ● 备份路径的总长度不能超过1023个字符。
IAM委托	指当前帐号授权云搜索服务访问或维护存储在OBS中数据。也可以单击右侧的“创建委托”新建IAM委托。 创建或者已存在的IAM委托需满足如下条件： ● “委托类型”选择“云服务”。 ● “云服务”选择“Elasticsearch”或者“云搜索服务 CSS”。 ● 设置当前委托具备“全局服务”中“对象存储服务”项目的“Tenant Administrator”权限。

5. 在“自动创建快照”右侧，单击开关开启自动创建快照功能，弹出“创建快照策略”页面。若已启用自动创建快照功能，也可以在开关右侧单击进行快照策略修改。
   - “快照名称前缀”：快照名称前缀的长度为1～32个字符，只能包含小写字母、数字、中划线和下划线，且必须以小写字母开头。快照名称由快照名称前缀加上时间组成，例如自动生成的快照名称snapshot-2018022405925。
   - “时区”：指备份时间对应的时区。请基于此时区选择“备份开始时间”。
   - “索引”：填写索引名称，支持选择索引进行备份。索引名称不能包含空格和大写字母，且不能包含"\<|>/?特殊字符，多个索引之间使用英文逗号隔开。如果不填写，则默认备份集群中所有索引。支持使用“*”匹配多个索引，例如：index*，表示备份名称前缀是index的所有索引的数据。

     在Kibana中使用**GET /_cat/indices**命令，可以查询集群中所有索引的名称。
   - “备份开始时间”：指每天自动开始备份的时间，只能指定整点时间，如00:00、01:00，取值范围为00:00～23:00。请在下拉框中选择备份时间。
   - “保留时间（天）”：指备份的快照在OBS的保留时间，以天为单位，取值范围为1～90，您可以根据自己的需求进行设置。系统在半点时刻会自动删除超过保留时间的快照。

图 7-1 自动创建快照



6. 设置完成后，单击“确定”保存快照策略。

   按照策略自动创建的快照将呈现在快照管理列表中。快照列表同时展示自动创建和手动创建的快照，您可以通过快照类型参数进行区分。在快照列表右上角，您可以输入快照名称或快照ID的关键字进行查找。

7. （可选）关闭自动创建快照功能。

   关闭自动创建快照功能后，系统将停止继续自动创建快照。如果系统正在根据策略自动创建快照，而快照列表还未呈现正在创建的快照时，无法关闭自动创建快照功能。如果您单击了关闭按钮，系统将提示您无法关闭。建议等快照自动创建成功后，即快照列表已出现最新创建的快照时，再单击关闭按钮，关闭自动创建快照功能。

   关闭自动创建快照功能时，您可以在弹出窗口中通过“删除自动创建的快照”选项，选择是否立即删除之前已自动创建的快照，默认不勾选。

   - 不勾选：表示不会删除关闭此功能前已自动创建的快照。如果不删除，后续还可以在快照列表中通过删除按钮手动删除，详细操作指导请参见**删除快照**。如果未手动删除，且之后用户又重新开启了自动创建快照功能，那么此集群中所有“快照类型”为自动创建的快照（包含开启自动创建快照功能前已存在的自动创建的快照）都无法手动删除，只会被系统自动删除。系统会基于重新开启自动创建快照功能时的配置策略进行自动删除，例如此策略中定义的保留时间为10天，那么系统中超过10天的快照将被系统自动删除。
   - 勾选：表示删除此集群快照列表中所有“快照类型”为自动创建的快照。

手动创建快照

1. 在云搜索服务管理控制台，单击左侧导航栏的“集群管理”。
2. 在“集群管理”页面，单击需要进行备份的集群名称，进入集群基本信息页面。在左侧导航栏选择“集群快照”，进入“集群快照”管理页面。
3. 在“集群快照”管理页面，单击“集群快照开关”右侧的开关，打开集群快照功能。

4. 打开集群快照功能后，云搜索服务会自动为客户创建OBS桶和IAM委托，用于存储快照。自动创建的OBS桶和IAM委托将直接展示在界面中。如果您不希望使用自动创建的OBS桶和IAM委托，您可以在“基础配置”右侧单击进行配置。

表 7-5 集群快照基础配置的参数说明

参数	说明
OBS桶	在下拉框中选择存储快照的OBS桶。也可以单击右侧的“创建桶”新建OBS。 创建或者已存在的OBS桶需满足如下条件： ● “存储类别”为“标准存储”。 ● CSS不支持加密的OBS桶，选择快照存储的OBS桶时，确保此OBS桶的加密功能关闭。
备份路径	快照在OBS桶中的存放路径。 备份路径配置规则： ● 备份路径不能包括下列符号：\:*?"<>| ● 备份路径不能以“/”开头。 ● 备份路径不能以“.”开头或结尾。 ● 备份路径的总长度不能超过1023个字符。
IAM委托	指当前帐号授权云搜索服务访问或维护存储在OBS中数据。也可以单击右侧的“创建委托”新建IAM委托。 创建或者已存在的IAM委托需满足如下条件： ● “委托类型”选择“云服务”。 ● “云服务”选择“Elasticsearch”或者“云搜索服务 CSS”。 ● 设置当前委托具备“全局服务”中“对象存储服务”项目的“Tenant Administrator”权限。

5. 完成基础配置后，单击“创建快照”可手动创建快照。
   - “快照名称”：手动创建的快照名称，4～64个字符，只能包含小写字母、数字、中划线和下划线，且必须以字母开头。与自动创建不同，手动创建的快照名称按照用户设置的名称，不会自动加上时间信息。
   - “索引”：填写索引名称，支持选择索引进行备份。索引名称不能包含空格和大写字母，且不能包含"\<|>/?特殊字符，多个索引之间使用英文逗号隔开。如果不填写，则默认备份集群中所有索引。支持使用“*”匹配多个索引，例如：index*，表示备份名称前缀是index的所有索引的数据。

     在Kibana中使用**GET /_cat/indices**命令，可以查询集群中所有索引的名称。
   - “快照描述”：创建的快照描述信息。0～256个字符，不能包含“<>”字符。

图 7-2 手动创建快照

6. 单击“确定”开始创建快照。

   快照创建完成后，将直接呈现在快照管理列表中，快照状态为“可用”表示快照创建成功。快照列表同时展示自动创建和手动创建的快照，您可以通过快照类型参数进行区分。在快照列表右上角，您可以输入快照名称或快照ID的关键字进行查找。

恢复数据

快照管理列表中“快照状态”为“可用”的快照，可以恢复集群中的数据。已存储的快照数据可恢复至其他集群。

恢复数据将覆盖集群中当前的数据，请谨慎操作。

1. 在快照管理列表中，选择需要恢复的快照，单击“操作”列的“恢复”。

   图 7-3 选择恢复快照

2. 在“恢复”页面配置集群的恢复参数。

   “索引”：指定需要进行恢复的索引名称，默认为空。如保持默认值，即不指定索引名称，则表示恢复所有的索引数据。0～1024个字符，不能包含空格和大写字母，且不能包含"\<|>/?特殊字符。支持使用“*”匹配多个索引，比如index*，表示恢复快照中名称前缀是index的所有索引。

   “索引名称匹配模式”：在恢复时，可以根据文本框中定义的过滤条件去恢复符合条件的索引，过滤条件请使用正则表达式。默认值“index_(.+)”表示所有的索引。0～1024个字符，不能包含空格和大写字母，且不能包含"\<|>/?,特殊字符。

   “索引名称替换模式”：索引重命名的规则。默认值“restored_index_$1”表示在所有恢复的索引名称前面加上“restored_”。0～1024个字符，不能包含空格和大写字母，且不能包含"\<|>/?,特殊字符。

说明

“索引名称匹配模式”和“索引名称替换模式”需要同时设置才会生效。

“集群”：选择需要进行恢复的集群名称，可选择当前集群或者其他集群。只能选择处于“可用”状态的集群，如果快照所属的集群处于“不可用”状态，那么也无法将快照恢复到本集群。恢复到其他集群时，目标集群中的Elasticsearch版本不低于本集群。如果已选择其他集群，且该集群中存在同名的索引，则恢复完成后，该同名的索引中的数据将会被覆盖，请谨慎操作。

图 7-4 恢复快照

3. 单击“确定”开始恢复。恢复成功，快照列表中“任务状态”将变更为“恢复成功”，索引数据将根据快照信息重新生成。

图 7-5 恢复成功

删除快照

当快照信息不需要使用时，您可以删除快照释放存储资源。当自动创建快照功能开启时，自动创建的快照无法手动删除，系统会按照设置的策略在半点时刻自动删除超过“保留时间”的快照。当自动创建快照功能关闭，且之前已自动创建的快照并未同步删除时，快照列表中自动创建的快照，可通过删除按钮手动删除。如果未手动删除，且之后用户又重新开启了自动创建快照功能，那么此集群中所有“快照类型”为自动创建的快照（包含开启自动创建快照功能前已存在的自动创建的快照）都无法手动删除，只会被系统自动删除。

说明

快照信息删除后，数据将无法恢复，请谨慎操作。

1. 在快照管理页面中，选择需要删除的快照.

2. 单击“操作”列的“删除”，在弹窗中确认要删除的快照信息后，单击“确定”删除快照。

7.4 绑定企业项目

企业可以根据组织架构规划企业项目，将企业分布在不同区域的资源按照企业项目进行统一管理，同时可以为每个企业项目设置拥有不同权限的用户组和用户。本章节为您介绍CSS集群如何绑定、修改企业项目。

前提条件

在绑定企业项目前，您已在“企业项目管理控制台”创建企业项目。

绑定企业项目

在创建集群时，可以在“企业项目”绑定已创建的企业项目，也可以单击“查看项目管理”，前往企业项目管理管理控制台，新建企业项目和查看已有的企业项目。

修改企业项目

针对之前已创建的集群，其绑定的企业项目可根据实际情况进行修改。

1. 登录在云搜索服务管理控制台，
2. 在左侧导航栏，选择“集群管理 > Elasticsearch”，进入集群管理页面。
3. 在集群列表中，单击集群名称进入集群“基本信息”页面。
4. 在集群“基本信息”页面，单击“企业项目”右侧的企业项目名称，进入项目管理页面。

图 7-6 进入企业项目管理页面

5. 在“资源”页签下，“区域”选项中选择当前集群所在的区域，“服务”选项中选“云搜索服务 CSS”。此时，资源列表将筛选出对应的CSS集群。

图 7-7 筛选 CSS 集群

6. 勾选需要修改企业项目的集群，然后单击“迁出”。
7. 在“迁出资源”页面，选择“迁出方式”，再选择“请选择要迁入的企业项目”，然后单击“确定”。

图 7-8 迁出资源

8. 迁出完成后，可以在云搜索服务管理控制台集群管理页面，查看修改后的集群企业项目信息。

7.5 重启集群

集群停止工作时，您可通过重启集群恢复运行。

前提条件

- 确认集群的“任务状态”没有正在执行中的任务，且集群未被冻结。
- 当集群处于可用状态时，确认集群已停止处理业务数据（如导入数据、搜索数据），否则重启集群时可能导致数据丢失等。建议在业务空闲时操作。

背景信息

重启集群支持快速重启和滚动重启。

快速重启

- 所有集群都支持。
- 当选择“节点类型”快速重启时，所选类型的所有节点会一起重启。
- 当选择“节点名称”快速重启时，一次只能重启一个节点。
- 快速重启过程中，集群不可用。

滚动重启

- 仅当集群的节点数量（含Master节点、Client节点和冷数据节点）大于等于3时，才支持滚动重启。
- 滚动重启只支持根据“节点类型”进行重启。选择节点类型滚动重启时，所选类型的节点会依次重启。
- 滚动重启过程中，只有正在重启的节点不可用，不在重启过程中的节点可以正常提供服务。
- 当数据量比较大时，滚动重启耗时较长。

快速重启

1. 登录云搜索服务管理控制台。
2. 在左侧导航栏，选择“集群管理 > Elasticsearch”，在对应集群的“操作”列中单击“更多>重启”。
3. 在“重启集群”页面，选择“快速重启”。

   快速重启支持根据“节点类型”或者“节点名称”重启。如果选择“节点类型”，则支持选择多种节点类型同时进行快速重启。如果选择“节点名称”，则一次只能快速重启一个节点。
4. 重启集群后，请刷新页面，观察集群状态。重启过程中，集群状态为“处理中”，任务状态为“重启中”。如果集群状态变更为“可用”，表示集群已重启成功。

滚动重启

1. 登录云搜索服务管理控制台。
2. 在左侧导航栏，选择“集群管理 > Elasticsearch”，在对应集群的“操作”列中单击“更多>重启”。
3. 在“重启集群”页面，选择“滚动重启”。

   滚动重启支持根据“节点类型”进行重启。如果只需要重启集群中的某些类型的节点时，可以选择需要重启的节点类型。
4. 重启集群后，请刷新页面，观察集群状态。重启过程中，集群状态为“处理中”，任务状态为“重启中”。如果集群状态变更为“可用”，表示集群已重启成功。

7.6 迁移集群

将一个集群的数据迁移到另一个集群，称之为集群迁移。集群迁移的应用场景很多，如当业务数据不断增加时，无法直接修改当前集群的规格以便满足需求时，可以选择创建一个规格较高的集群，然后通过集群迁移的操作，快速将数据全部迁移至新集群中，以满足业务需求。另一个场景，如通过集群迁移可将两个集群的索引合并到一个集群中，以满足业务的需要。在云搜索服务中，通过备份与恢复索引功能可实现集群迁移，即将一个集群的快照恢复到另一个集群。

本文以将集群“Es-1”中的数据迁移到集群“Es-2”为例。其中“Es-2”集群的版本高于“Es-1”集群，且节点数要高于“Es-1”节点数的1/2。

迁移条件

- 原集群和目标集群在同一个region下。
- 目标集群的版本等于或高于原集群。
- 目标集群节点数要大于原集群节点数的一半。

迁移建议

- 目标集群的节点数不少于原集群的shard副本数。
- 目标集群的CPU、MEM和Disk配置大于等于原集群，使迁移后业务受损最小化。

迁移时长

迁移过程的耗时长短依赖于源集群和目的集群的节点个数或索引shard个数。迁移过程分为备份阶段和恢复阶段，备份阶段耗时由源集群决定，恢复阶段耗时由目的集群决定。迁移总时长的评估公式如下：

- 当索引shard个数大于节点个数时

  $$总时长(s)=(800G\div40MB\div源集群节点个数+800G\div40MB\div目的集群节点个数)\times索引个数$$

- 当索引shard个数小于节点个数时

  $$总时长(s)=(800G\div40MB\div源集群索引shard个数+800G\div40MB\div目的集群索引shard个数)\times索引个数$$

说明

评估公式是基于理想状态下（即单节点以最快速度40MB/s传输）的迁移时长，实际迁移时长还会受到网络、资源等因素影响。

操作步骤

1. 在集群管理界面中，单击集群名称“Es-1”进入集群“基本信息”页面。
2. 在左侧导航栏，选择“集群快照”页签，打开集群快照开关，完成基础配置。详细请参见**手动创建快照**。
3. 单击“创建快照”手动创建快照，在弹出框中输入快照名称并单击“确定”，等待快照创建完成。
4. 快照创建完成后，在快照管理页面，单击该快照操作列的“恢复”按钮，将数据恢复至Es-2集群。
   - 在“索引”的文本框中输入“*”，表示对集群“Es-1”的全部索引进行恢复。
   - 在“集群”的下拉框中选择“Es-2”，将该快照恢复到集群“Es-2”中。

   最后单击“确定”按钮开始恢复。
5. 恢复完成后，即完成了集群“Es-1”中的数据到集群“Es-2”的迁移。

7.7 删除集群

当用户已完成数据搜索业务，无需继续使用某一集群时，可删除集群释放资源。

说明

- 删除集群时，会清理集群业务数据，请谨慎操作。
- 如果集群启用过快照功能，且OBS桶中创建的快照并未被删除，删除集群时，并不会释放这部分备份数据。

操作步骤

1. 登录云搜索服务管理控制台。
2. 在左侧导航栏，选择“集群管理 > Elasticsearch”进入集群列表界面。
3. 在对应集群的“操作”列中单击“更多>删除”。
4. 在弹出的确认提示框中，输入需要删除的集群名称，单击“确定”完成集群删除。

7.8 标签管理

标签是集群的标识。为集群添加标签，可以方便用户识别和管理拥有的集群资源。

您可以在创建集群时添加标签，也可以在集群创建完成后，在集群的详情页添加标签。

新建集群的标签管理

1. 登录云搜索服务管理控制台。
2. 单击右上角的“创建集群”，进入创建集群页面。
3. 在创建集群页面，“高级配置”选择“自定义”后，为集群添加标签。

   您可以选择预定义标签，并为此标签设置“标签值”。您可以单击“查看预定义标签”，进入“标签管理服务”，了解此用户下已有的标签。

   您也可以自定义“标签键”和“标签值”。

图 7-9 创建集群时添加标签

云搜索服务的每个集群最多可以设置10个标签。当设置不正确时，可单击标签右侧的“删除”按钮，删除此标签。当不设置标签时，可保持为空。

表 7-6 标签命名规则

参数	说明
标签键	对于同一个集群，标签键值唯一。 长度不超过36个字符。 只能包含数字、英文字母、下划线、中划线。

参数	说明
标签值	长度不超过43个字符。 只能包含数字、英文字母、下划线、中划线。 不能为空。

已有集群的标签管理

您可以对已经创建的集群的标签进行修改，删除，也可以添加标签。

1. 登录云搜索服务管理控制台。
2. 在集群管理页面，单击待管理标签的集群名称。

   系统跳转至该集群“基本信息”页面。
3. 左侧菜单栏选择“标签”，在此可以对集群标签进行添加，修改，删除操作。
   - 查看

     在“标签”页，可以查看当前集群的标签详情，包括标签个数，以及每个标签的键和值。
   - 添加

     单击左上角的“添加标签”，在弹出的“添加标签”窗口，输入新添加标签的键和值，并单击“确定”。
   - 修改

     只能修改已有标签的标签值。

     单击标签所在行“操作”列下的“编辑”，在弹出的“编辑标签”窗口，输入修改后标签值，并单击“确定”。
   - 删除

     单击标签所在行“操作”列下的“删除”，如果确认删除，在弹出的“删除标签”窗口，单击“确定”。

通过标签搜索集群

1. 登录云搜索服务管理控制台。
2. 在集群管理页面，单击集群列表右上角的“标签搜索”。
3. 选择或输入需要搜索的标签键和标签值，单击“添加”将标签加入搜索输入框中。

   标签键和标签值仅支持从下拉列表中选择，当标签键和标签值全匹配时，系统可以自动查询到目标集群。当有多个标签条件时，会取各个标签的交集，进行集群查询。

   系统最多支持10个不同标签的组合搜索。
4. 单击“搜索”。

   系统根据标签键和标签值搜索目标集群。

7.9 公网访问

针对启用HTTPS访问的安全集群（6.5.4及之后版本的集群支持开启“安全模式”），云搜索服务的集群支持配置公网访问，配置完成后，通过提供的公网IP，您可以在外网接入安全集群。

说明

CSS开启公网访问后，会使用到EIP和带宽资源，涉及相关资源费用。

创建集群时配置公网访问

1. 登录云搜索服务管理控制台。
2. 在创建集群页面，开启“安全模式”。设置管理员密码，并启用HTTPS访问。
3. “公网访问”选择“自动绑定”，配置公网访问相关参数。

图 7-10 创建集群时配置公网访问

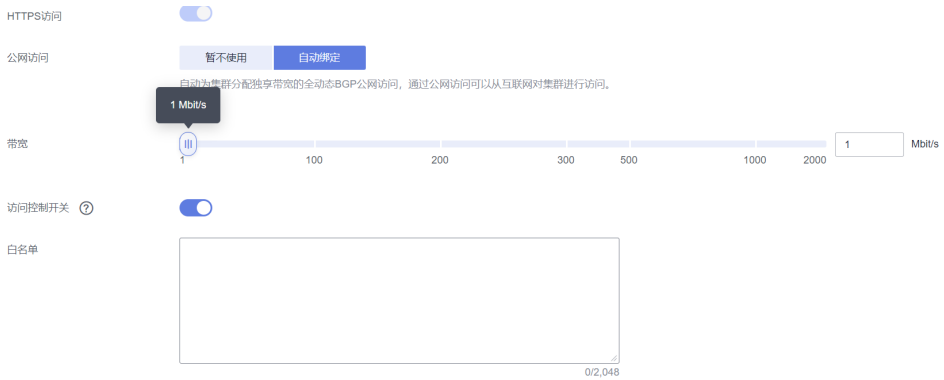


表 7-7 公网访问参数说明

参数	说明
带宽	设置公网访问的带宽。
访问控制开关	如果关闭访问控制开关，则允许任何IP通过公网IP访问集群。如果开启访问控制开关，则只允许白名单列表中的IP通过公网IP访问集群。
白名单	设置允许访问的IP地址或网段，中间用英文逗号隔开。仅当打开“访问控制开关”时才需要配置。

已有集群公网访问管理

您可以对已经创建集群的公网访问进行修改，查看，解绑，也可以配置公网访问。

1. 登录云搜索服务管理控制台。
2. 在集群管理页面，单击需要配置公网访问的集群名称，进入集群基本信息页面，管理公网访问相关配置。

图 7-11 修改公网访问相关配置

- 配置公网访问

  如果创建安全集群时，开启了HTTPS访问但未配置公网访问，集群创建成功后，可以在集群基本信息页面配置公网访问。

  单击“公网访问”参数右侧的“绑定”，设置访问带宽后，单击“确定”。

  如果绑定失败，用户可以等待几分钟后，再次尝试重新绑定公网访问。

- 修改

  对已经配置了公网访问的集群，可以通过单击“带宽”参数右侧的“修改”，修改带宽大小，也可以通过单击“访问控制”右侧的“设置”，设置访问控制开关和访问白名单。

- 查看

  在“基本信息”页面，可以查看当前集群绑定的公网IP地址。

- 解绑

  对于已经绑定的公网IP，可以通过单击“公网访问”参数右侧的“解绑”，解绑公网IP。

通过公网 IP 接入集群

公网访问配置完成后，集群将会获得一个“公网访问”的IP地址，用户可以通过公网IP地址和端口接入集群。

例如，查看集群中的索引信息，集群中某一个节点的公网访问地址为“10.62.179.32”，端口为“9200”，使用curl执行如下命令。

- 如果接入集群未启用安全模式，接入方式为：

  ```
  curl 'http://10.62.179.32:9200/_cat/indices'
  ```

- 如果接入集群已启用安全模式，则需要使用https方式访问，并附加用户名和密码，在curl命令中添加-u选项。

  ```
  curl -u username:password -k 'https://10.62.179.32:9200/_cat/indices'
  ```

7.10 日志管理

为了方便用户使用日志定位问题，云搜索服务提供了日志备份和日志查询功能。用户可以将集群的日志备份在OBS桶中，然后通过OBS可以直接下载需要的日志文件，进行问题分析定位。

开启日志管理

1. 登录云搜索服务管理控制台。

2. 在“集群管理”页面，单击需要配置日志备份的集群名称，进入集群基本信息页面。
3. 左侧导航栏，选择“日志管理”，在“日志管理开关”右侧单击开关，打开集群的日志管理功能。
4. 在“编辑日志备份配置”弹窗中，完成参数配置。

   弹窗中默认填写了云搜索服务自动为用户创建的“OBS桶”、“备份路径”和“IAM委托”，用于日志备份。支持用户参考**表7-8**修改默认值。

   如果集群已经启用了日志管理功能，也可以单击“日志备份配置”右侧的，在“编辑日志备份配置”窗口，参考**表7-8**更新日志备份的配置参数。

   表 7-8 日志备份配置的参数说明

参数	说明	注意事项
“OBS桶”	选择日志存储的OBS桶。单击右侧的“创建桶”支持新建OBS桶。	OBS桶的所在区域必须跟集群的所在区域保持一致。 **说明** • CSS不支持加密的OBS桶，选择日志存储的OBS桶时，确保此OBS桶的加密功能关闭 • 如果是子帐号，需要同时设置GetBucketStoragePolicy、GetBucketLocation、ListBucket、ListAllMyBuckets权限，才能看到OBS桶。
“备份路径”	填写日志在OBS桶中的存放路径。	备份路径配置规则： • 备份路径不能包括下列符号：\:*?"<>| • 备份路径不能以“/”开头。 • 备份路径不能以“.”开头或结尾。 • 备份路径的总长度不能超过1023个字符。
“IAM委托”	选择IAM委托，指当前帐号授权云搜索服务访问或维护存储在OBS中数据。单击右侧的“创建委托”支持新建委托。	IAM委托需满足如下条件： • “委托类型”选择“云服务”。 • “云服务”选择“Elasticsearch”或者“云搜索服务 CSS”。 • 必选策略：“Tenant Administrator”

5. 日志备份。
   - 自动备份日志。

     在“自动备份开关”右侧，单击开关，开启自动备份日志功能。

     开启“自动备份开关”后，在“修改日志备份策略”弹窗中设置“备份开始时间”。设置成功后，系统会按照设置的时间进行自动备份。

     打开“自动备份开关”后，单击开关右侧的，可以修改“备份开始时间”。

- 手动备份日志。

  单击“日志备份”下面的“开始备份”，在弹出的确认提示框中，单击“确定”，开始备份日志。

  日志备份列表中的“任务状态”为“Successful”时，表示日志备份成功。

  说明

  云搜索服务会把集群中当前的所有日志全部复制到指定的OBS路径中，用户可以在自己的OBS桶对应的路径中直接查看或者下载日志文件。

6. 日志查询。

   在“日志查询”页面，选择需要查询的节点、日志类型和日志级别信息后，单击  ，显示查询结果。

   查询日志时，是从最近时刻的1万条日志中进行匹配，查询结果最多显示100条。

日志信息

日志备份成功后，用户可以单击“OBS桶”，进入到OBS控制台，找到备份路径查看备份的日志信息。

图 7-12 进入 OBS

云搜索服务备份的日志信息主要包括废弃操作日志、运行日志、慢索引日志、慢查询日志。在OBS桶中的存储类型如表7-9所示。

表 7-9 日志类型信息

日志名称	描述
clustername_deprecation.log	弃用操作的日志记录。
clustername_index_indexing_slowlog.log	慢索引日志。

日志名称	描述
clustername_index_search_slowlog.log	慢索引查询日志。
clustername.log	Elasticsearch运行日志。
clustername_access.log	接入日志。
clustername_audit.log	审计日志。

7.11 插件管理

云搜索服务的Elasticsearch集群自带系统默认插件。可以通过控制台查看或在Kibana查询系统默认插件信息。

通过控制台查看

1. 登录云搜索服务管理控制台。
2. 在集群管理页面，单击需要查看插件的集群名称，跳转至该集群基本信息页面。
3. 选择“插件管理”。
4. 在“系统默认插件列表”页查看当前版本支持的系统默认插件信息。

在 Kibana 查询

1. 登录云搜索服务控制台。
2. 在集群管理列表，选择需要查看插件的集群，单击操作列的“Kibana”登录Kibana界面。
3. 进入Dev Tools，执行如下命令查看集群插件信息：

   ```
   GET _cat/plugins?v
   ```

   响应体示例如下：

   ```
   name                component                      version
   css-test-ess-esn-1-1 analysis-dynamic-synonym      7.6.2-xx-ei-css-v1.0.1
   css-test-ess-esn-1-1 analysis-icu                  7.6.2-xx-ei-css-v1.1.6
   css-test-ess-esn-1-1 analysis-ik                   7.6.2-xx-ei-css-v1.0.1
   ......
   ```

   “name”是集群的节点名称，“component”是插件名称，“version”是插件版本。

7.12 冷热数据存储

云搜索服务提供了冷数据节点供企业选择，企业可以将部分现查要求秒级返回的数据放在高性能机器上面，对于历史数据要求分钟级别返回的数据放在大容量低规格节点。

说明

- 创建集群时，数据节点为必选，选择冷数据节点后，其他数据节点变为热节点。
- 选择冷数据节点的同时，支持独立选择Master和Client节点。
- 冷数据节点支持节点和磁盘扩容，前提是冷节点规格支持（本地盘不支持磁盘扩容）。

冷热数据切换

选择冷数据节点后，冷数据节点将会打上cold”标签，用来表示冷节点。同时，数据节点将会上升为热节点，会被打上“hot”标签。用户可以通过配置指定索引，将数据分配到冷热节点。

通过设置template，可以通过模板将相应的index存储到指定冷热节点。

如下，登录集群的Kibana Console页面，配置myindex开头的索引，储存在冷节点上面。 这样可以通过模板在创建的时候把myindex*的数据存储在冷数据节点上面。

6.x及以上版本使用以下命令创建模板：

```
PUT _template/test
{
  "order": 1,
  "index_patterns": "myindex*",
  "settings": {
    "refresh_interval": "30s",
    "number_of_shards": "3",
    "number_of_replicas": "1",
    "routing.allocation.require.box_type": "cold"
  }
}
```

同时也可以单独对已经创建好的索引进行操作。

```
PUT myindex/_settings
 {
       "index.routing.allocation.require.box_type": "cold"
   }
```

也可以去掉冷热数据配置，不受冷热数据标签影响。

```
PUT myindex/_settings
{
       "index.routing.allocation.require.box_type": null
   }
```

7.13 参数配置

云搜索服务支持用户修改elasticsearch.yml文件。

修改参数配置

1. 登录云搜索服务管理控制台。
2. 在“集群管理”页面，单击需要修改参数配置的集群名称，进入集群基本信息页面。
3. 选择“参数配置”，单击“编辑”，根据需求修改对应模块的参数值。

表 7-10 模块参数信息说明

模块名称	参数名称	说明
跨域访问	http.cors.allow-credentials	跨域访问是否返回头部的Access-Control-Allow-Credentials。 取值范围：true、false。 默认值：false。
	http.cors.allow-origin	允许跨域访问的IP，配置样例如122.122.122.122:9200。
	http.cors.max-age	浏览器默认缓存时间。如果超过设置的时间后，缓存将自动清除。 单位：秒。 默认值：1728000。
	http.cors.allow-headers	跨域访问允许的headers，包括X-Requested-With, Content-Type, Content-Length，中间用英文逗号和空格分开。
	http.cors.enabled	是否允许跨域访问。 取值范围：true、false。 默认值：false。
	http.cors.allow-methods	跨域访问允许的方法，包括OPTIONS, HEAD, GET, POST, PUT, DELETE，中间用英文逗号和空格分开。
集群索引重建	reindex.remote.whitelist	配置该参数可以将本集群数据通过reindex接口迁移到配置的集群，配置样例如122.122.122.122:9200。
自定义缓存	indices.queries.cache.size	查询阶段的缓存大小。 取值范围：1-100。 单位：%。 默认值：10%。
线程池队列大小	thread_pool.bulk.queue_size	Bulk请求的队列大小。输入的参数值为整数类型。 默认值：200。
	thread_pool.write.queue_size	线程池写入队列大小。输入的参数值为整数类型。 默认值：200。
	thread_pool.force_merge.size	用来做forcemerge的队列大小。输入的参数值为整数类型。 默认值：1。

模块名称	参数名称	说明
自定义	用户可以根据实际情况，添加相关参数名称。	自定义参数的取值。 **说明** • 如果自定义参数有多个取值，则取值的输入格式为[value1, value1, value1...]。 • 取值之间用英文逗号和空格隔开。 • 自定义参数值中不能包含冒号。

4. 修改完成后，单击上方的“提交”弹出“提交配置”窗口，确认参数无误后勾选“参数修改后需要手动重启才能生效”，单击“确定”。

   当下方的参数修改列表显示“作业状态”为“成功”时，表示修改保存成功。系统最多显示20条修改记录。

5. 返回集群列表，单击集群操作列的“更多 > 重启”重启集群，使修改的配置生效。
   - 如果修改了参数配置，未重启集群，则在“集群管理”页面的“任务状态”栏显示为“配置未更新”。
   - 如果修改后重启集群，“任务状态”显示“配置错误”，则表示修改参数配置文件失败。

7.14 终端节点服务

云搜索服务提供了终端节点服务，用户开启了此服务后，可以通过节点IP或内网域名访问集群。在开启终端节点服务时，系统会默认给用户创建一个终端节点，内网域名由用户自己选择是否创建。

⚠ 注意

公网访问和终端节点服务功能使用的是同一个负载均衡。如果开启了公网访问白名单，由于白名单是作用在负载均衡上面，会同时限制公网访问集群和内网通过VPCEP访问集群的IP。此时需要在公网访问白名单中添加一个网络白名单198.19.128.0/17，该白名单用来放通经过VPCEP的流量。

创建集群时开启终端节点服务

1. 登录云搜索服务管理控制台。
2. 在右上方单击“创建集群”。
3. 在创建集群页面，“高级配置”选择“自定义”后，开启终端节点服务。

图 7-13 开启终端节点服务

- “创建内网域名”：如果开启，系统将会自动为用户创建一个内网域名，可以通过内网域名访问集群。
- “终端节点服务白名单”：您可以在“终端节点服务白名单”中添加需要授权的帐号ID，只要其帐号ID被添加到终端节点服务白名单中，，就可以通过内网域名或者节点IP访问集群。
- 单击“添加”可以添加多个帐号。
- 单击“操作”列的“删除”，可以删除不允许访问的帐号。

说明

- 授权帐号ID配置成*，则表示允许全部用户访问该集群。
- 需要授权的帐号ID可在“我的凭证”中进行查看。

已有集群终端节点服务管理

如果创建集群时未开启终端节点服务，集群创建成功后，可以通过如下步骤进行开启。

1. 登录云搜索服务管理控制台。
2. 在集群管理页面，单击需要开启终端节点服务的集群名称，进入集群基本信息页面。
3. 选择“终端节点服务”，在“终端节点服务”右侧单击开关，打开集群的终端节点服务功能。

   在弹出的提示框中，您可以根据需求，选择是否创建内网域名。单击“是”，开启终端节点服务。

   说明

   - 开启终端节点服务后，您可以通过终端节点产生的“内网域名”或者“节点IP”访问此集群。详细请参考**通过内网域名或节点IP访问集群**。
   - 关闭终端节点服务功能后，所有的用户将不能通过内网域名访问此集群。
4. （可选）打开终端节点服务后，您可以单击“终端节点服务白名单”后面的“修改”，更新已有的白名单。
5. 管理终端节点。

   在终端节点服务页面下，显示所有连接当前终端节点服务的终端节点。

图 7-14 管理终端节点

- 单击操作列的“接受”或者“拒绝”可以修改节点的“状态”。如果对某个终端节点“拒绝”操作之后，其生成的内网域名将不能再访问到当前集群。
- 单击操作列的“修改”，可以更改当前节点的“最大连接数”。

通过内网域名或节点 IP 访问集群

1. 获取内网域名或者节点IP。

   登录**云搜索服务**控制台，进入集群列表，单击集群名称，进入集群“基本信息”页面，选择“终端节点服务”，查看内网域名。

   图 7-15 查看节点 IP 和内网域名信息

2. 在弹性云服务器中，直接通过curl执行API或者开发程序调用API并执行程序即可使用集群。Elasticsearch操作和接口请参见《**Elasticsearch：权威指南**》。

   弹性云服务器需要满足如下要求：

   - 为弹性云服务分配足够的磁盘空间。
   - 此弹性云服务器的VPC需要与集群在同一个VPC中，开通终端节点服务后，可以实现跨VPC访问。
   - 此弹性云服务器的安全组需要和集群的安全组相同。

     如果不同，请修改弹性云服务器安全组或配置弹性云服务器安全组的出入规则允许集群所有安全组的访问。修改操作请参见《虚拟私有云用户指南》。
   - 待接入的CSS集群，其安全组的出方向和入方向需允许TCP协议及9200端口，或者允许端口范围包含9200端口。

   例如，使用curl执行如下命令，查看集群中的索引信息，集群中的内网访问地址为“vpcep-7439f7f6-2c66-47d4-b5f3-790db4204b8d.region01.xxxx.com”，端口为“9200”。

   - 如果接入集群未启用安全模式，接入方式为：

     ```
     curl 'http://vpcep-7439f7f6-2c66-47d4-b5f3-790db4204b8d.region01.xxxx.com:9200/_cat/indices'
     ```
   - 如果接入集群已启用安全模式，则需要使用https方式访问，并附加用户名和密码，在curl命令中添加-u选项。

     ```
     curl -u username:password -k 'https://vpcep-7439f7f6-2c66-47d4-b5f3-790db4204b8d.region01.xxxx.com:9200/_cat/indices'
     ```

7.15 Kibana 公网访问

针对安全模式集群，云搜索服务支持配置Kibana开启公网访问，配置完成后，对应集群将会获得一个Kibana公网访问地址，通过这个地址可以在公网上面访问集群的Kibana。

对于安全模式集群来说，支持在创建的时候配置Kibana公网访问，同时也支持安全模式集群创建完之后再开启Kibana公网访问。

说明

- 6.5.4及之后版本的集群支持开启“安全模式”。
- 在该特性上线之前（即2020年6月前）创建的安全模式的集群，不支持此功能。
- Kibana公网访问配置的白名单依赖ELB的白名单能力。更新白名单后，白名单对新建的连接是实时生效的，但对于已存在的长连接，可能会出现去掉的白名单IP地址还能访问Kibana的场景，这是因为要等长连接断开后才生效，预计1分钟左右。

创建集群时配置 Kibana 公网访问

1. 登录云搜索服务管理控制台。
2. 单击右上角的“创建集群”，进入创建集群页面。
3. 在创建集群页面，开启“安全模式”。
4. “高级配置”选择“自定义”后，开启Kibana公网访问，配置相关参数。

表 7-11 Kibana 公网访问参数说明

参数	说明
带宽	设置公网访问的带宽。 取值范围：1-100。 单位：Mbit/s。
访问控制开关	如果关闭访问控制开关，则允许任何IP通过公网IP访问集群Kibana。如果开启访问控制开关，则只允许白名单列表中的IP通过公网IP访问集群Kibana。
白名单	设置允许访问的IP地址或网段，中间用英文逗号隔开。仅当打开“访问控制开关”时才需要配置。 建议开启白名单。

集群创建成功后，单击集群名称，进入集群基本信息页面，在“Kibana公网访问”页签，可以查看kibana公网访问地址。

已有集群开启 Kibana 公网访问

您可以对已经创建的安全模式集群的Kibana公网访问进行开启、关闭、修改、查看等操作。

1. 登录云搜索服务管理控制台。

2. 在集群管理页面，单击需要配置Kibana公网访问的集群名称，进入集群基本信息页面。
3. 选择“Kibana公网访问”，在“Kibana公网访问”右侧单击开关，打开Kibana公网访问功能。
4. 在开启Kibana公网访问页面，配置相关参数。

   表 7-12 Kibana 公网访问参数说明

参数	说明
带宽	设置公网访问的带宽。 取值范围：1-100。 单位：Mbit/s。
访问控制开关	如果关闭访问控制开关，则允许任何IP通过公网IP访问集群Kibana。如果开启访问控制开关，则只允许白名单列表中的IP通过公网IP访问集群Kibana。
白名单	设置允许访问的IP地址或网段，中间用英文逗号隔开。仅当打开“访问控制开关”时才需要配置。 建议开启白名单。

5. 配置完成后，单击“确定”。

修改 Kibana 公网访问

对已经配置了Kibana公网访问的集群，云搜索服务支持修改带宽、修改访问控制和关闭Kibana公网访问。

1. 登录云搜索服务管理控制台。
2. 在集群管理页面，单击需要修改Kibana公网访问的集群名称，进入集群基本信息页面。
3. 选择“Kibana公网访问”，修改Kibana公网访问。
   - 修改带宽

     单击“带宽”参数右侧的“修改”，在“修改Kibana公网访问带宽”页面修改带宽大小，修改完成后，单击“确定”。
   - 修改访问控制

     单击“访问控制开关”右侧的“修改”，在“修改Kibana公网访问控制”页面设置“访问控制开关”和访问“白名单”，修改完成后，单击“确定”。
   - 关闭Kibana公网访问

     在“Kibana公网访问”右侧单击开关，确认关闭Kibana公网访问功能。

通过公网 IP 访问 Kibana

Kibana公网访问配置完成后，将会获得一个Kibana公网访问地址，用户可以通过此IP地址访问集群的Kibana。

1. 登录云搜索服务管理控制台。
2. 在集群管理页面，单击需要配置Kibana公网访问的集群名称，进入集群基本信息页面。

3. 选择“Kibana公网访问”，获取kibana公网访问地址。

图 7-16 获取 Kibana 公网访问地址

4. 通过该地址，就可以在公网上面访问云搜索服务集群的Kibana。

8 向量检索

8.1 场景描述

随着大规模图像检索、视频搜索、推荐等场景的数据规模越来越大，对高维空间向量检索的时延和准确率提出了更高的要求。云搜索服务针对大规模的向量检索场景提供了具体的解决方案，基于云自研的向量搜索引擎，结合Elasticsearch的插件机制，高效集成了向量检索能力。

原理

向量检索从本质上讲，其思维框架和传统的检索方法没有区别。为了提升向量检索的性能，通常需要解决以下两个问题：

- **减少候选向量集**

  和传统的文本检索类似，向量检索也需要某种索引结构来避免在全量的数据上做匹配，传统文本检索是通过倒排索引来过滤掉无关文档，而向量检索是通过对向量建立索引结构来绕过不相关的向量，减小需要考察的范围。

- **降低单个向量计算的复杂度**

  向量检索支持漏斗模型，先对所有向量进行量化和近似计算，筛选出一定量接近检索目标的数据集，然后基于筛选的数据集进行精细的计算和排序。本方法不需要对所有向量都进行复杂的计算，可以有效提高检索效率。

向量检索即在一个给定的向量数据集中，按照某种度量方式，检索出与查询向量相近的K个向量（K-Nearest Neighbor，KNN），但由于KNN计算量过大，通常只关注近似近邻（Approximate Nearest Neighbor，ANN）问题。

功能

云自研向量检索引擎集成了暴力检索、图索引（HNSW）、乘积量化、IVF-HNSW等多种向量索引，支持欧式、内积、余弦、汉明等多种相似度计算方式，召回率和检索性能均优于开源引擎。能够满足高性能、高精度、低成本、多模态等多种应用场景及需求。

向量检索支持原生Elasticsearch的所有能力，包括分布式、多副本、错误恢复、快照、权限控制等；兼容所有原生Elasticsearch生态，包括集群监测工具cerebro，可视化工具kibana，实时数据采集工具logstash等；提供Python/Java/Go/C++等多种客户端语言支持。

约束限制

- 仅7.6.2和7.10.2版本的集群支持向量检索。
- 向量检索插件涉及较高的内存计算，内存要求比普通索引高，建议集群规格配置为内存优化型的计算规格。

8.2 向量检索的集群规划

向量检索的索引构建与查询均使用堆外内存，所以集群容量与索引类型、总堆外内存大小等因素相关。通过预估全量索引所需的堆外内存大小，可以选择合适的集群规格。

不同类型的索引所需堆外内存大小的预估方式不同，计算公式如下：

- **GRAPH索引**

$$mem\_needs = (dim \times dim\_size + neighbros \times 4) \times num + delta$$

说明

若有实时更新索引的需求，还需要考虑向量索引构建和自动merge所需的堆外内存开销，保守估计需要1.5~2倍*mem_needs*。

- **PQ类索引**

$$mem\_needs = frag\_num \times frag\_size \times num + delta$$

- **FALT、IVF索引**

$$mem\_needs = dim \times dim\_size \times num + delta$$

表 8-1 参数说明

参数	说明
dim	向量维度。
neighbors	图节点邻居数，默认值为64。
dim_size	每一维度值所需的字节数，默认为float类型，需要4字节。
num	向量总条数。
delta	元数据大小，该项通常可以忽略。

frag_num	量化编码时的向量分段数，创建索引时若未配置该值，则由向量维度“dim”决定。
	`if dim <= 256:`
	`  frag_num = dim / 4`
	`elif dim <= 512:`
	`  frag_num = dim / 8`
	`else :`
	`  frag_num = 64`

frag_size	量化编码时中心点编码的size，默认为1，当“frag_num”大于256时，该值等于2。

基于上述计算方法，可预估出完整向量索引所需堆外内存的大小。选择集群规格时，还需考虑每个节点的堆内存开销。

节点的堆内存分配策略：每个节点的堆内存大小为节点物理内存的一半，且最大不超过31GB。

例如，基于SIFT10M数据集创建GRAPH索引，其“dim”为“128”，“dim_size”为“4”，“neighbors”采用默认值“64”，“num”为“1000万”，将各值代入上述公式得到GRAPH索引所需堆外内存大小约为：

$$mem\_needs = (128 \times 4 + 64 \times 4) \times 10000000 \approx 7.5GB$$

同时考虑到堆内存的开销，单台“8U 16G”规格的机器可以满足该场景的需求。如果实际场景还有实时写入或更新的需求，则需要考虑申请更大的内存规格。

8.3 创建向量索引

前提条件

- 已经参考**向量检索的集群规划**完成集群创建，集群必须是7.6.2或7.10.2版本。
- 根据实际需要参考**集群高级配置**完成集群高级设置。

创建向量索引

1. 登录云搜索服务管理控制台。
2. 在“集群管理”页面，选择需要启用向量检索的集群，单击操作列“Kibana”，登录Kibana界面。
3. 单击左侧导航栏的“Dev Tools”，执行如下命令创建向量索引。

   创建一个名为“my_index”的索引，该索引包含一个名为“my_vector”的向量字段和一个名为“my_label”的文本字段。其中，向量字段创建了GRAPH图索引，并使用欧式距离作为相似度度量。

```
PUT my_index
{
  "settings": {
    "index": {
      "vector": true
    }
  },
  "mappings": {
    "properties": {
      "my_vector": {
        "type": "vector",
        "dimension": 2,
        "indexing": true,
        "algorithm": "GRAPH",
        "metric": "euclidean"
      },
      "my_label": {
        "type": "text"
      }
    }
  }
}
```

表 8-2 创建索引参数说明

类型	参数	说明
Index settings参数	vector	当需要使用向量索引加速时，需要设置该值为true。
Field mappings参数	type	字段类型，“vector”表示该字段为向量字段。
	dimension	向量数据维度。 取值范围：[1, 4096]
	indexing	是否开启向量索引加速。 可选值： • false：表示关闭向量索引加速，向量数据仅写入docvalues，只支持使用ScriptScore以及Rescore进行向量查询。 • true：表示开启向量索引加速，系统将创建额外的向量索引，索引算法由"algorithm"字段指定，写入数据后可以使用VectorQuery进行查询。 默认值：false。
	algorithm	索引算法。仅当“indexing”为“true”时生效。 可选值： • FLAT：暴力计算，目标向量依次和所有向量进行距离计算，此方法计算量大，召回率100%。适用于对召回准确率要求极高的场景。 • GRAPH：图索引，内嵌深度优化的HNSW算法，主要应用在对性能和精度均有较高要求且单shard中文档数量在千万个以内的场景。 • GRAPH_PQ：将HNSW算法与PQ算法进行了结合，通过PQ降低原始向量的存储开销，能够使HNSW轻松支撑上亿规模的检索场景。 • IVF_GRAPH：算法将IVF与HNSW结合，对全量空间进行划分，每一个聚类中心向量代表了一个子空间，极大地提升检索效率，同时会带来微小的检索精度损失。适用于数据量在上亿以上同时对检索性能要求较高的场景。 • IVF_GRAPH_PQ：PQ算法与IVF-HNSW的结合，PQ可以通过配置选择与HNSW结合和IVF结合，进一步提升系统的容量并降低系统开销，适用于shard中文档数量在十亿级别以上同时对检索性能要求较高的场景。 默认值：GRAPH。 **说明** 当选择IVF_GRAPH或者IVF_GRAPH_PQ索引时，需要额外进行预构建中心点索引以及注册等步骤，具体内容请参考**（可选）预构建与注册**。

类型	参数	说明
	表8-3	当使用向量索引加速时（即“indexing”为“true”时），为了获得更高的查询性能以及查询精度，CSS提供了与向量索引相关的可选参数配置。
	metric	计算向量之间距离的度量方式。 可选值： ● euclidean：欧式距离。 ● inner_product：内积距离。 ● cosine：余弦距离。 ● hamming：汉明距离。 默认值：euclidean

表 8-3 可选参数说明

类型	参数	说明
GRAPH类索引配置参数	neighbors	图索引中每个向量的邻居数，默认值为64，值越大查询精度越高。索引越大，构建速度以及后续的查询速度也会变慢。 取值范围：[10, 255]
	shrink	构建hnsw时的裁边系数，默认值1.0f。 取值范围：(0.1, 10)
	scaling	构建hnsw时上层图节点数的缩放比例，默认值50。 取值范围：(0, 128]
	efc	构建hnsw时考察邻居节点的队列大小，默认值为200，值越大精度越高，构建速度将会变慢。 取值范围：(0, 100000]
	max_scan_num	扫描节点上限，默认值为10000，值越大精度越高，索引速度变慢。取值范围：(0, 1000000]。
PQ类索引配置参数	centroid_num	每一段的聚类中心点数目，默认值为255。 取值范围：(0, 65535]
	fragment_num	段数，默认值为0，插件自动根据向量长度设置合适的段数。 取值范围：[0, 4096]

导入向量数据

执行如下命令，导入向量数据。向“my_index”索引中写入向量数据时，需要指定向量字段名称和向量数据。

- 向量数据输入格式为逗号分隔的浮点型数组时：

```
POST my_index/_doc
{
  "my_vector": [1.0, 2.0]
}
```

- 向量数据输入格式为小端字节序编码的Base64字符串时：

  在向量维度较高、数值有效位较多时，使用Base64编码格式传输、解析更加高效。

```
POST my_index/_doc
{
  "my_vector": "AACAPwAAAEA="
}
```

- 当写入大规模数据时，建议使用Bulk操作：

```
POST my_index/_bulk
{"index": {}}
{"my_vector": [1.0, 2.0], "my_label": "red"}
{"index": {}}
{"my_vector": [2.0, 2.0], "my_label": "green"}
{"index": {}}
{"my_vector": [2.0, 3.0], "my_label": "red"}
```

集群高级配置

- 在离线导入数据场景下，为了提高批量写入性能，建议将索引的refresh_interval参数设置为-1，即关闭自动刷新索引。
- 建议将备份数number_of_replicas设置为0，当离线数据导入完成后，再设置为需要的值。
- 其他高级功能的参数配置说明：

表 8-4 集群配置参数

参数	说明
native.cache.circuit_breaker.enabled	是否开启堆外内存熔断。 默认值：true
native.cache.circuit_breaker.cpu.limit	向量索引堆外内存使用上限。 假设使用128GB内存的机器且堆内存大小为31GB，默认堆外内存使用上限为(128 - 31) * 45% = 43.65GB，堆外内存使用量超过该值将会触发写入熔断。 默认值：45%
native.cache.expire.enabled	是否开启缓存超时设置。开启时，如果某些缓存项长时间没有被访问过将会被清除。 取值范围：true、false 默认值：false
native.cache.expire.time	超时时长。 默认值：24h
native.vector.index_threads	创建底层索引时所使用的线程数，每个shard均会使用多个构建线程。该值建议不要设置过大，避免产生过多的构建线程抢占查询资源。 默认值：4

8.4 向量查询

标准查询

针对创建了向量索引的向量字段，提供了标准向量查询语法。下述查询命令将会返回所有数据中与查询向量最近的size（topk）条数据。

```
POST my_index/_search
{
  "size":2,
  "_source": false,
  "query": {
    "vector": {
      "my_vector": {
        "vector": [1, 1],
        "topk":2
      }
    }
  }
}
```

表 8-5 标准查询的参数说明

参数	说明
vector（第一个）	表示该查询类型为VectorQuery。
my_vector	指定了需要查询的向量字段名称。
vector（第二个）	指定查询向量的具体值，支持数组形式以及Base64编码形式的输入。
topk	topk的值通常与size保持一致。
表8-6	通过调整不同索引的查询参数，可以获得更高的查询性能或者查询精度。

表 8-6 可选的查询参数说明

参数	子参数	说明
GRAPH类索引配置参数	ef	查询时考察邻居节点的队列大小。值越大查询精度越高，查询速度会变慢。默认值为200。 取值范围：(0, 100000]
	max_scan_num	扫描节点上限。值越大精度越高，查询速度变慢。默认值为10000。 取值范围：(0, 1000000]
IVF类索引配置参数	nprobe	查询考察中心点的数目。值越大精度越高，查询速度变慢。默认值为100。 取值范围：(0, 100000]

复合查询

向量检索支持与其他ES子查询组合进行复合查询，比如布尔查询、后置过滤等。

以下两个示例的查询结果：首先查询top10条与查询向量距离最近的结果，filter作为过滤条件将仅保留my_label字段为“red”的结果。

- 布尔查询示例

```
POST my_index/_search
{
  "size": 10,
  "query": {
    "bool": {
      "must": {
        "vector": {
          "my_vector": {
            "vector": [1, 2],
            "topk": 10
          }
        }
      },
      "filter": {
        "term": { "my_label": "red" }
      }
    }
  }
}
```

- 后置过滤示例

```
GET my_index/_search
{
  "size": 10,
  "query": {
    "vector": {
      "my_vector": {
        "vector": [1, 2],
        "topk": 10
      }
    }
  },
  "post_filter": {
    "term": { "my_label": "red" }
  }
}
```

ScriptScore 查询

写入向量数据后，针对向量字段可以使用ScriptScore进行最近邻查询，查询语法如下所示。

前置过滤条件可以为任意查询，script_score仅针对前置过滤的结果进行遍历，计算向量相似度并排序返回。此种查询方式的性能取决于前置过滤后中间结果集的大小，当前置过滤条件为"match_all"时，相当于全局暴力检索。

```
POST my_index/_search
 {
   "size":2,
   "query": {
   "script_score": {
       "query": {
         "match_all": {}
       },
       "script": {
         "source": "vector_score",
         "lang": "vector",
         "params": {
```



```
        "field": "my_vector",
        "vector": [1.0, 2.0],
        "metric": "euclidean"
      }
    }
  }
 }
}
```

表 8-7 script_score 参数说明

参数	说明
source	script脚本描述，使用向量相似度打分时为固定值"vector_score"。
lang	script语法描述，使用固定值"vector"。
field	向量字段名称。
vector	查询向量数据。
metric	度量方式，可选值为：euclidean、inner_product、cosine、hamming。 默认值：euclidean

重打分查询

当使用GRAPH_PQ索引或者IVF_GRAPH_PQ索引时，查询结果是根据PQ计算的非对称距离进行排序。CSS支持Rescore的方式对查询结果进行重打分精排，提升召回率。

假设my_index是PQ类型的索引，Rescore示例如下：

```
GET my_index/_search
{
  "size": 10,
  "query": {
    "vector": {
      "my_vector": {
        "vector": [1.0, 2.0],
        "topk": 100
      }
    }
  },
  "rescore": {
    "window_size": 100,
    "vector_rescore": {
      "field": "my_vector",
      "vector": [1.0, 2.0],
      "metric": "euclidean"
    }
  }
}
```


表 8-8 Rescore 参数说明

参数	说明
window_size	向量检索将会返回topk条结果，仅取前window_size条结果精排。
field	向量字段名称。
vector	查询向量数据。
metric	度量方式，可选值为：euclidean、inner_product、cosine、hamming。 默认值：euclidean

8.5 向量检索的性能调优

写入性能优化

- 关闭副本，待数据导入完成后再开启副本，减少副本构建的开销。
- 调整“refresh_interval”为120s或者更大，避免频繁刷新索引生成大量小的segments，同时减少merge带来的向量索引构建开销。
- 适当调大“native.vector.index_threads”的值（默认为4），增加向量索引构建的线程数。

```
PUT _cluster/settings
{
  "persistent": {
    "native.vector.index_threads": 8
  }
}
```

查询性能优化

- 在批量导入场景下，数据写入完成后，执行forcemerge操作能有效提升查询效率。

```
POST index_name/_forcemerge?max_num_segments=1
```

- 如果向量索引所需外内存超过了熔断线，查询时索引的缓存管理器会控制索引的换进换出，导致查询变慢，此时可适当调大熔断线的配置。

```
PUT _cluster/settings
{
  "persistent": {
    "native.cache.circuit_breaker.cpu.limit": "75%"
  }
}
```

- 如果端到端时延明显大于返回结果中的took值，说明查询的fetch阶段开销较大，可通过配置“_source”减小fdt文件的大小，从而降低fetch开销。

```
PUT my_index
{
  "settings": {
    "index": {
      "vector": "true"
    },
    "index.soft_deletes.enabled": false
  },
  "mappings": {
    "_source": {
```

```
    "excludes": ["my_vector"]
  },
  "properties": {
    "my_vector": {
      "type": "vector",
      "dimension": 128,
      "indexing": true,
      "algorithm": "GRAPH",
      "metric": "euclidean"
    }
  }
 }
}
```

8.6 （可选）预构建与注册

在**创建向量索引**时，若选择使用“IVF_GRAPH”和“IVF_GRAPH_PQ”的索引算法就需要对中心点向量进行预构建和注册。

背景信息

在向量索引加速算法中，IVF_GRAPH和IVF_GRAPH_PQ适用于超大规模场景。这两种算法需要通过对子空间的切割缩小查询范围，子空间的划分通常采用聚类或者随机采样的方式。在预构建之前，需要通过聚类或者随机采样得到所有的中心点向量。

当完成生成中心点向量的工作之后，需要对中心点向量进行预构建和注册，以实现将中心点向量预构建GRAPH或者GRAPH_PQ索引，同时注册到CSS集群内，实现在多个节点间共享此索引文件。中心点索引在shard间复用能够有效减少训练的开销、中心点索引查询次数，提升写入以及查询的性能。

操作步骤

1. 选择启用向量检索的集群，单击操作列“Kibana”，登录Kibana界面。
2. 单击左侧导航栏的“Dev Tools”，进入操作界面。
3. 创建中心点索引表。
   - 创建的索引命名为my_dict，注意该索引的number_of_shards数必须设置为1，否则无法注册。
   - 当需要使用IVF_GRAPH索引时，中心点索引的algorithm设置为GRAPH。
   - 当需要使用IVF_GRAPH_PQ索引时，中心点索引的algorithm设置为GRAPH_PQ。

```
PUT my_dict
{
  "settings": {
    "index": {
      "vector": true
    },
    "number_of_shards": 1,
    "number_of_replicas": 0
  },
  "mappings": {
    "properties": {
      "my_vector": {
        "type": "vector",
        "dimension": 2,
        "indexing": true,
        "algorithm": "GRAPH",
        "metric": "euclidean"
      }
```



```
    }
  }
}
```

4. 写入中心点向量数据。

   参考**导入向量数据**将采样或者聚类得到的中心点向量写入上述创建的my_dict索引中。

5. 调用注册接口。

   将上述创建的my_dict索引注册具有全局唯一标识名称（dict_name）的Dict对象。

```
PUT _vector/register/my_dict
{
  "dict_name": "my_dict"
}
```

6. 创建IVF_GRAPH或IVF_GRAPH_PQ索引。

   在创建IVF_GRAPH或者IVF_GRAPH_PQ索引时，不再需要指定dimension以及metric信息，只需指定之前注册好的dict名称即可。

```
PUT my_index
{
  "settings": {
    "index": {
      "vector": true
    }
  },
  "mappings": {
    "properties": {
      "my_vector": {
        "type": "vector",
        "indexing": true,
        "algorithm": "IVF_GRAPH",
        "dict_name": "my_dict",
        "offload_ivf": false
      }
    }
  }
}
```

表 8-9 Field mappings 参数

参数	说明
dict_name	指定依赖的中心点索引名称。该索引字段的向量维度和度量方式将与dict索引保持一致，不再需要额外指定。
offload_ivf	将底层索引实现的IVF倒排索引卸载到ES端实现，可以减少堆外内存的使用，以及减少写入/合并的性能开销，但是查询的性能也有一定的损失。采用默认值即可。 取值范围：true、false。 默认值：false。

8.7 管理向量索引缓存

CSS的向量检索引擎使用C++实现，使用的是堆外内存，该插件提供了接口对向量索引的缓存进行管理。

- **查看缓存统计信息**
  ```
  GET /_vector/stats
  ```
  在向量插件实现中，向量索引与Lucene其他类型索引一样，每一个segment构造并存储一份索引文件，在查询时，该索引文件会被加载到堆外内存中。插件使用缓存机制对这些堆外内存进行管理。上述API能够查询当前堆外内存使用量、缓存命中次数、加载次数等信息。
- **预加载向量索引**
  ```
  PUT /_vector/warmup/{index_name}
  ```
  使用上述接口能将指定index_name的向量索引预加载至堆外内存供查询使用。
- **清除缓存**
  ```
  PUT /_vector/clear/cache
  PUT /_vector/clear/cache/index_name
  ```
  在使用向量索引时，缓存机制会限制堆外内存使用量。当总索引大小超出缓存大小限制时，将会发生索引项的换进换出，此时将会影响查询的性能。通过清除缓存API能够将不再使用的索引缓存清空，保证热数据索引的查询性能。

8.8 向量检索的客户端代码示例

Elasticsearch提供了标准的REST接口，以及Java、Python、Go等语言编写的客户端。

基于开源数据集SIFT1M（ http://corpus-texmex.irisa.fr/ ）和Python Elasticsearch Client，本节提供一份创建向量索引、导入向量数据和查询向量数据的代码示例，介绍如何使用客户端实现向量检索。

前提条件

客户端已经安装python依赖包。如果未安装可以执行如下命令安装：

```
pip install numpy
pip install elasticsearch==7.6.0
```

代码示例

```
import numpy as np
import time
import json

from concurrent.futures import ThreadPoolExecutor, wait
from elasticsearch import Elasticsearch
from elasticsearch import helpers

endpoint = 'http://xxx.xxx.xxx.xxx:9200/'

# 构建es客户端对象
es = Elasticsearch(endpoint)

# 索引mapping信息
index_mapping = '''
{
  "settings": {
    "index": {
      "vector": "true"
    }
  },
  "mappings": {
    "properties": {
      "my_vector": {
        "type": "vector",
        "dimension": 128,
```



```
      "indexing": true,
      "algorithm": "GRAPH",
      "metric": "euclidean"
    }
   }
  }
}
'''

# 创建索引
def create_index(index_name, mapping):
    res = es.indices.create(index=index_name, ignore=400, body=mapping)
    print(res)

# 删除索引
def delete_index(index_name):
    res = es.indices.delete(index=index_name)
    print(res)


# 刷新索引
def refresh_index(index_name):
    res = es.indices.refresh(index=index_name)
    print(res)


# 索引段合并
def merge_index(index_name, seg_cnt=1):
    start = time.time()
    es.indices.forcemerge(index=index_name, max_num_segments=seg_cnt, request_timeout=36000)
    print(f"在{time.time() - start}秒内完成merge")


# 加载向量数据
def load_vectors(file_name):
    fv = np.fromfile(file_name, dtype=np.float32)
    dim = fv.view(np.int32)[0]
    vectors = fv.reshape(-1, 1 + dim)[:, 1:]
    return vectors


# 加载ground_truth数据
def load_gts(file_name):
    fv = np.fromfile(file_name, dtype=np.int32)
    dim = fv.view(np.int32)[0]
    gts = fv.reshape(-1, 1 + dim)[:, 1:]
    return gts


def partition(ls, size):
    return [ls[i:i + size] for i in range(0, len(ls), size)]


# 写入向量数据
def write_index(index_name, vec_file):
    pool = ThreadPoolExecutor(max_workers=8)
    tasks = []

    vectors = load_vectors(vec_file)
    bulk_size = 1000
    partitions = partition(vectors, bulk_size)

    start = time.time()
    start_id = 0
    for vecs in partitions:
        tasks.append(pool.submit(write_bulk, index_name, vecs, start_id))
        start_id += len(vecs)
    wait(tasks)
    print(f"在{time.time() - start}秒内完成写入")
```



```
def write_bulk(index_name, vecs, start_id):
    actions = [
        {
            "_index": index_name,
            "my_vector": vecs[j].tolist(),
            "_id": str(j + start_id)
        }
        for j in range(len(vecs))
    ]
    helpers.bulk(es, actions, request_timeout=3600)


# 查询索引
def search_index(index_name, query_file, gt_file, k):
    print("Start query! Index name: " + index_name)

    queries = load_vectors(query_file)
    gt = load_gts(gt_file)

    took = 0
    precision = []
    for idx, query in enumerate(queries):
        hits = set()
        query_json = {
                "size": k,
                "_source": False,
                "query": {
                  "vector": {
                    "my_vector": {
                      "vector": query.tolist(),
                      "topk": k
                    }
                  }
                }
              }
        res = es.search(index=index_name, body=json.dumps(query_json))

        for hit in res['hits']['hits']:
            hits.add(int(hit['_id']))
        precision.append(len(hits.intersection(set(gt[idx, :k]))) / k)
        took += res['took']

    print("precision: " + str(sum(precision) / len(precision)))
    print(f"在{took / 1000:.2f}秒内完成检索，平均took大小为{took / len(queries):.2f}毫秒")


if __name__ == "__main__":
    vec_file = r"./data/sift/sift_base.fvecs"
    qry_file = r"./data/sift/sift_query.fvecs"
    gt_file = r"./data/sift/sift_groundtruth.ivecs"

    index = "test"
    create_index(index, index_mapping)
    write_index(index, vec_file)
    merge_index(index)
    refresh_index(index)

    search_index(index, qry_file, gt_file, 10)
```


9 使用 Kibana 相关操作

9.1 登录 Kibana

前提条件

已创建CSS集群。

操作步骤

- 通过控制台访问方式登录
  a. 登录云搜索服务管理控制台。
  b. 在“集群管理”页面选择需要登录的集群，单击“操作”列中的“Kibana”进入Kibana登录界面。
     - 非安全模式的集群：将直接进入Kibana操作界面。
     - 安全模式的集群：需要在登录页面输入用户名和密码，单击“Log In”进入Kibana操作界面。用户名默认为admin，密码为创建集群时设置的管理员密码。
  c. 登录成功后，可在Kibana界面进行相关操作访问Elasticsearch集群。
- 通过公网访问方式登录
  如果您在创建集群时候，开启了Kibana公网访问功能，则可以通过Kibana公网访问地址进行登录。

9.2 使用 Kibana 创建用户并授权

前提条件

集群必须开启安全模式。

参数说明

表 9-1 kibana 创建用户和授权

参数	描述
Permission	单个动作，例如创建索引（例如indices:admin/create）。
Action group 操作组	一组权限。例如，预定义的SEARCH操作组授权角色使用_search和_msearchAPI。
Role 角色	角色定义为权限或操作组的组合，包括对集群，索引，文档或字段的操作权限。
Backend role 后端角色	（可选）来自授权后端的其他外部角色（例如LDAP / Active Directory）。
User 用户	用户可以向Elasticsearch集群发出操作请求。用户具有凭证（例如，用户名和密码）、零个或多个后端角色以及零个或多个自定义属性。
Role mapping 角色映射	用户在成功进行身份验证后会担任角色。角色映射，就是将角色映射到用户（或后端角色）。例如，kibana_user（角色）到jdoe（用户）的映射意味着John Doe在获得kibana_user身份验证后获得了所有权限。同样，all_access（角色）到admin（后端角色）的映射意味着具有后端角色admin（来自LDAP / Active Directory服务器）的任何用户都获得了all_access身份验证后的所有权限。您可以将每个角色映射到许多用户和/或后端角色。

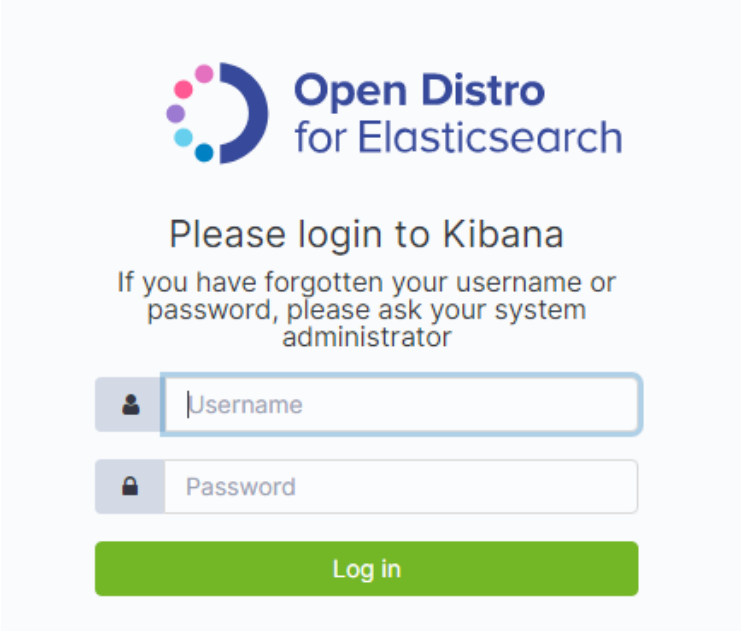
操作步骤

说明

- 不同的版本之间Kibana界面有差异，本章节以7.6.2版本为例。
- Kibana中可以自定义用户名、角色名、租户名等。

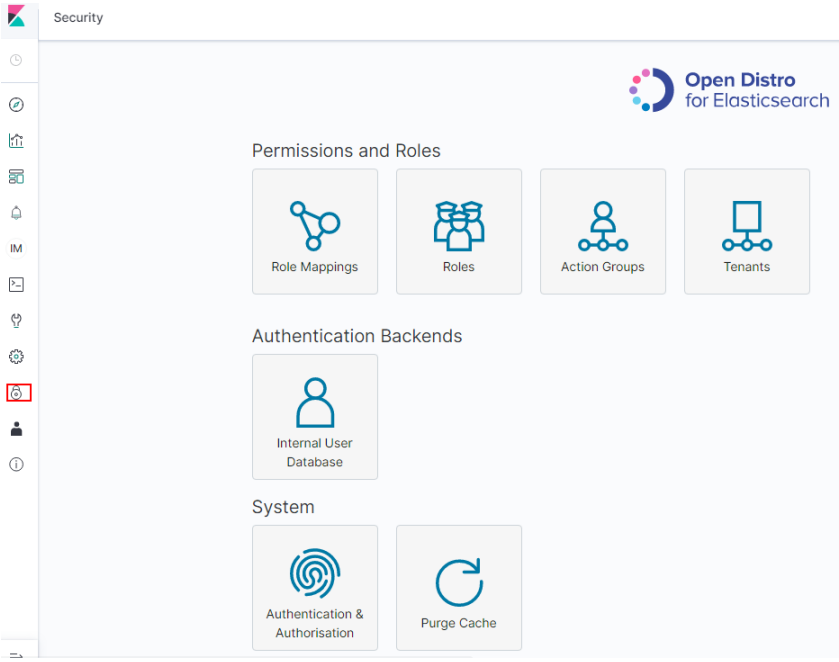
1. 登录云搜索服务控制台。
2. 在集群管理列表，选择对应集群，单击操作列的“Kibana”。
   输入管理员帐户名和密码登录Kibana。
   - 帐户名：admin（默认管理员帐户名）
   - 密码：创建安全模式的集群时，设置的管理员密码。

图 9-1 登录页面



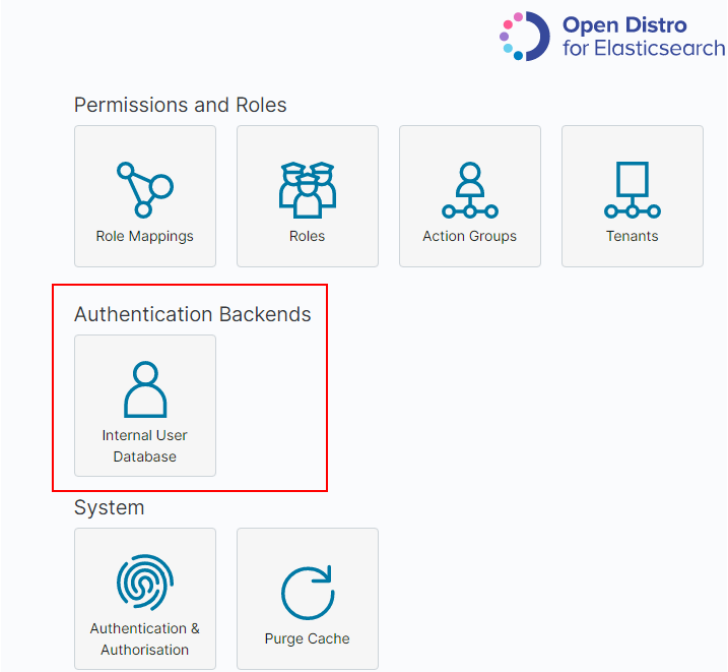
3. 登录成功后，在Kibana操作界面，选择“Security”，进入对应页面。

图 9-2 进入 Security 界面



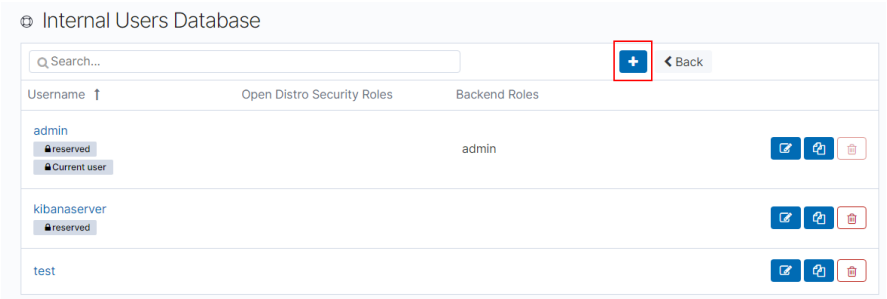
4. 创建用户。
   a. 选择“Authentication Backends”>“Internal Users Database”，进入创建用户页面。

图 9-3 添加用户（1）



b. 在“Internal Users Database”页面，选择  ，进入添加用户信息页面。

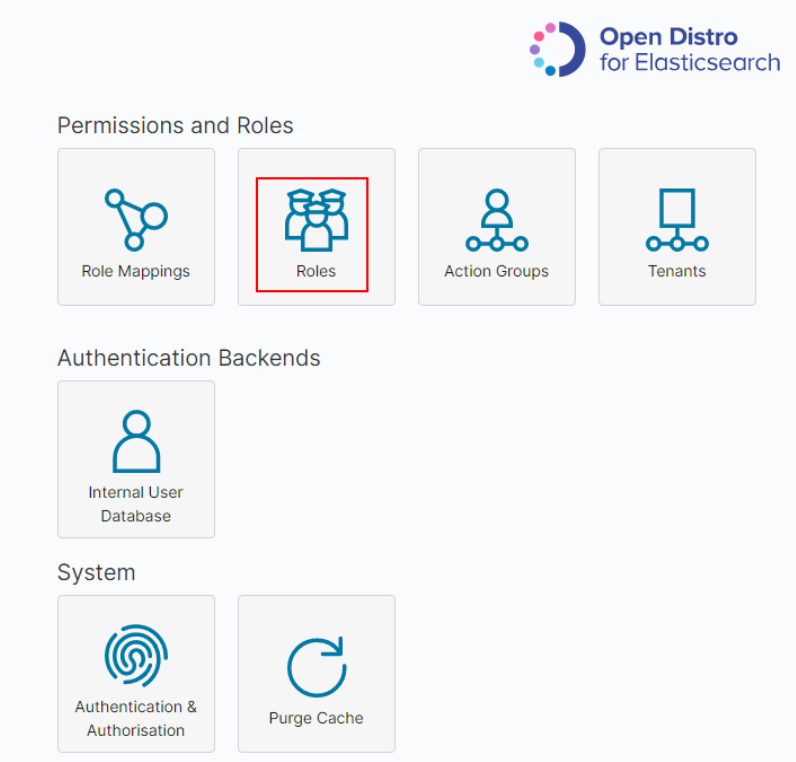
图 9-4 添加用户（2）



c. 在创建用户页面，输入“Username”和“Password”，单击“Submit”。创建成功后，可以在列表中看到新创建的用户。

5. 用户创建成功后，需要创建角色类型，设置权限。
   a. 在“Security”中选择“Roles”，进入Open Distro Security Roles页面。

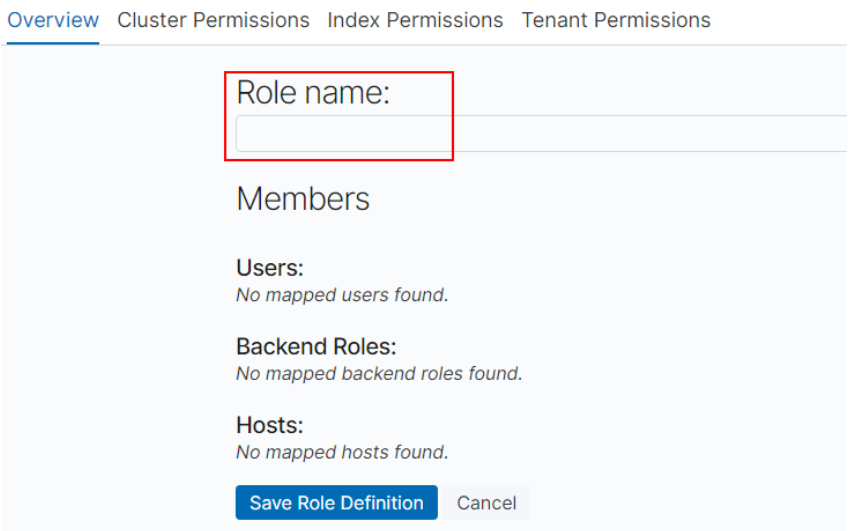
图 9-5 添加角色



b. 在Open Distro Security Roles页面，单击  添加角色权限。

i. 在Overview页面设置角色名。

图 9-6 添加角色名称



ii. 在“Cluster Permissions”页面设置CSS集群权限。此处无需设置。

iii. 在“Index Permissions”页面设置索引权限。

Index patterns：配置为需要设置权限的索引名称，例如，索引模板名称为my_store。

说明

建议索引名称和创建的用户名不要相同。

Permissions: Action Groups：根据需要开通的权限设置。例如，只读权限选择Search。

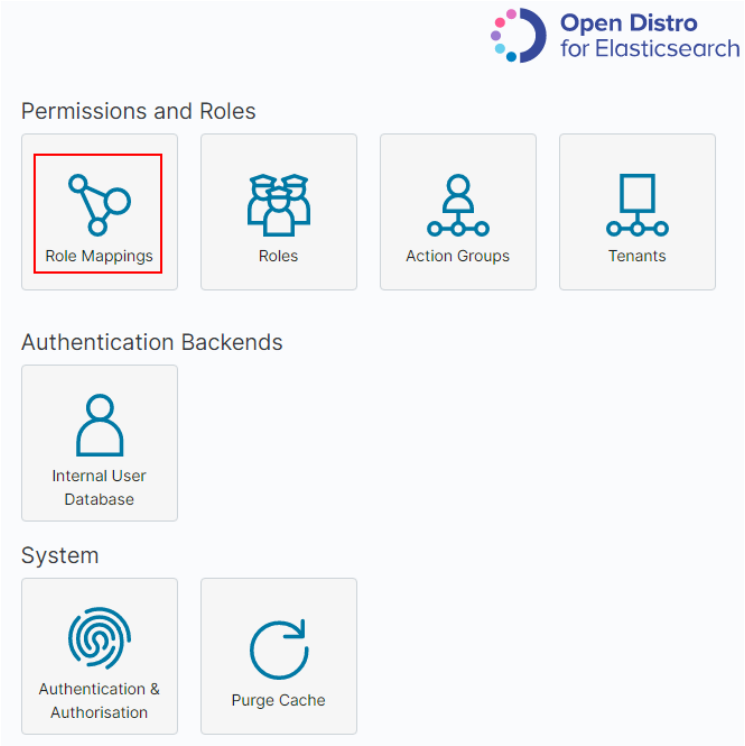
iv. “Tenant Permissions”页面无需设置。

设置完成后，即可看到设置的角色。

6. 给用户配置角色。

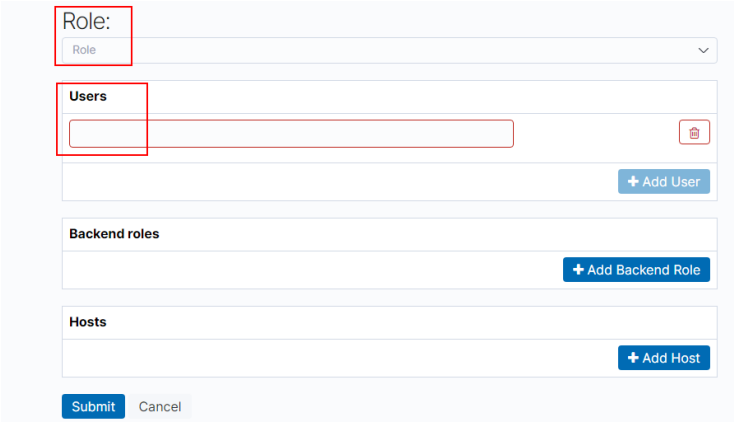
a. 在“Security”中选择“Role Mappings”，进入Role Mappings页面。

图 9-7 角色映射



b. 在Role Mappings页面，单击  ，添加用户和角色映射。

图 9-8 用户和角色映射



c. 添加完成后，单击“Submit”。

7. 配置完成后，可以在Kibana中进行验证是否生效。

9.3 索引状态管理

9.3.1 创建及管理索引

Elasticsearch 7.6.2及以上版本的集群支持索引状态管理。索引状态管理（ISM）是一个插件，通过该插件，您可以根据索引使用期限，索引大小或文档数的变化触发这些定期的管理操作，从而使它们自动化。使用ISM插件时，您可以根据需要定义自动处理索引滚动或删除的策略。

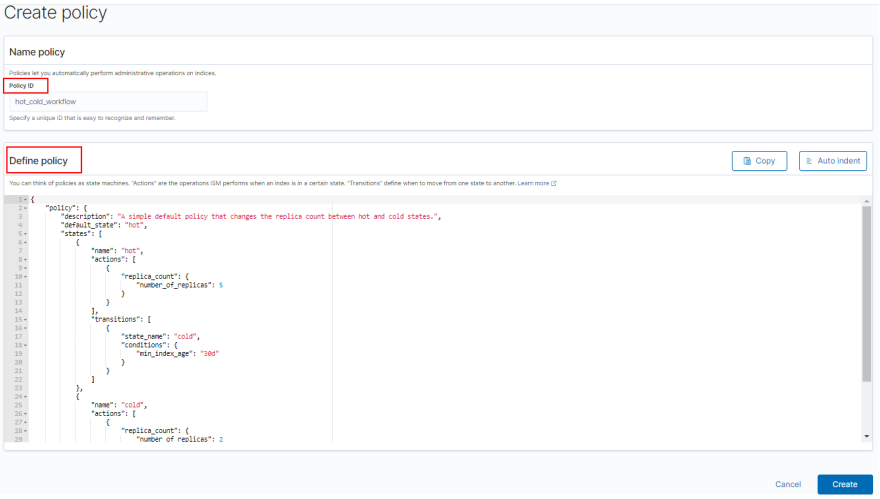
说明

如下操作步骤，以7.6.2版本为例，不同版本的Kibana界面有些差别，但是操作类似。

创建索引策略

1. 登录Kibana，在左侧选择“IM”或“Index Management”，进入索引管理页面 。
2. 右侧单击**Create policy**，创建索引策略。
3. 在**Policy ID**部分输入策略ID，**Define policy**部分输入您的策略。

图 9-9 配置策略



4. 单击**Create**完成索引策略的创建。

将策略附加到索引

创建索引策略后，可以将此策略附加到一个或多个索引，匹配该索引模板创建出的索引都将被附加该策略。

- **方式1：Kibana命令行**

  在Kibana的“Dev Tools”页面，执行如下命令在索引模板中关联策略ID。

```
PUT _template/<template_name>
{
```



```
  "index_patterns": ["index_name-*"],
  "settings": {
    "opendistro.index_state_management.policy_id": "policy_id"
  }
}
```

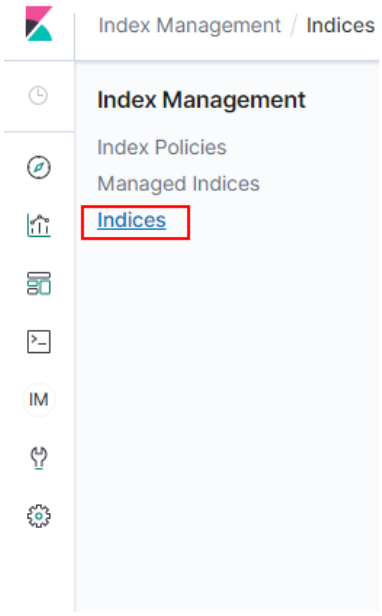
- <template_name>：需要替换为创建的索引模板名。
- policy_id：需要替换为自定义的策略ID，即Policy ID。

更多创建索引模板的说明可参考**索引模板**。

- **方式2：Kibana控制台**

  a. 在Kibana“Index Management”页面，选择**Indices**。

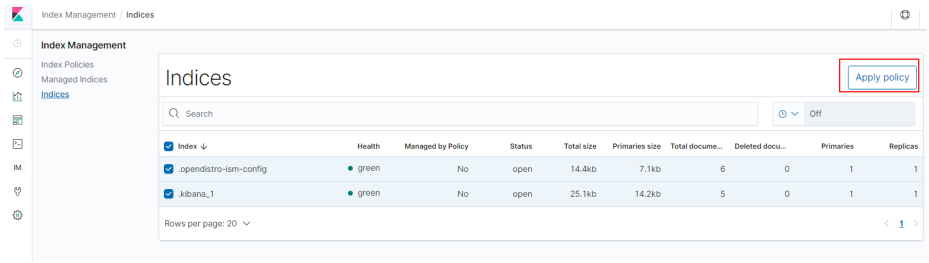
  图 9-10 选择 Indices

  b. 在**Indices**列表中选择您要附加策略的一个或多个索引。

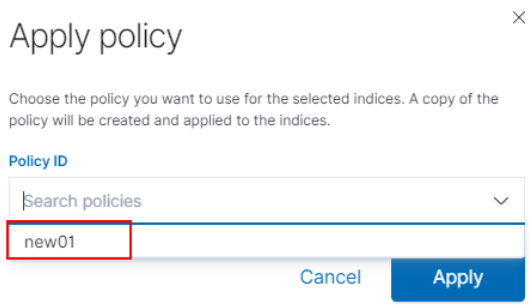
  c. 单击右上角的**Apply policy**，添加应用策略。

  图 9-11 添加引用策略

  d. 从**Policy ID**菜单中，选择您创建的策略。

图 9-12 选择



e. 单击**Apply**。

将策略附加到索引后，ISM会默认创建每5分钟运行一次的作业，以执行策略操作，检查条件并将索引转换为不同的状态。

管理索引策略

1. 选择**Managed Indices**。
2. 如果您要更改策略，可以选择**Change policy**，详情请参考**变更策略**。
3. 如果您要删除策略，请选择您的策略，然后选择**Remove policy**。
4. 如果您要重试策略，请选择您的策略，然后选择**Retry policy**。

具体使用可参考**索引管理官方介绍**。

9.3.2 变更策略

您可以更改任何托管索引策略，但是ISM有一些约束条件可以确保策略更改不会破坏索引。

如果索引卡在其当前状态，永不进行，并且您想立即更新其策略，请确保新策略包括与旧策略相同的状态（名称，操作，顺序相同）。在这种情况下，即使策略处于执行操作中，ISM也会应用新策略。

如果在不包含相同状态的情况下更新策略，则ISM仅在当前状态下的所有操作执行完成后才更新策略。或者，您可以在旧策略中选择特定状态，然后让新策略生效。

在Kibana中更改更改政策，操作步骤如下：

1. 在**Managed indices**下，选择需要更换新策略的索引。
2. 单击右上角的**Change policy**，进入**Choose managed indices**页面，选择更换新策略的相关信息 。

表 9-2 更换索引策略参数信息

参数	说明
Managed indices	选择需要更换策略的索引名称。支持选择多个索引。
State filters	选择索引状态。选择后，会将新策略附加到处于特定状态的索引。
New policy	选择新策略。

3. 选择完成后，单击**Change**。

9.4 自建 Kibana 如何对接 Elasticsearch?

自建Kibana对接Elasticsearch，需满足如下条件：

1. 本地环境需要支持外网访问。
2. 通过同vpc下ecs服务搭建Kibana，本地公网访问Kibana即可。

Kibana配置文件参考：

安全模式：

```
elasticsearch.username: "***"
elasticsearch.password: "***"
elasticsearch.ssl.verificationMode: none
server.ssl.enabled: false
server.rewriteBasePath: false
server.port: 5601
logging.dest: /home/Ruby/log/kibana.log
pid.file: /home/Ruby/run/kibana.pid
server.host: 192.168.25.226
elasticsearch.hosts: https://10.0.0.207:9200
elasticsearch.requestHeadersWhitelist: ["securitytenant","Authorization"]
opendistro_security.multitenancy.enabled: true
opendistro_security.multitenancy.tenants.enable_global: true
opendistro_security.multitenancy.tenants.enable_private: true
opendistro_security.multitenancy.tenants.preferred: ["Private", "Global"]
opendistro_security.multitenancy.enable_filter: false
```

说明

- 安全模式需要安装插件opendistro_security_kibana，详细请参考*https://opendistro.github.io/for-elasticsearch-docs/docs/kibana/plugins/*。
- 安装的插件版本需要和集群版本保持一致，可通过**GET _cat/plugins**获取到集群安全插件的版本号。

非安全模式：

```
server.port: 5601
logging.dest: /home/Ruby/log/kibana.log
pid.file: /home/Ruby/run/kibana.pid
server.host: 192.168.25.226
elasticsearch.hosts: http://10.0.0.207:9200
```

9.5 Kibana 使用限制

Kibana中可以自定义用户名、角色名、租户名等。

10 查询 Elasticsearch SQL

在6.5.4及之后版本中提供Open Distro for Elasticsearch SQL插件允许您使用SQL而不是Elasticsearch查询域特定语言（DSL）编写查询。

如果您已经熟悉SQL并且不想学习DSL查询，那么此功能是一个很好的选择。

基本操作

- Kibana（推荐）
  - 登录Kibana，在DevTools中将请求发送到_opendistro/_sqlURI，可以使用请求参数或请求正文。

    ```
    GET _opendistro/_sql?sql=select * from my-index limit 50
    POST _opendistro/_sql
    {
      "query": "SELECT * FROM my-index LIMIT 50"
    }
    ```

  - 默认情况下，查询返回JSON。您也可以选择CSV格式返回数据，选择CSV格式需要对format参数进行如下设置：

    ```
    POST _opendistro/_sql?format=csv
    {
      "query": "SELECT * FROM my-index LIMIT 50"
    }
    ```

    CSV格式返回数据时，每行对应一个文档，每列对应一个字段。
- curl命令

  您也可以在ECS中使用curl命令，来执行该SQL操作。

  ```
  curl -XPOST https://localhost:9200/_opendistro/_sql -u username:password -k -d '{"query": "SELECT * FROM kibana_sample_data_flights LIMIT 10"}' -H 'Content-Type: application/json'
  ```

支持操作

支持的SQL操作包括声明、条件、聚合函数、Include和Exclude、常用函数、连接join和展示等操作。

- 声明statements

表 10-1 声明 statements

Statement	Example
Select	SELECT * FROM my-index
Delete	DELETE FROM my-index WHERE _id=1
Where	SELECT * FROM my-index WHERE ['field']='value'
Order by	SELECT * FROM my-index ORDER BY _id asc
Group by	SELECT * FROM my-index GROUP BY range(age, 20,30,39)
Limit	SELECT * FROM my-index LIMIT 50 (default is 200)
Union	SELECT * FROM my-index1 UNION SELECT * FROM my-index2
Minus	SELECT * FROM my-index1 MINUS SELECT * FROM my-index2

说明

与任何复杂查询一样，大型UNION和MINUS语句可能会使集群资源紧张甚至崩溃。

- 条件Conditions

表 10-2 条件 Conditions

Condition	Example
Like	SELECT * FROM my-index WHERE name LIKE 'j%'
And	SELECT * FROM my-index WHERE name LIKE 'j%' AND age > 21
Or	SELECT * FROM my-index WHERE name LIKE 'j%' OR age > 21
Count distinct	SELECT count(distinct age) FROM my-index
In	SELECT * FROM my-index WHERE name IN ('alejandro', 'carolina')
Not	SELECT * FROM my-index WHERE name NOT IN ('jane')
Between	SELECT * FROM my-index WHERE age BETWEEN 20 AND 30
Aliases	SELECT avg(age) AS Average_Age FROM my-index
Date	SELECT * FROM my-index WHERE birthday='1990-11-15'
Null	SELECT * FROM my-index WHERE name IS NULL

- 聚合函数Aggregation

  表 10-3 聚合函数 Aggregation

Aggregation	Example
avg()	SELECT avg(age) FROM my-index
count()	SELECT count(age) FROM my-index
max()	SELECT max(age) AS Highest_Age FROM my-index
min()	SELECT min(age) AS Lowest_Age FROM my-index
sum()	SELECT sum(age) AS Age_Sum FROM my-index

- Include和Exclude字段

  表 10-4 Include 和 Exclude

Pattern	Example
include()	SELECT include('a*'), exclude('age') FROM my-index
exclude()	SELECT exclude('*name') FROM my-index

- 函数Functions

  表 10-5 函数 Functions

Function	Example
floor	SELECT floor(number) AS Rounded_Down FROM my-index
trim	SELECT trim(name) FROM my-index
log	SELECT log(number) FROM my-index
log10	SELECT log10(number) FROM my-index
substring	SELECT substring(name, 2,5) FROM my-index
round	SELECT round(number) FROM my-index
sqrt	SELECT sqrt(number) FROM my-index
concat_ws	SELECT concat_ws(' ', age, height) AS combined FROM my-index

Function	Example
/	SELECT number / 100 FROM my-index
%	SELECT number % 100 FROM my-index
date_format	SELECT date_format(date, 'Y') FROM my-index

说明

必须在文档映射中启用fielddata才能使大多数字符串函数正常工作。

- 连接操作Joins

表 10-6 连接操作 Joins

Join	Example
Inner join	SELECT s.firstname, s.lastname, s.gender, sc.name FROM student s JOIN school sc ON sc.name = s.school_name WHERE s.age > 20
Left outer join	SELECT s.firstname, s.lastname, s.gender, sc.name FROM student s LEFT JOIN school sc ON sc.name = s.school_name
Cross join	SELECT s.firstname, s.lastname, s.gender, sc.name FROM student s CROSS JOIN school sc

相关约束和限制，参考**连接操作Joins**。

- 展示Show

展示show操作与索引模式匹配的索引和映射。您可以使用*或%使用通配符。

表 10-7 展示 show

Show	Example
Show tables like	SHOW TABLES LIKE logs-*

连接操作 Joins

Open Distro for Elasticsearch SQL支持inner joins, left outer joins,和cross joins。Join操作有许多约束：

- 您只能加入两个参数。

- 您必须为索引使用别名（例如people p）。
- 在ON子句中，您只能使用AND条件。
- 在WHERE语句中，不要将包含多个索引的树组合在一起。例如，以下语句有效：
  ```
  WHERE (a.type1 > 3 OR a.type1 < 0) AND (b.type2 > 4 OR b.type2 < -1)
  ```
  以下声明无效：
  ```
  WHERE (a.type1 > 3 OR b.type2 < 0) AND (a.type1 > 4 OR b.type2 < -1)
  ```
- 您不能使用GROUP BY或ORDER BY来获得结果。
- LIMIT和OFFSET不支持一起使用（例如LIMIT 25 OFFSET 25）。

JDBC 驱动

Java数据库连接（JDBC）驱动程序允许您将Open Distro for Elasticsearch与您的商业智能（BI）应用程序集成。

有关下载和使用JAR文件的信息，请参阅**GitHub仓库**。

11 增强特性

11.1 存算分离

11.1.1 背景信息

云搜索服务支持将热数据存储在SSD来达到最佳的查询检索性能。将历史数据存储在OBS中降低数据的存储成本，该特性为存算分离特性。

使用场景

对于有海量数据写入和存储的场景，一般数据有明显的冷热区分，新写入的数据存储在SSD中，随着时间的推移，历史数据不再写入，查询QPS也降低，这时候可以调用云搜索服务提供的API将存储在SSD的热数据转储到OBS，这个转储的过程称为冻结索引，也就是存算分离。

约束限制

- 目前仅7.6.2和7.10.2版本支持存算分离。
- 由于存算分离的特性依赖OBS，所以使用过程中要遵守OBS的“带宽”和“每秒请求数（QPS）”的使用限制。当超过限制时，集群中涉及到OBS查询的性能都会下降，例如恢复分片的速度变慢、查询数据时变慢等。

11.1.2 冻结索引

注意事项

- 在执行冻结操作前，需冻结的索引没有数据写入。在冻结操作执行前，会将索引配置为read only，会导致写入数据出错。
- 在执行冻结操作后：
  - 索引变为只读。
  - 索引数据将会转储到OBS，转移过程中，会占用网络带宽。
  - 转储后的索引，查询时延会增加。聚合时，由于查询复杂，数据读取多，时延变长会体现的更明显。

- 已冻结的索引不支持解冻，即不可回退为可写的索引。
- 冻结完成以后，会删除本地磁盘中的索引数据。

操作步骤

1. 登录云搜索服务管理控制台。
2. 在“集群管理”页面，选择需要冻结索引的集群，单击操作列“Kibana”，登录Kibana界面。
3. 单击左侧导航栏的“Dev Tools”，进入操作页面。
4. 执行如下命令，将指定索引冻结到OBS中。

```
POST ${index_name}/_freeze_low_cost
```

表 11-1 请求参数说明

参数名	说明
index_name	需要冻结的索引的名字。

返回结果如下：

```
{
   "freeze_uuid": "pdsRgUtSTymVDWR_HoTGFw"
}
```

表 11-2 返回参数说明

参数名	说明
freeze_uuid	提交冻结请求后会启动一个异步任务，请求返回异步任务的ID，使用该ID查询异步任务的进度。

说明

索引冻结请求下发后，会禁止索引的数据写入，冻结过程中，查询请求不受影响。在冻结结束后，会将索引先close再open，在这段时间内，索引不可查询，集群可能短暂出现red状态，open结束后恢复。

5. 执行如下命令获取冻结任务进度。

```
GET _freeze_low_cost_progress/${freeze_uuid}
```

表 11-3 请求参数说明

参数名	说明
freeze_uuid	异步任务的ID，该ID由**4**获取的。

返回结果如下：

```
{

 "stage" : "STARTED",
```



```
"shards_stats" : {
  "INIT" : 0,
  "FAILURE" : 0,
  "DONE" : 0,
  "STARTED" : 3,
  "ABORTED" : 0
},
"indices" : {
  "data1" : [
    {
      "uuid" : "7OS-G1-tRke2jHZPlckexg",
      "index" : {
        "name" : "data1",
        "index_id" : "4b5PHXJITLaS6AurImfQ9A",
        "shard" : 2
      },
      "start_ms" : 1611972010852,
      "end_ms" : -1,
      "total_time" : "10.5s",
      "total_time_in_millis" : 10505,
      "stage" : "STARTED",
      "failure" : null,
      "size" : {
        "total_bytes" : 3211446689,
        "finished_bytes" : 222491269,
        "percent" : "6.0%"
      },
      "file" : {
        "total_files" : 271,
        "finished_files" : 12,
        "percent" : "4.0%"
      },
      "rate_limit" : {
        "paused_times" : 1,
        "paused_nanos" : 946460970
      }
    },
    {
      "uuid" : "7OS-G1-tRke2jHZPlckexg",
      "index" : {
        "name" : "data1",
        "index_id" : "4b5PHXJITLaS6AurImfQ9A",
        "shard" : 0
      },
      "start_ms" : 1611972010998,
      "end_ms" : -1,
      "total_time" : "10.3s",
      "total_time_in_millis" : 10359,
      "stage" : "STARTED",
      "failure" : null,
      "size" : {
        "total_bytes" : 3221418186,
        "finished_bytes" : 272347118,
        "percent" : "8.0%"
      },
      "file" : {
        "total_files" : 372,
        "finished_files" : 16,
        "percent" : "4.0%"
      },
      "rate_limit" : {
        "paused_times" : 5,
        "paused_nanos" : 8269016764
      }
    },
    {
      "uuid" : "7OS-G1-tRke2jHZPlckexg",
      "index" : {
        "name" : "data1",
```



```
      "index_id" : "4b5PHXJITLaS6AurImfQ9A",
      "shard" : 1
    },
    "start_ms" : 1611972011021,
    "end_ms" : -1,
    "total_time" : "10.3s",
    "total_time_in_millis" : 10336,
    "stage" : "STARTED",
    "failure" : null,
    "size" : {
      "total_bytes" : 3220787498,
      "finished_bytes" : 305789614,
      "percent" : "9.0%"
    },
    "file" : {
      "total_files" : 323,
      "finished_files" : 14,
      "percent" : "4.0%"
    },
    "rate_limit" : {
      "paused_times" : 3,
      "paused_nanos" : 6057933087
    }
   }
  ]
 }
}
```


表 11-4 返回参数说明

参数名	说明
stage	当前所处状态。取值包括： ● INIT：刚启动或者正在初始化。 ● FAILURE：失败。 ● DONE：完成。 ● STARTED：已启动。 ● ABORTED：取消，预留字段。
shards_stats	处在各个状态的shard个数。
indices	每个索引的状态详情。

表 11-5 indices 返回值说明

参数名	说明
uuid	freeze的uuid。
index	索引信息和shard信息。
start_ms	开始时间。
end_ms	结束时间，如果没有结束则显示为-1。
total_time	已花费时间。
total_time_in_millis	已花费时间毫秒数。

参数名	说明
stage	当前shard所处的状态。
failure	失败原因，如果没有失败则显示为null。
size.total_bytes	总共需要冻结的文件的字节数。
size.finished_bytes	已经完成冻结的字节数。
size.percent	已经完成冻结的字节数百分比。
file.total_bytes	总共需要冻结的文件个数。
file.finished_bytes	已经完成冻结的文件个数。
file.percent	已经完成冻结的文件个数百分比。
rate_limit.paused_times	达到限速导致冻结暂停的次数。
rate_limit.paused_nanos	达到限速导致冻结暂停的时间纳秒数。

冻结完成的索引会增加以下settings，可参考**表11-6**。

表 11-6 冻结索引 settings

参数	说明
index.frozen_low_cost	标识该索引为冻结索引。取值为true。
index.blocks.write	冻结后的索引禁止写入。取值为true。
index.store.type	标识该索引的存储类型为obs。取值为obs。

6. 索引冻结后，会将数据进行缓存。执行如下命令获取当前缓存状态。关于缓存详见**配置缓存**。

```
GET _frozen_stats
GET _frozen_stats/${node_id}
```

表 11-7 请求参数说明

参数名	说明
node_id	获取单个节点的缓存状态，此参数为需要获取的节点id。

返回结果如下：

```
{
  "_nodes" : {
    "total" : 3,
    "successful" : 3,
```



```
  "failed" : 0
},
"cluster_name" : "css-zzz1",
"nodes" : {
  "7uwKO38RRoaON37YsXhCYw" : {
    "name" : "css-zzz1-ess-esn-2-1",
    "transport_address" : "10.0.0.247:9300",
    "host" : "10.0.0.247",
    "ip" : "10.0.0.247",
    "block_cache" : {
      "default" : {
        "type" : "memory",
        "block_cache_capacity" : 8192,
        "block_cache_blocksize" : 8192,
        "block_cache_size" : 12,
        "block_cache_hit" : 14,
        "block_cache_miss" : 0,
        "block_cache_eviction" : 0,
        "block_cache_store_fail" : 0
      }
    },
    "obs_stats" : {
      "list" : {
        "obs_list_count" : 17,
        "obs_list_ms" : 265,
        "obs_list_avg_ms" : 15
      },
      "get_meta" : {
        "obs_get_meta_count" : 79,
        "obs_get_meta_ms" : 183,
        "obs_get_meta_avg_ms" : 2
      },
      "get_obj" : {
        "obs_get_obj_count" : 12,
        "obs_get_obj_ms" : 123,
        "obs_get_obj_avg_ms" : 10
      },
      "put_obj" : {
        "obs_put_obj_count" : 12,
        "obs_put_obj_ms" : 2451,
        "obs_put_obj_avg_ms" : 204
      },
      "obs_op_total" : {
        "obs_op_total_ms" : 3022,
        "obs_op_total_count" : 120,
        "obs_op_avg_ms" : 25
      }
    },
    "reader_cache" : {
      "hit_count" : 0,
      "miss_count" : 1,
      "load_success_count" : 1,
      "load_exception_count" : 0,
      "total_load_time" : 291194714,
      "eviction_count" : 0
    }
  },
  "73EDpEqoQES749umJqxOzQ" : {
    "name" : "css-zzz1-ess-esn-3-1",
    "transport_address" : "10.0.0.201:9300",
    "host" : "10.0.0.201",
    "ip" : "10.0.0.201",
    "block_cache" : {
      "default" : {
        "type" : "memory",
        "block_cache_capacity" : 8192,
        "block_cache_blocksize" : 8192,
        "block_cache_size" : 12,
        "block_cache_hit" : 14,
```



```
      "block_cache_miss" : 0,
      "block_cache_eviction" : 0,
      "block_cache_store_fail" : 0
    }
   },
   "obs_stats" : {
    "list" : {
      "obs_list_count" : 17,
      "obs_list_ms" : 309,
      "obs_list_avg_ms" : 18
    },
    "get_meta" : {
      "obs_get_meta_count" : 79,
      "obs_get_meta_ms" : 216,
      "obs_get_meta_avg_ms" : 2
    },
    "get_obj" : {
      "obs_get_obj_count" : 12,
      "obs_get_obj_ms" : 140,
      "obs_get_obj_avg_ms" : 11
    },
    "put_obj" : {
      "obs_put_obj_count" : 12,
      "obs_put_obj_ms" : 1081,
      "obs_put_obj_avg_ms" : 90
    },
    "obs_op_total" : {
      "obs_op_total_ms" : 1746,
      "obs_op_total_count" : 120,
      "obs_op_avg_ms" : 14
    }
   },
   "reader_cache" : {
    "hit_count" : 0,
    "miss_count" : 1,
    "load_success_count" : 1,
    "load_exception_count" : 0,
    "total_load_time" : 367179751,
    "eviction_count" : 0
   }
  },
  "EF8WoLCUQbqJl1Pkqo9-OA" : {
   "name" : "css-zzz1-ess-esn-1-1",
   "transport_address" : "10.0.0.18:9300",
   "host" : "10.0.0.18",
   "ip" : "10.0.0.18",
   "block_cache" : {
    "default" : {
      "type" : "memory",
      "block_cache_capacity" : 8192,
      "block_cache_blocksize" : 8192,
      "block_cache_size" : 12,
      "block_cache_hit" : 14,
      "block_cache_miss" : 0,
      "block_cache_eviction" : 0,
      "block_cache_store_fail" : 0
    }
   },
   "obs_stats" : {
    "list" : {
      "obs_list_count" : 17,
      "obs_list_ms" : 220,
      "obs_list_avg_ms" : 12
    },
    "get_meta" : {
      "obs_get_meta_count" : 79,
      "obs_get_meta_ms" : 139,
      "obs_get_meta_avg_ms" : 1
    },
```



```
      "get_obj" : {
        "obs_get_obj_count" : 12,
        "obs_get_obj_ms" : 82,
        "obs_get_obj_avg_ms" : 6
      },
      "put_obj" : {
        "obs_put_obj_count" : 12,
        "obs_put_obj_ms" : 879,
        "obs_put_obj_avg_ms" : 73
      },
      "obs_op_total" : {
        "obs_op_total_ms" : 1320,
        "obs_op_total_count" : 120,
        "obs_op_avg_ms" : 11
      }
    },
    "reader_cache" : {
      "hit_count" : 0,
      "miss_count" : 1,
      "load_success_count" : 1,
      "load_exception_count" : 0,
      "total_load_time" : 235706838,
      "eviction_count" : 0
    }
  }
 }
}
```

7. 执行如下命令重置缓存状态。

```
POST _frozen_stats/reset
```

返回结果如下：

```
{
 "_nodes" : {
  "total" : 1,
  "successful" : 1,
  "failed" : 0
 },
 "cluster_name" : "Es-0325-007_01",
 "nodes" : {
  "mqTdk2YRSPyOSXfesREFSg" : {
   "result" : "ok"
  }
 }
}
```

说明

此命令用于性能问题的调试，如重置缓存状态后再次执行查询，可以清晰看到本次查询的缓存命令情况。在业务运行阶段不需要执行此命令。

8. 执行如下命令获取当前已经冻结的所有索引。

```
GET _cat/freeze_indices
```

返回结果如下：

```
green open data2 0bNtxWDtRbOSkS4JYaUgMQ 3 0  5 0  7.9kb  7.9kb
green open data3 oYMLvw31QnyasqUNuyP6RA 3 0 51 0 23.5kb 23.5kb
```

说明

此命令的参数和返回值与开源ES的_cat/indices一致。

11.1.3 配置缓存

将数据转储到OBS后，为了尽可能的减少对OBS的访问请求，并提升ES的查询性能，系统将会缓存部分数据。第一次获取到数据时，会直接访问OBS，之后将获取到的数据缓存在内存中，后续访问会先检查是否有缓存。数据缓存支持内存和文件。

ES访问不同的文件访问的模式是不一样的，缓存系统支持多级缓存，分别使用不同的block大小来缓存不同的文件，如对fdx，tip文件，使用大量的小block缓存，对fdt文件，使用较少的大block缓存。

表 11-8 针对缓存的所有配置

配置名	类型	说明
low_cost.obs.blockcache.names	Array	缓存系统支持多级缓存，分别用来缓存不同访问粒度的数据。此配置列出所有缓存的名字，即使不配置，系统也会默认有一个缓存，名字为default。如果自定义配置，请确保有一个名字为default的缓存，其他名字任意。 默认值：default。
low_cost.obs.blockcache.<NAME>.type	ENUM	缓存的类型，支持memory和file。 当使用memory类型的缓存时，会占用一定的内存大小。当使用file类型的缓存时，会使用磁盘作为缓存。建议使用超高IO型的磁盘提升缓存性能。 默认值：memory。
low_cost.obs.blockcache.<NAME>.blockshift	Integer	缓存每个block的大小，为字节左移数，即2^x字节。如配置为16，表示block大小为2^{16}字节，等于65536字节，即64K。 默认值：13（即8K）。
low_cost.obs.blockcache.<NAME>.bank.count	Integer	缓存分区数。 默认值：1。
low_cost.obs.blockcache.<NAME>.number.blocks.perbank	Integer	每个缓存分区中包含的block数。 默认值：8192。
low_cost.obs.blockcache.<NAME>.exclude.file.types	Array	不缓存的文件后缀名。如果某些后缀既不包含在exclude列表，也不包含在include列表，则会使用default缓存。
low_cost.obs.blockcache. <NAME>.file.types	Array	缓存的文件后缀名。如果某些后缀既不包含在exclude列表，也不包含在include列表，则会使用default缓存。

以下为一个较为常见的缓存配置，该配置使用两级缓存，名字分别为default和large。其中default缓存使用64K的block大小，并且一共有30*4096个block，default缓存用于缓存除fdt后缀的其他文件。large缓存使用2M的block大小，一共有5*1000个block，large缓存用于缓存fdx，dvd，tip后缀的文件。

```
low_cost.obs.blockcache.names: ["default", "large"]
low_cost.obs.blockcache.default.type: file
low_cost.obs.blockcache.default.blockshift: 16
low_cost.obs.blockcache.default.number.blocks.perbank: 4096
low_cost.obs.blockcache.default.bank.count: 30
```



```
low_cost.obs.blockcache.default.exclude.file.types: ["fdt"]

low_cost.obs.blockcache.large.type: file
low_cost.obs.blockcache.large.blockshift: 21
low_cost.obs.blockcache.large.number.blocks.perbank: 1000
low_cost.obs.blockcache.large.bank.count: 5
low_cost.obs.blockcache.large.file.types: ["fdx", "dvd", "tip"]
```

表 11-9 其他可配置参数

配置名	类型	说明
index.frozen.obs.max_bytes_per_sec	String	冻结过程中往OBS上传文件最大限速。动态配置，修改后立即生效。 默认值：150MB。
low_cost.obs.index.upload.threshold.use.multipart	String	冻结过程中文件大小超过此配置会使用OBS的分段上传。 默认值：1GB。
index.frozen.reader.cache.expire.duration.seconds	Integer	此参数设置超时时间。 为了减少冻结后的索引占用的堆内存，在索引shard启动后，reader会缓存一段时间，超时后关闭。 默认值：300s。
index.frozen.reader.cache.max.size	Integer	配置缓存最大值。 默认值：100。

11.2 流量控制

11.2.1 背景信息

特性介绍

云搜索服务支持流量控制特性，提供节点级别的流量控制功能，可提供单个节点基于黑白名单的访问限制、HTTP并发连接数限制、HTTP最大连接数限制、基于请求Path的堆内存最大使用量流控能力、基于CPU最大占用率流控能力，一键断流能力，同时也提供节点访问IP统计和URI的采样统计能力。每个功能配置独立的控制开关，默认关闭。所有参数配置为null可以恢复默认值。

开启流控功能会使请求在入口处直接阻塞，可以缓解节点高并发场景下的集群压力，降低P99时延，减少节点不可用的风险。

- **HTTP/HTTPS流控**：
  - HTTP/HTTPS黑白名单设置IP和子网控制客户端IP访问，如果节点IP在黑名单中，则该客户端的连接将直接中断，节点不会处理任何请求。白名单规则优先于黑名单规则，如果客户端IP在黑白名单中都出现，客户端请求将不会被拒绝。
  - HTTP/HTTPS并发连接数流控通过限制节点每秒中的HTTP连接总数来限制节点流量。

  - HTTP/HTTPS新建连接数流控通过限制节点新建的连接数来限制节点流量。
- **内存流控**基于节点堆内存使用量限制请求Path，支持设置内存流控白名单、全局内存使用阈值和基于单个Path设置堆内存使用阈值。全局内存流控阈值优先于单个Path内存阈值，白名单配置Path不参与内存流控。
- **Path全局免流控白名单**可以根据客户需要，设置Path白名单全局免流控，当用户需要自定义插件时，可适当配置。
- **请求采样统计**可以记录客户端IP的访问数量和采样用户的请求Path，用户可以基于统计值识别客户端IP的访问流量和分析哪些请求Path访问量大。
- **流量控制**提供单独的流量统计查看接口，记录触发流控的数量，用户可以基于统计值评估流控配置阈值和衡量集群压力。
- **访问日志**可以记录一段时间内节点接收的HTTP/HTTPS请求URL和Body，用户可以基于访问日志信息分析当前的流量压力。
- **CPU流控**基于节点配置的最大CPU占用率来限制节点访问流量。
- **一键断流**可以切断节点的所有访问流量，不包括kibana访问和elasticsearch monitor类接口。

约束限制

- 目前仅7.6.2和7.10.2版本支持流量控制特性。
- 开启流量控制功能会消耗部分节点性能。
- 开启流控会直接拒绝超过阈值的用户请求。
- 内存流控和CPU流控都是基于请求Path的流控，Path长度和个数不应该配置过多，否则影响集群性能。

11.2.2 HTTP/HTTPS 流控

背景信息

通过在Kibana执行命令，可以开启或关闭集群的HTTP/HTTPS流控。执行命令涉及的配置参数如下：

表 11-10 HTTP/HTTPS 流控的配置参数说明

配置名	类型	说明
flowcontrol.http.enabled	Boolean	HTTP/HTTPS流控开关，默认关闭，开启会影响节点访问性能。 取值范围：true、false 默认值：false
flowcontrol.http.allow	List<String>	IP地址访问白名单。 支持配置多个IP地址和掩码，或者IP地址列表形式，中间用英文逗号隔开。例如“xx.xx.xx.xx/24,xx.xx.xx.xx/24”或“xx.xx.xx.xx,xx.xx.xx.xx”形式。 默认值为空。

配置名	类型	说明
flowcontrol.http.deny	List<String>	IP访问黑名单。 支持配置多个IP和掩码，或者IP列表形式，中间用英文逗号隔开。 默认值为空。
flowcontrol.http.concurrent	Integer	HTTP/HTTPS请求的并发连接数阈值。 默认值：节点可用核数 * 400
flowcontrol.http.newconnect	Integer	HTTP/HTTPS请求的每秒可以创建的新建连接数阈值。 默认值：节点可用核数 * 200
flowcontrol.http.warmup_period	Integer	HTTP/HTTPS新建连接数达到最大速率的需要的时间，如果“flowcontrol.http.newconnect”配置为“100”且“flowcontrol.http.warmup_period”配置为“5000ms”，表示5s以后系统的新建连接数才可以达到每秒100。 取值范围：0~10000 单位：ms 默认值：0

操作步骤

1. 登录云搜索服务管理控制台。
2. 在“集群管理”页面选择目标集群，单击操作列“Kibana”，登录Kibana界面。
3. 单击左侧导航栏的“Dev Tools”，执行命令开启或关闭HTTP/HTTPS流控。
   - **开启HTTP/HTTPS节点流控**

     ```
     PUT /_cluster/settings
     {
       "persistent": {
         "flowcontrol.http.enabled": true,
         "flowcontrol.http.allow": ["192.168.0.1/24", "192.168.2.1/24"],
         "flowcontrol.http.deny": "192.168.1.1/24",
         "flowcontrol.http.concurrent": 1000,
         "flowcontrol.http.newconnect": 1000,
         "flowcontrol.http.warmup_period": 0
       }
     }
     ```

     说明

     当所有参数设置为null时，表示恢复配置默认值。

   - **关闭HTTP/HTTPS节点流控**

     ```
     PUT /_cluster/settings
     {
       "persistent": {
         "flowcontrol.http.enabled": false
       }
     }
     ```


11.2.3 内存流控

背景信息

Elasticsearch内部有熔断器机制，可以配置内存使用的阈值，当节点内存超过指定值，触发熔断，请求操作终止。但是Elasticsearch在调用API时没有判断当前的堆内存使用量，如果在请求处理过程中计算，即使熔断也会造成堆内存的消耗，频繁熔断会导致节点不可用，同时熔断器不支持单个请求的熔断阈值配置。但是，当在Rest请求入口处设置堆内存使用限制时，可以阻断API的请求，达到保护节点的目的。而内存流控可以配置节点全局流控和基于单个请求Path的精细化内存控制，其中单个请求Path的流控通过配置请求Path和堆内存阈值，在请求处理前判断配置的堆内存阈值，超过阈值中断当前的请求Path。

说明

- 开启内存流控会消耗部分请求性能。
- 开启内存流控会导致Kibana的部分search请求失败。
- ES 551版本开启内存流控会导致_mget请求被拦截，Kibana访问异常，可以把_mget请求加入请求白名单规避。

在开启或关闭集群的内存流控时，执行命令涉及的配置参数如下：

表 11-11 内存流控的配置参数说明

配置名	类型	说明
flowcontrol.memory.enabled	Boolean	内存流控开关，默认关闭，开启内存流控对节点性能有些许影响。 取值范围：true、false 默认值：false

配置名	类型	说明
flowcontrol.memory.allow_path	List<String>	内存流控白名单Path。 配置的路径不参与内存流控，可以支持通配符配置。集群控制的查询类接口默认放通，不参与内存流控，避免内存达到阈值，不能查询集群信息。 例如： ● "flowcontrol.memory.allow_path": "/index/_search", ● "flowcontrol.memory.allow_path": "/index*/_search", ● "flowcontrol.memory.allow_path": ["/index/_search", "/index1/_bulk"], 支持最大配置10个Path，每个Path最大长度限制小于32。 默认值为为空。
flowcontrol.memory.heap_limit	String	限制节点全局堆内存的最大使用率。不能低于堆内存的10%。 取值范围：10%-100% 默认值：90%

配置名	类型	说明
flowcontrol.memory.*.filter_path	String	配置需要进行内存流控的访问Path，控制单个请求Path的堆内存使用阈值。 默认值 "**", 表示匹配所有的路径。如果只配置了单路径“flowcontrol.memory.heap_limit”，没有配置“flowcontrol.memory.*.filter_path”，表示除去白名单外的所有path都影响。白名单规则优先于单路径规则，如果一个路径同时配置了“flowcontrol.memory.allow_path”和“flowcontrol.memory.*.filter_path”，此请求路径会被允许。 例如同时配置了“flowcontrol.memory.allow_path”和“flowcontrol.memory.*.filter_path”，其中flowcontrol.memory.*.filter_path="abc/_search"且flowcontrol.memory.allow_path="abc/_search"，此种情况abc/_search将不被流控。 最大长度：32
flowcontrol.memory.*.heap_limit	String	配置请求Path的堆内存阈值，堆内存超过阈值触发流控。 取值范围：0%-100% 默认值：90%

操作步骤

1. 登录云搜索服务管理控制台。
2. 在“集群管理”页面选择目标集群，单击操作列“Kibana”，登录Kibana界面。
3. 单击左侧导航栏的“Dev Tools”，执行命令开启或关闭内存流控。
   - **开启内存流控**

```
PUT /_cluster/settings
{
  "persistent": {
    "flowcontrol.memory.enabled": true,
```



```
    "flowcontrol.memory.allow_path": "/index/_search",
    "flowcontrol.memory.heap_limit": "85%"
  }
}
```

- **开启单个请求Path的内存流控**

  基于单个索引和请求Path设置堆内存使用阈值，可以基于此规则做优先级调度。

```
PUT /_cluster/settings
{
  "persistent": {
    "flowcontrol.memory.enabled": true,
    "flowcontrol.memory": {
      "flowcontrol_search": {
        "filter_path": "index1/_search",
        "heap_limit": "50%"
      },
      "flowcontrol_bulk": {
        "filter_path": "index*/_bulk",
        "heap_limit": "50%"
      }
    }
  }
}
```

- **删除单个请求Path的内存流控配置**

```
PUT /_cluster/settings
{
  "persistent": {
    "flowcontrol.memory.enabled": true,
    "flowcontrol.memory": {
      "flowcontrol_search": {
        "filter_path": null,
        "heap_limit": null
      }
    }
  }
}
```

- **关闭集群内存流控**

```
PUT /_cluster/settings
{
  "persistent": {
    "flowcontrol.memory.enabled": false
  }
}
```

11.2.4 Path 全局免流控白名单

背景信息

在添加集群的Path全局免流控白名单时，执行命令涉及的配置参数如下：

表 11-12 Path 全局免流控白名单的配置参数说明

配置名	类型	说明
flowcontrol.path.white_list	List<String>	Path全局免流控白名单，配置的路径不参与内存、CPU流控和一键断流，IP流控除外。 支持最大配置10个Path，每个Path最大长度限制小于32。 默认值为为空。 **说明** 一般不建议配置此值，仅在自定义插件时根据业务需求配置。

操作步骤

1. 登录云搜索服务管理控制台。
2. 在“集群管理”页面选择目标集群，单击操作列“Kibana”，登录Kibana界面。
3. 单击左侧导航栏的“Dev Tools”，执行命令添加Path全局免流控白名单。

```
PUT _cluster/settings
{
  "persistent": {
    "flowcontrol.path.white_list": "xxxx"
  }
}
```

11.2.5 请求采样统计

背景信息

开启请求采样统计可以记录访问节点的IP地址和数量，同时可以采样请求的Path，记录请求URL和Body，用于追踪访问量大的客户端IP地址和请求Path。

在开启或关闭集群的请求采样统计时，执行命令涉及的配置参数如下：

表 11-13 请求采样统计的配置参数说明

配置名	类型	说明
flowcontrol.statics.enabled	Boolean	请求采样统计开关。开启请求采样统计对节点性能会有影响。 取值范围：true、false 默认值：false

配置名	类型	说明
flowcontrol.statics.threshold	Integer	统计最近时间访问的请求数量。配置为100，表示会统计出最近访问最多的100个IP地址和基于采样统计的访问最多的100个URL。 最小值：10 最大值：1000 默认值：100
flowcontrol.statics.sample_frequency	Integer	Path采样频率。配置为100，表示每100个请求采样统计一次。 最小值：50 默认值：100

说明

- IP统计和URL采样统计基于访问时间缓存策略，节点会记录最近访问的IP和请求URL，如果缓存空间达到设置的阈值（flowcontrol.statics.threshold配置值），访问时间距离现在最久的记录将被清除掉。
- URL采样统计当前基于URL hash值确认访问Path的一致性。

操作步骤

1. 登录云搜索服务管理控制台。
2. 在“集群管理”页面选择目标集群，单击操作列“Kibana”，登录Kibana界面。
3. 单击左侧导航栏的“Dev Tools”，执行命令开启或关闭请求采样统计。
   - **开启采样统计**

     ```
     PUT /_cluster/settings
     {
       "persistent": {
         "flowcontrol.statics.enabled": true,
         "flowcontrol.statics.threshold": 100,
         "flowcontrol.statics.sample_frequency": 50
       }
     }
     ```

   - **关闭采样统计**

     ```
     PUT /_cluster/settings
     {
       "persistent": {
         "flowcontrol.statics.enabled": false
       }
     }
     ```

11.2.6 流控控制

流量控制提供单独的接口查看节点的流量控制情况。

1. 登录云搜索服务管理控制台。
2. 在“集群管理”页面选择目标集群，单击操作列“Kibana”，登录Kibana界面。
3. 单击左侧导航栏的“Dev Tools”，执行命令查询流量控制情况。

- 查看所有节点的流量控制情况
  ```
  GET /_nodes/stats/filter
  ```
- 查看某个具体节点的流量控制情况
  ```
  GET /_nodes/{nodeId}/stats/filter
  ```
  {nodeId}为需要查看流量控制的节点ID。

响应示例：

```
{
  "_nodes" : {
    "total" : 1,
    "successful" : 1,
    "failed" : 0
  },
  "cluster_name" : "css-flowcontroller",
  "nodes" : {
    "ElBRNCMbTj6L1C-Wke-Dnw" : {
      "name" : "css-flowcontroller-ess-esn-1-1",
      "host" : "10.0.0.133",
      "timestamp" : 1613979513747,
      "flow_control" : {
        "transport" : {
          "concurrent_req" : 0,
          "rejected_concurrent" : 0,
          "rejected_new" : 0,
          "rejected_deny" : 0
        },
        "http" : {
          "concurrent_req" : 0,
          "rejected_concurrent" : 0,
          "rejected_new" : 0,
          "rejected_deny" : 0
        },
        "memory" : {
          "memory_allow" : 41,
          "memory_rejected" : 0
        },
        "cpu": {
          "rejected_cpu" : 0
        }
        "ip_address" : [
          {
            "ip" : "/10.0.0.198",
            "count" : 453
          },
          {
            "ip" : "/198.19.49.1",
            "count" : 42
          }
        ],
        "url_sample" : [
          {
            "url" : "/*/_search?pretty=true",
            "method" : "GET",
            "remote_address" : "/10.0.0.198:16763",
            "count" : 1
          }
        ]
      }
  }
}
```

返回值以Node级别分开，http记录并发和新建连接数据统计，memory记录内存流控统计，ip_address记录最近最多访问的客户端IP，url_sample记录采样的最近最多请求URL。cpu记录CPU流控统计。

表 11-14 响应参数说明

参数名	说明
concurrent_req	节点实际的TCP连接数据信息， 没有开启流控这个配置也会记录，参考GET /_nodes/stats/http接口current_open值， 但是会比这个值小，这里忽略了白名单IP和内部节点IP。
rejected_concurrent	HTTP流控开启生效，关闭后不清零，开启流控期间拒绝的并发连接数。
rejected_new	HTTP流控开启生效，关闭后不清零，开启流控期间拒绝的新建连接数。
rejected_deny	HTTP流控开启生效，关闭后不清零，配置黑名单拒绝的请求数。
memory_allow	内存流控开启生效，关闭后不清零，内存流控允许的请求数，触发内存流控后允许的请求数量，allow_path白名单中通过请求不会被记录，如果allow_path配置为“**”，所有请求都不会被记录。
memory_rejected	内存流控开启生效，关闭后不清零，内存流控拒绝的请求数，触发内存流控后拒绝的请求数量，allow_path白名单中通过请求不会被记录，如果allow_path配置为“**”，所有请求都不会被记录。
rejected_cpu	CPU流控开启生效，关闭后不清零，超过CPU流控阈值拒绝的请求数。
ip_address	IP地址统计，基于配置值统计节点访问的IP地址和请求数量。参数说明请参见**表11-15**。
url_sample	请求Path采样统计，基于配置时间和采样间隔统计相同请求URL数量。参数说明请参见**表11-16**。

表 11-15 ip_address

参数名	说明
ip	访问节点的源IP地址。
method	对应IP地址的访问次数统计。

表 11-16 url_sample

参数名	说明
url	请求的采样统计，记录访问节点的请求URL。
method	对应请求Path的方法。

参数名	说明
remote_address	请求对应的源IP地址和端口。
count	对应请求Path的采样统计次数。

11.2.7 访问日志

背景信息

流量控制提供两种方式查看访问日志。

- 一种是提供单独的API开启和查看访问日志，API参数配置记录访问日志时间和大小，访问日志内容通过Rest接口返回。
- 一种是通过日志打印的方式记录访问日志，开启后用户的访问日志会以文件的方式打印到后端日志中，用户通过查看日志文件查看访问日志。

开启访问日志会影响集群性能。

在开启或关闭访问日志时，执行命令涉及的配置参数如下：

表 11-17 访问日志的配置参数说明

配置名	类型	说明
duration_limit	String	访问日志记录时间。 取值范围：10~120 单位：s 默认值：30
capacity_limit	String	访问日志记录大小。统计开启访问日志后记录的请求大小，当统计的大小大于该配置值，访问日志记录终止。 取值范围：1~5 单位：MB 默认值：1

说明

duration_limit和capacity_limit只要有一个参数达到阈值，访问日志记录就会停止。

操作步骤

1. 登录云搜索服务管理控制台。
2. 在“集群管理”页面选择目标集群，单击操作列“Kibana”，登录Kibana界面。
3. 单击左侧导航栏的“Dev Tools”，执行命令开启访问日志。
   - **开启集群所有节点的访问日志**
     ```
     PUT /_access_log?duration_limit=30s&capacity_limit=1mb
     ```


- **开启集群中某一节点的访问日志**
  ```
  PUT /_access_log/{nodeId}?duration_limit=30s&capacity_limit=1mb
  ```
  {nodeId}为需要开启访问日志的节点ID。

4. 执行命令查看访问日志API。
   - 查看集群所有节点的访问日志API
     ```
     GET /_access_log
     ```
   - 查看集群中某一节点的访问日志API
     ```
     GET /_access_log/{nodeId}
     ```
     {nodeId}为需要开启访问日志的节点ID。

   响应示例：

```
{
  "_nodes" : {
    "total" : 1,
    "successful" : 1,
    "failed" : 0
  },
  "cluster_name" : "css-flowcontroller",
  "nodes" : {
    "8x-ZHu-wTemBQwpcGivFKg" : {
      "name" : "css-flowcontroller-ess-esn-1-1",
      "host" : "10.0.0.98",
      "count" : 2,
      "access" : [
        {
          "time" : "2021-02-23 02:09:50",
          "remote_address" : "/10.0.0.98:28191",
          "url" : "/_access/security/log?pretty",
          "method" : "GET",
          "content" : ""
        },
        {
          "time" : "2021-02-23 02:09:52",
          "remote_address" : "/10.0.0.98:28193",
          "url" : "/_access/security/log?pretty",
          "method" : "GET",
          "content" : ""
        }
      ]
    }
  }
}
```

表 11-18 响应参数说明

参数名	说明
name	节点名称。
host	节点对应的IP地址。
count	统计周期内，访问节点的请求数量。
access	统计周期内，访问节点的请求详情。参数说明请参见**表 11-19**。

表 11-19 access

参数名	说明
time	记录请求时间。
remote_address	请求对应的源IP地址和端口。
url	请求的原始URL
method	对应请求Path的方法。
content	对应请求的内容。

5. 执行命令开启或关闭访问日志文件记录。

   访问日志功能提供配置开关可以记录用户的所有访问日志。日志默认会记录到后台的acces_log.log日志文件中。日志文件单个文件最大支持250M，最多保存5个文件。访问日志文件可以通过日志备份到OBS中查看。

   - **开启访问日志文件记录**

     ```
     PUT /_cluster/settings
     {
       "persistent": {
         "flowcontrol.accesslog.enabled": true
       }
     }
     ```

   - **关闭访问日志文件记录**

     ```
     PUT /_cluster/settings
     {
       "persistent": {
         "flowcontrol.accesslog.enabled": false
       }
     }
     ```

11.2.8 CPU 流控

背景信息

CPU流控可以基于当前节点的CPU占用率实现流量控制。

CPU流控通过配置节点的最大CPU占用率来避免流量冲击下节点掉线风险，可以基于流量阈值预估CPU占用率最大值。当节点CPU超过配置阈值后，CPU流控会丢弃节点请求，达到保护集群的目的，节点内流量和elasticsearch monitor类接口不会被流控。

在开启或关闭CPU流控时，执行命令涉及的配置参数如下：

表 11-20 CPU 流控的配置参数说明

配置名	类型	说明
flowcontrol.cpu.enabled	Boolean	CPU流控开关，开启会影响节点访问性能。 取值范围：true、false 默认值：false

配置名	类型	说明
flowcontrol.cpu.percent_limit	Integer	节点最大CPU占用率配置。 取值范围：0~100 默认值：90
flowcontrol.cpu.allow_path	List	CPU流控白名单，CPU流控基于请求path做流控，配置allow_path白名单，请求将不参与CPU流控。 默认值为空。 单path最大支持32个字符，最多支持配置10个请求路径，支持通配符配置。例如，配置“auto_*/_search”将不限制所有的auto_前缀索引的search请求。
flowcontrol.cpu.*.filter_path	String	配置需要进行CPU流控的访问Path，控制单个请求Path的CPU使用阈值。 最大长度32。 例如： "flowcontrol.cpu.search.filter_path": "/index/_search", "flowcontrol.cpu.search.limit": 60, 默认值 "**", 表示匹配所有的路径。如果只配置了单路径limit，没有配置filter_path，表示除去白名单外的所有path都影响。白名单规则优先于单路径规则，如果一个路径同时配置了allow_path和filter_path, 此请求路径会被允许。 例如同时配置了filter_path和allow_path，其中filter_path="abc/_search", allow_path="abc/_search"，此种情况abc/_search将不被流控。
flowcontrol.cpu.*.limit	Integer	配置请求Path的CPU阈值，CPU超过阈值触发流控。 取值范围：0~100 默认值：90

操作步骤

1. 登录云搜索服务管理控制台。
2. 在“集群管理”页面选择目标集群，单击操作列“Kibana”，登录Kibana界面。
3. 单击左侧导航栏的“Dev Tools”，执行命令开启或关闭CPU流控。
   - **开启CPU流控**

```
PUT /_cluster/settings
{
  "persistent": {
    "flowcontrol.cpu.enabled": true,
    "flowcontrol.cpu.percent_limit": 80,
```



```
    "flowcontrol.cpu.allow_path": ["index/_search"]
  }
}
```

- **关闭CPU流控**

```
PUT /_cluster/settings
{
  "persistent": {
    "flowcontrol.cpu.enabled": false
  }
}
```

11.2.9 一键断流

一键断流可以切断节点上除运维接口外的所有流量，用于应对突发流量场景下的集群异常，达到快速恢复集群的目的。

1. 登录云搜索服务管理控制台。
2. 在“集群管理”页面选择目标集群，单击操作列“Kibana”，登录Kibana界面。
3. 单击左侧导航栏的“Dev Tools”，执行命令开启或关闭一键断流。
   - **开启一键断流**

```
PUT /_cluster/settings
{
  "persistent": {
    "flowcontrol.break.enabled": true
  }
}
```

   - **关闭一键断流**

```
PUT /_cluster/settings
{
  "persistent": {
    "flowcontrol.break.enabled": false
  }
}
```

11.3 大查询隔离

11.3.1 背景信息

大查询隔离特性针对查询请求进行独立管理，将高内存、长耗时的查询请求进行隔离，保证节点内存安全。在节点堆内存过高使用时，触发中断控制程序，根据选择的中断策略将其中一条大查询请求进行中断，取消其正在运行的查询任务。

大查询隔离同时设置了全局查询超时特性，用户可实时配置所有查询请求的超时时间，中断超时查询请求。

说明

目前仅7.6.2和7.10.2版本支持大查询隔离特性。

11.3.2 操作步骤

大查询隔离特性和全局超时特性默认关闭，用户可根据需要实时配置，配置后立即生效。以下是详细的配置方法：

1. 登录云搜索服务管理控制台。
2. 在“集群管理”页面，选择待配置大查询隔离的集群，单击操作列“Kibana”，登录Kibana界面。

3. 在Kibana的左侧导航中选择“Dev Tools”，执行如下命令开启大查询隔离和全局超时的特性开关。

```
PUT _cluster/settings
{
  "persistent": {
    "search.isolator.enabled": true,
    "search.isolator.time.enabled": true
  }
}
```

以上两个开关是独立功能，分别具有以下独立的参数配置：

表 11-21 大查询隔离和全局超时的参数配置

特性开关	配置参数	参数说明
search.isolator.enabled	search.isolator.memory.task.limit search.isolator.time.management	单个分片查询任务被定义为大查询任务的阈值。
	search.isolator.memory.pool.limit search.isolator.memory.heap.limit search.isolator.count.limit	触发隔离池内查询任务中断的阈值。 **说明** 参数“search.isolator.memory.heap.limit”定义了节点实际堆内存的使用限制，包括写入和查询等操作，超过限制时将选取隔离池内的大查询任务进行中断。
	search.isolator.strategy search.isolator.strategy.ratio	中断隔离池中某一条查询任务的选取策略。
search.isolator.time.enabled	search.isolator.time.limit	全局查询任务超时设置。

4. 大查询隔离和全局超时的分别具有独立的参数配置，可以根据实际场景执行不同的命令进行配置。
   - 单个分片查询任务被定义为大查询任务的阈值。

```
PUT _cluster/settings
{
  "persistent": {
    "search.isolator.memory.task.limit": "50MB",
    "search.isolator.time.management": "10s"
  }
}
```


表 11-22 参数说明

参数名	数据类型	说明
search.isolator.memory.task.limit	String	查询任务用于聚合等操作向节点申请的大内存，申请内存超过此阈值将进行隔离观察。 取值范围：0b~节点最大堆内存 默认值：50MB **说明** 可以通过如下命令查询集群堆内存使用情况和最大值。 **GET _cat/nodes?&h=id,ip,port,r,ramPercent,ramCurrent,heapMax,heapCurrent**
search.isolator.time.management	String	查询任务创建至今的时长（即开始使用集群资源进行查询），超过阈值将被隔离观察。 取值范围：≥ 0ms 默认值：10s

- 触发隔离池内查询任务中断的阈值。

```
PUT _cluster/settings
{
  "persistent": {
    "search.isolator.memory.pool.limit": "50%",
    "search.isolator.memory.heap.limit": "90%",
    "search.isolator.count.limit": 1000
  }
}
```

表 11-23 参数说明

参数名	数据类型	说明
search.isolator.memory.pool.limit	String	当前节点最大堆内存百分比，当隔离池所有大查询任务申请的内存超过此阈值将触发中断控制程序，取消执行隔离池其中一条大查询任务。 取值范围：0.0~100.0% 默认值：50%
search.isolator.memory.heap.limit	String	当前节点堆内存的实际使用阈值，当节点堆内存使用超过阈值百分比时触发中断控制程序，取消执行隔离池其中一条大查询任务。 取值范围：0.0~100.0% 默认值：90%

参数名	数据类型	说明
search.isolator.count.limit	Integer	当前节点隔离池的大查询任务数阈值，被观察的查询任务数超过此阈值将触发中断控制程序，不再接受新的大查询。若继续触发大查询请求，则直接取消该请求。 取值范围：10~50000 默认值：1000

说明

根据业务设置“search.isolator.memory.pool.limit”，“search.isolator.count.limit”参数时，可结合“search.isolator.memory.task.limit”，“search.isolator.time.management”两个参数控制查询任务进入到隔离池的数量。

- 中断隔离池中某一条查询任务的选取策略。

```
PUT _cluster/settings
{
  "persistent": {
    "search.isolator.strategy": "fair",
    "search.isolator.strategy.ratio": "0.5%"
  }
}
```

参数名	数据类型	说明
search.isolator.strategy	String	触发中断控制程序时大查询选取的策略。根据策略选取一条查询进行中断。 **说明** 大查询隔离池每秒检查一次，直至堆内存下降到安全范围。 取值范围：fair、mem-first、time-first ● mem-first策略是指触发中断时，选取隔离池中堆内存使用最大的一条查询任务进行中断。 ● time-first策略是指触发中断时，选取隔离池中已运行时间最长的一条查询任务进行中断。 ● fair策略是综合考虑内存和时间两种因素，若分片查询的堆内存申请大小相差不超过“最大堆内存 乘 search.isolator.strategy.ratio”的大小，则认为时间较长的查询更应该中断。否则认为堆内存使用较大的查询更应该中断。 默认值：fair

参数名	数据类型	说明
search.isolator.strategy.ratio	String	fair策略的阈值，仅当“search.isolator.strategy”值为“fair”生效。综合考虑大查询的运行时间及内存，当大查询任务内存相差不超过此阈值时，考虑选取运行时间长的大查询进行中断。当查询内存相差超过此阈值时，选取大内存查询任务进行中断。 取值范围：0.0-100.0% 默认值：1%

- 全局查询任务超时设置。

```
PUT _cluster/settings
{
  "persistent": {
    "search.isolator.time.limit": "120s"
  }
}
```

参数名	数据类型	说明
search.isolator.time.limit	String	当全局查询超时功开启时，所有已创建的查询任务超过此时长将被取消执行。 取值范围：≥ 0ms 默认值：120s

11.4 索引监控

11.4.1 背景信息

索引监控提供了丰富的监控指标，用以监控集群索引的运行状况和变化趋势，衡量业务使用情况，同时可以针对可能存在的风险及时处理，保障集群的稳定运行。

索引监控会采集索引的stats信息保存到集群的监控索引中(monitoring-eye-css-[yyyy-mm-dd])，索引默认保存一周。

目前仅7.6.2和7.10.2版本支持索引监控能力。

11.4.2 启用索引监控

1. 登录云搜索服务管理控制台。
2. 在“集群管理”页面，选择需要启用索引监控的集群，单击操作列“Kibana”，登录Kibana界面。
3. 单击左侧导航栏的“Dev Tools”，执行如下命令打开索引监控开关：

```
PUT _cluster/settings
{
  "persistent": {
    "css.monitoring.index.enabled": "true"
```



```
  }
}
```

4. （可选）如果需要监控单个索引，可以在Kibana的“Dev Tools”执行如下命令：

```
PUT _cluster/settings
{
  "persistent": {
    "css.monitoring.index.enabled": "true",
    "css.monitoring.index.interval": "30s",
    "css.monitoring.index.indices": ["index_name"],
    "css.monitoring.history.duration": "3d"
  }
}
```

表 11-24 配置参数说明

参数名	数据类型	说明
css.monitoring.index.enabled	Boolean	索引监控控制开关，设置为true将打开集群索引监控功能。 默认值：false
css.monitoring.index.interval	Time	索引监控的监控数据采集时间间隔。 最小值：1s 默认值：10s
css.monitoring.index.indices	String	索引监控的索引名称，默认监控所有索引，可以配置监控单个索引，也可以配置通配符监控某一类索引。 例如： ● “"css.monitoring.index.indices": ["index_name"]”表示只监控“index_name”索引。 ● “"css.monitoring.index.indices": ["log_*"]”表示监控以“log_”开头的索引。 ● “"css.monitoring.index.indices": ["index1", "index2"]”表示监控“index1”,"index2"两个索引。 默认值：*（表示监控所有索引）
css.monitoring.history.duration	Time	监控数据存储的索引保留时间，默认保存一周。 最小值：1d 默认值：7d

须知

索引监控不会监控monitoring-eye-css-* 开始的索引，避免使用的索引名称匹配到监控索引。

11.4.3 查看索引读写流量

索引监控提供了查询接口，方便查询一段时间内的索引读写流量。

前提条件

已创建好集群，且已启用索引监控。

操作步骤

1. 登录云搜索服务管理控制台。
2. 在“集群管理”页面，选择已创建的集群，单击操作列“Kibana”，登录Kibana界面。
3. 单击左侧导航栏的“Dev Tools”，执行如下命令查看索引读写流量：
   - 查看所有索引读写流量
     ```
     GET  /_cat/monitoring
     ```
   - 查看某一索引的读写流量
     ```
     GET  /_cat/monitoring/{indexName}
     ```
     {indexName}为需要查看读写流量的索引名称。
   - 查看索引不同时间段的读写流量
     ```
     GET _cat/monitoring?begin=1650099461000
     GET _cat/monitoring?begin=2022-04-16T08:57:41
     GET _cat/monitoring?begin=2022-04-16T08:57:41&end=2022-04-17T08:57:41
     ```

     表 11-25 请求参数说明

参数名	是否必选	说明
begin	否	查看监控的起始时间，UTC时间，默认是当前时间的前5分钟。 支持时间格式：strict_date_optional_time|epoch_millis 默认值：当前时间减去5分钟。
end	否	查看监控的结束时间，UTC时间，默认是当前时间。 支持时间格式： strict_date_optional_time|epoch_millis 默认值：当前时间。

说明

不支持查看系统索引，以“.”开头的索引是系统索引。

返回信息示例：

```
index   begin                   end                     status pri rep init unassign docs.count docs.deleted store.size
pri.store.size delete.rate indexing.rate search.rate
test 2022-03-25T09:46:53.765Z 2022-03-25T09:51:43.767Z yellow  1   1  0    1     9         0
5.9kb        5.9kb         0/s           0/s          0/s
```

表 11-26 返回信息的参数说明

参数名	说明
index	索引名称。
begin	查看监控数据的起始时间。
end	查看监控数据的结束时间。
status	查询监控时间间隔内的索引状态。
pri	查询监控时间间隔内的索引的shard数量。
rep	查询监控时间间隔内的索引副本数量。
init	查询监控时间间隔内的索引的初始化数量。
unassign	查询监控时间间隔内的索引的未分配数量。
docs.count	查询监控时间间隔内的文档数量。
docs.deleted	查询监控时间间隔内的文档删除数量。
store.size	查询监控时间间隔内存储的索引大小。
pri.store.size	查询监控时间间隔内的索引主分片的大小。
delete.rate	监控时间间隔内的索引每秒删除数量。
indexing.rate	监控时间间隔内的索引每秒写入数量。
search.rate	监控时间间隔内的索引每秒查询数量。

11.5 监控增强

11.5.1 P99 时延监控

背景信息

Elasticsearch社区针对search请求的监控都是平均时延，无法有效反应集群的实际search情况，此特性新增对集群search请求的P99时延监控。

前提条件

P99时延监控目前仅支持7.6.2和7.10.2版本集群。

获取监控信息

1. 登录云搜索服务管理控制台。
2. 在“集群管理”页面，选择需要启用索引监控的集群，单击操作列“Kibana”，登录Kibana界面。
3. 在左侧导航栏，选择“Dev Tools”，执行以下命令获取当前集群的P99时延：

```
GET /search/stats/percentile
```


返回样例如下：

```
{
  "overall" : {
    "1.0" : 2.0,
    "5.0" : 2.0,
    "25.0" : 6.5,
    "50.0" : 19.5,
    "75.0" : 111.0,
    "95.0" : 169.0,
    "99.0" : 169.0,
    "max" : 169.0,
    "min" : 2.0
  },
  "last_one_day" : {
    "1.0" : 2.0,
    "5.0" : 2.0,
    "25.0" : 6.5,
    "50.0" : 19.5,
    "75.0" : 111.0,
    "95.0" : 169.0,
    "99.0" : 169.0,
    "max" : 169.0,
    "min" : 2.0
  },
  "latest" : {
    "1.0" : 26.0,
    "5.0" : 26.0,
    "25.0" : 26.0,
    "50.0" : 26.0,
    "75.0" : 26.0,
    "95.0" : 26.0,
    "99.0" : 26.0,
    "max" : 26.0,
    "min" : 26.0
  }
}
```

说明

- 其中“overall”表示集群从启动到当前时间的统计值，“last_one_day”表示最近一天的统计值，“latest”表示从上次重置到当前时间的统计值。
- P99时延的计算是近似值，不提供精确值，越靠近两端的统计值越准确，即99%的时延比50%的时延更准确。
- 如果重启集群，P99时延数据将被清空，P99时延数据将从集群重启成功后重新计算。

其他操作

- 自定义百分百数值。

  您可以自行指定百分百数值：

  ```
  GET /search/stats/percentile
  {
    "percents": [1, 50, 90]
  }
  ```

- 重置latest统计值。

  您可以执行以下命令重置latest统计值：

  ```
  POST /search/stats/reset
  ```

  返回样例：

  ```
  {
    "nodes" : {
      "css-c9c8-ess-esn-1-1" : "ok"
    }
  }
  ```


11.5.2 Http 状态码监控

背景信息

外部通过HTTP访问Elasticsearch都会返回response和相应的状态码，开源Elasticsearch服务端没有对状态码进行统计，无法准确知道调用ES接口的实际状态。用户无法通过监控知道整个集群的请求情况。Http状态码监控提供监控集群的Http状态码的能力。

前提条件

Http状态码监控目前仅7.6.2和7.10.2版本集群支持。

获取状态码

1. 登录云搜索服务管理控制台。
2. 在“集群管理”页面，选择需要启用索引监控的集群，单击操作列“Kibana”，登录Kibana界面。
3. 在左侧导航栏，选择“Dev Tools”。
4. 在Dev Tools的Console界面中执行根据集群版本执行对应的命令。
   - 7.6.2版本集群，请执行以下命令获取状态码统计：
     ```
     GET /_nodes/http_stats
     ```
     返回样例:
     ```
     {
       "_nodes" : {
         "total" : 1,
         "successful" : 1,
         "failed" : 0   },
        "cluster_name" : "css-8362",
        "nodes" : {
         "F9IFdQPARaOJI7oL7HOXtQ" : {
           "http_code" : {
             "200" : 114,
             "201" : 5,
             "429" : 0,
             "400" : 7,
             "404" : 0,
             "405" : 0
           }
         }
       }
     }
     ```
   - 7.10.2版本集群，请执行以下命令获取状态码统计：
     ```
     GET _nodes/stats/http
     ```
     返回样例:
     ```
     {
     // ...
       "cluster_name" : "css-2985",
       "nodes" : {
     // ...
        "omvR9_W-TsGApraMApREjA" : {

     // ...
         "http" : {
           "current_open" : 4,
           "total_opened" : 37,
           "http_code" : {
             "200" : 25,
     ```



```
        "201" : 7,
        "429" : 0,
        "400" : 3,
        "404" : 0,
        "405" : 0
      }
    }
  }
 }
}
```


12 监控

12.1 支持的监控指标

功能说明

本节定义了云搜索服务上报云监控服务的监控指标的命名空间，监控指标列表和维度定义。用户可以通过云监控服务提供管理控制台或API接口来检索云搜索服务产生的监控指标和告警信息。

命名空间

SYS.ES

监控指标

- 监控的**指标ID**、**指标名称**、**指标含义**以及**取值范围**参见**表12-1**。
- 监控的**测量对象**：CSS集群
- 监控的**监控周期**（原始指标）：1分钟

说明

累计值：从节点启动时开始叠加数值，当节点重启后清零重新累计。

表 12-1 云搜索服务支持的监控指标

指标ID	指标名称	指标含义	取值范围
status	集群健康状态	该指标用于统计测量监控对象的状态。	0,1,2,3 0：集群是100%可用的。 1：数据是完整的，部分副本缺失。高可用性在某种程度上弱化，存在风险，请及时关注集群情况。 2：数据缺失，集群使用时将出现异常。 3：没有获取到集群状态。
disk_util	磁盘使用率	该指标用于统计测量对象的磁盘使用率。 单位：百分比	0-100%
max_jvm_heap_usage	最大JVM堆使用率	CSS集群中各个节点的JVM堆使用率的最大值。 单位：百分比。	0-100%
max_jvm_young_gc_time	最大JVM Young GC耗时	CSS集群中各个节点的JVM Young GC耗时累计值的最大值。 单位：ms	≥ 0 ms
max_jvm_young_gc_count	最大JVM Young GC次数	CSS集群中各个节点的JVM Young GC次数累计值的最大值。	≥ 0
max_jvm_old_gc_time	最大JVM Old GC耗时	CSS集群中各个节点的JVM Old GC耗时累计值的最大值。 单位：ms	≥ 0 ms
max_jvm_old_gc_count	最大JVM Old GC次数	CSS集群中各个节点的JVM Old GC次数累计值的最大值。	≥ 0
total_fs_size	文件系统总大小	CSS集群的文件系统总大小。 单位：byte	≥ 0 bytes
free_fs_size	文件系统可用大小	CSS集群的文件系统可用大小。 单位：byte	≥ 0 bytes

指标ID	指标名称	指标含义	取值范围
max_cpu_usage	最大CPU利用率	CSS集群中各个节点的CPU利用率的最大值。 单位：百分比	0-100%
max_cpu_time_of_jvm_process	最大JVM进程使用的CPU时间	CSS集群中各个节点JVM进程使用CPU的时间累计值的最大值。 单位：ms	≥ 0 ms
max_virtual_memory_size_of_jvm_process	最大JVM进程使用的虚拟内存大小	CSS集群中各个节点JVM进程可使用的虚拟内存大小的最大值。 单位：byte	≥ 0 bytes
max_current_opened_http_count	最大当前打开的Http连接数	CSS集群中各个节点打开且尚未关闭的Http连接数的最大值。	≥ 0
max_total_opened_http_count	最大全部打开的Http连接数	CSS集群中各个节点打开过的Http连接数累计值的最大值。	≥ 0
indices_count	索引数量	CSS集群的索引数量。	≥ 0
total_shards_count	分片数量	CSS集群的分片数量。	≥ 0
primary_shards_count	主分片数量	CSS集群的主分片数量。	≥ 0
docs_count	文档数量	CSS集群的文档数量。	≥ 0
docs_deleted_count	被删除的文档数量	CSS集群的被删除的文档数量。	≥ 0
nodes_count	节点数量	CSS集群的节点数量。	≥ 0
data_nodes_count	数据节点数量	CSS集群的数据节点数量。	≥ 0
coordinating_nodes_count	协调节点数量	CSS集群的协调节点数量。	≥ 0
master_nodes_count	Master节点数量	CSS集群的Master节点数量。	≥ 0

指标ID	指标名称	指标含义	取值范围
ingest_nodes_count	Client节点数量	CSS集群的Client节点数量。	≥ 0
max_load_average	最大节点Load值	CSS集群中各个节点在操作系统中1分钟平均排队任务数的最大值。	≥ 0
avg_cpu_usage	平均CPU使用率	CSS集群中各节点CPU利用率的平均值。 单位：百分比	0-100%
avg_load_average	平均节点Load值	CSS集群中各节点在操作系统中1分钟平均排队任务数的平均值。	≥ 0
avg_jvm_heap_usage	平均JVM堆使用率	CSS集群中各节点JVM堆内存使用率的平均值。 单位：百分比	0-100%
max_open_file_descriptors	已打开的最大文件描述符数	CSS集群中各个节点已打开的文件描述符数的最大值。	≥ 0
avg_open_file_descriptors	已打开的平均文件描述符数	CSS集群中各节点已打开的文件描述符数的平均值。	≥ 0
sum_max_file_descriptors	最大允许的文件描述符数	CSS集群中各节点最大允许的文件描述符数之和。	≥ 0
sum_open_file_descriptors	已打开的文件描述符数	CSS集群中各节点已打开的文件描述符数之和。	≥ 0
sum_thread_pool_write_queue	Write队列中总排队任务数	写入线程池中的排队任务数。	≥ 0
sum_thread_pool_search_queue	Search队列中总排队任务数	CSS集群中各节点在搜索线程池中的排队任务数之和。	≥ 0
sum_thread_pool_force_merge_queue	ForceMerge队列中总排队任务数	CSS集群中各节点在强制合并线程池中的排队任务数之和。	≥ 0
sum_thread_pool_write_rejected	Write队列中总的已拒绝任务数	CSS集群中各节点在写入线程池中的已拒绝任务数之和。	≥ 0

指标ID	指标名称	指标含义	取值范围
sum_thread_pool_search_rejected	Search队列中总的已拒绝任务数	CSS集群中各节点在搜索线程池中的已拒绝任务数之和。	≥ 0
sum_thread_pool_force_merge_rejected	Forcemerge队列中总的已拒绝任务数	CSS集群中各节点在强制合并线程池中的已拒绝任务数之和。	≥ 0
max_thread_pool_search_queue	Search队列中最大排队任务数	CSS集群中各个节点在搜索线程池中的排队任务数的最大值。	≥ 0
max_thread_pool_force_merge_queue	ForceMerge队列中最大排队任务数	CSS集群中各个节点在强制合并线程池中的排队任务数的最大值。	≥ 0
sum_thread_pool_write_threads	Write线程池总大小	CSS集群中各节点写入线程池的大小之和。	≥ 0
sum_thread_pool_search_threads	Search线程池总大小	CSS集群中各节点搜索线程池的大小之和。	≥ 0
sum_thread_pool_force_merge_threads	ForceMerge线程池总大小	CSS集群中各节点强制合并线程池的大小之和。	≥ 0
avg_thread_pool_write_queue	Write队列中平均排队任务数	CSS集群中各节点在写入线程池中的排队任务数的平均值。	≥ 0
avg_thread_pool_search_queue	Search队列中平均排队任务数	CSS集群中各节点在搜索线程池中的排队任务数的平均值。	≥ 0
avg_thread_pool_force_merge_queue	ForceMerge队列中平均排队任务数	CSS集群中各节点在强制合并线程池中的排队任务数的平均值。	≥ 0
avg_thread_pool_search_threads	Search线程池平均大小	CSS集群中各节点搜索线程池的大小的平均值。	≥ 0
avg_thread_pool_write_threads	Write线程池平均大小	CSS集群中各节点写入线程池的大小的平均值。	≥ 0
avg_thread_pool_force_merge_threads	ForceMerge线程池平均大小	CSS集群中各节点强制合并线程池的大小的平均值。	≥ 0

指标ID	指标名称	指标含义	取值范围
avg_thread_pool_write_rejected	Write队列中平均已拒绝任务数	CSS集群中各节点写入线程池中的已拒绝任务数的平均值。	≥ 0
min_free_fs_size	最小可用存储空间	CSS集群中各个节点可用存储空间的最小值。 单位：byte	≥ 0 bytes
avg_jvm_old_gc_count	JVM老年代平均GC次数	CSS集群中各个节点“老年代”垃圾回收的运行次数的累计值的平均值。	≥ 0
avg_jvm_old_gc_time	JVM老年代平均GC时间	CSS集群中各个节点执行“老年代”垃圾回收所花费的时间累计值的平均值。 单位：ms	≥ 0 ms
avg_jvm_young_gc_count	JVM年轻代平均GC次数	CSS集群中各个节点“年轻代”垃圾回收的运行次数的累计值的平均值。	≥ 0
avg_jvm_young_gc_time	JVM年轻代平均GC时间	CSS集群中各个节点执行“年轻代”垃圾回收所花费的时间累计值的平均值。 单位：ms	≥ 0 ms
avg_max_file_descriptors	最大允许的文件描述符数-平均值	CSS集群中各节点最大允许的文件描述符数的平均值。	≥ 0
avg_mem_free_in_bytes	平均可用内存空间	CSS集群中各节点未使用的内存容量的平均值。 单位：byte	≥ 0 bytes
avg_mem_free_percent	平均可用内存比例	CSS集群中各节点未使用的内存比例的平均值。 单位：百分比	0-100%
avg_mem_used_in_bytes	平均已用内存空间	CSS集群中各节点已使用的内存容量的平均值。 单位：byte	≥ 0 bytes

指标ID	指标名称	指标含义	取值范围
avg_mem_used_percent	平均已用内存比例	CSS集群中各节点已使用的内存比例的平均值。 单位：百分比	0-100%
max_mem_free_in_bytes	最大可用内存空间	CSS集群中各个节点未使用的内存容量的最大值。 单位：byte	≥ 0 bytes
max_mem_free_percent	最大可用内存比例	CSS集群中各个节点未使用的内存比例的最大值。 单位：百分比	0-100%
max_mem_used_in_bytes	最大已用内存空间	CSS集群中各个节点已使用的内存容量的最大值。 单位：byte	≥ 0 bytes
max_mem_used_percent	最大已用内存比例	CSS集群中各个节点已使用的内存比例的最大值。 单位：百分比	0-100%
sum_jvm_old_gc_count	JVM老年代总GC次数	CSS集群中各个节点“老年代”垃圾回收的运行次数的累计值之和。	≥ 0
sum_jvm_old_gc_time	JVM老年代总GC时间	CSS集群中各个节点执行“老年代”垃圾回收所花费的时间累计值之和。 单位：ms	≥ 0ms
sum_jvm_young_gc_count	JVM年轻代总GC次数	CSS集群中各个节点“年轻代”垃圾回收的运行次数的累计值之和。	≥ 0
sum_jvm_young_gc_time	JVM年轻代总GC时间	CSS集群中各个节点执行“年轻代”垃圾回收所花费的时间累计值之和。 单位：ms	≥ 0 ms
sum_current_opened_http_count	当前已打开http连接数	CSS集群中各个节点打开且尚未关闭的Http连接数之和。	≥ 0

指标ID	指标名称	指标含义	取值范围
sum_total_opened_http_count	历史已打开http连接数	CSS集群中各个节点打开过的Http连接数累计值之和。	≥ 0
IndexingLatency	平均索引延迟	分片完成索引操作所需的平均时间。 单位：ms	≥ 0 ms
IndexingRate	平均索引速率	入库TPS，集群每秒平均索引操作数。 单位：s。	≥ 0s
SearchLatency	平均查询延迟	分片完成搜索操作所需的平均时间。 单位：ms。	≥ 0 bytes
SearchRate	平均查询速率	查询QPS，集群每秒平均查询操作数。 单位：s	≥ 0/s

维度

表 12-2 维度说明

Key	Value
cluster_id	CSS集群

12.2 配置集群监控

云搜索服务支持通过云监控服务CES对已创建成功的集群进行日常监控。配置集群监控后，就可以在CES管理控制台直观查看集群的监控指标数据。

配置集群监控的操作流程：

1. **配置告警规则**：根据实际业务需要对监控指标设置自定义告警规则，当监控指标超过设置的阈值时，会以邮箱、HTTP、HTTPS等方式通知您。
2. **配置监控对象**：为集群或集群中某个节点配置监控指标。
3. **查看监控指标**：您可以选择不同的监控时间周期，查看监控指标数据变化情况。

前提条件

- 集群处于“可用”或“处理中”状态。
- 集群正常运行时长大于10分钟。

推荐配置的监控指标

- 监控集群的**cpu、jvm使用情况**，推荐重点配置如下监控指标：平均JVM堆使用率、最大JVM堆使用率、平均CPU使用率、最大CPU利用率
- 监控集群的**写入、查询延迟和吞吐量情况**，推荐重点配置如下监控指标：平均索引延迟、平均索引速率、平均查询延迟、平均查询速率
- 监控集群的**写入、查询的排队队列和拒绝情况**，推荐重点配置如下监控指标：Write队列中总排队任务数、Search队列中总排队任务数、Write队列中总的已拒绝任务数、Search队列中总的已拒绝任务数

配置告警规则

1. 登录云监控服务CES管理控制台。
2. 左侧导航栏选择“告警 > 告警规则”，进入告警规则列表页面。
3. 在“资源类型”列，筛选“云搜索服务”，查看是否有满足要求的告警规则。

图 12-1 查看告警规则

如果没有，请参考云监控服务CES的“创建告警规则和通知”章节，新建CSS服务的告警规则。其中，“资源类型”和“维度”参数的填写说明请参见**表12-3**，其他参数可以根据CES服务的参数说明自定义。

表 12-3 告警内容的配置说明

参数	参数解释	配置说明
资源类型	配置告警规则监控的服务名称。	选择**云搜索服务**。
维度	用于指定告警规则对应指标的维度名称。	CSS支持2个维度，根据实际需要选择维度。 • **CSS集群**：以集群维度指定告警规则。 • **CSS集群 - 云服务节点**：以集群中的某个节点维度指定告警规则。

配置监控对象

1. 参考云监控服务CES的“创建监控面板”章节，创建一个监控面板。如果已有监控面板，可以跳过该步骤。
2. 参考云监控服务CES的“添加监控视图”章节，添加CSS监控视图。

   其中，“资源类型”和“维度”参数的填写说明请参见**表12-4**，其他参数可以根据CES服务的参数说明自定义。

表 12-4 监控视图的配置说明

参数	参数解释	配置说明
资源类型	添加监控视图的服务名称。	选择**云搜索服务**。
维度	指定监控的维度名称。	CSS支持2个维度，根据实际需要选择维度。 • **CSS集群**：以集群维度监控。 • **CSS集群 - 云服务节点**：以集群中的某个节点维度监控。

查看监控指标

1. 登录云搜索服务管理控制台。
2. 选择目标集群，单击操作列“监控信息”查看监控指标。

图 12-2 查看监控信息

3. 选择待查看的时间段页签。
4. 查看监控指标数据。

13 审计

13.1 支持云审计的关键操作

通过云审计服务，您可以记录与云搜索服务相关的操作事件，便于日后的查询、审计和回溯。

前提条件

已开通云审计服务。

支持审计的关键操作列表

表 13-1 支持审计的关键操作列表

操作名称	资源类型	事件名称
创建集群	cluster	createCluster
删除集群	cluster	deleteCluster
扩容集群	cluster	roleExtendCluster
重启集群	cluster	rebootCluster
设置集群快照的基础配置	cluster	updateSnapshotPolicy
设置自动创建快照策略	cluster	updateAutoSnapshotPolicy
集群升级	cluster	upgradeCluster
升级重试	cluster	retryAction
手动创建快照	snapshot	createSnapshot
恢复快照	snapshot	restoreSnapshot
删除快照	snapshot	deleteSnapshot

13.2 查看审计日志

在您开启了云审计服务后，系统会记录云搜索服务的相关操作，且控制台保存最近7天的操作记录。本节介绍如何在云审计服务管理控制台查看最近7天的操作记录。

操作步骤

1. 登录云审计服务管理控制台。
2. 在管理控制台左上角单击 图标，选择区域。
3. 在左侧导航栏中，单击“事件列表”，进入“事件列表”页面。
4. 事件列表支持通过筛选来查询对应的操作事件。当前事件列表支持四个维度的组合查询，详细信息如下：
   - 事件来源、资源类型和筛选类型。

     在下拉框中选择查询条件。

     其中筛选类型选择事件名称时，还需选择某个具体的事件名称。

     选择资源ID时，还需输入某个具体的资源ID。

     选择资源名称时，还需选择或手动输入某个具体的资源名称。
   - 操作用户：在下拉框中选择某一具体的操作用户，此操作用户指用户级别，而非租户级别。
   - 事件级别：可选项为“所有事件级别”、“normal”、“warning”、“incident”，只可选择其中一项。
   - 时间范围：可选择查询最近七天内任意时间段的操作事件。
5. 在需要查看的事件左侧，单击 展开该事件的详细信息。
6. 单击需要查看的事件“操作”列的“查看事件”，可以在弹窗中查看该操作事件结构的详细信息。

   更多关于云审计服务事件结构的信息，请参见《云审计服务用户指南》。

14 最佳实践

14.1 集群迁移

14.1.1 迁移方案概述

CSS服务的迁移方案适用于云ES之间的数据迁移、自建ES到云ES之间的数据迁移和第三方友商ES到云ES之间的数据迁移。可以根据实际迁移场景，选择合适的集群迁移方案。

迁移场景

不同数据来源的集群，迁移方案会有差别，本章主要介以下场景的迁移方案。

- 源端为Elasticsearch的集群迁移

  Elasticsearch集群的数据迁移有多种方式可以选择，例如使用Logstash、CDM、OBS备份与恢复、ESM、跨集群复制插件等进行数据迁移。

  - Logstash：Elasticsearch官方提供的数据清洗工具，ELK生态中的一部分，功能强大，可以完成不同数据源和ES数据的迁移，还可以进行数据的清洗和加工。具体操作可以参考**使用Logstash迁移集群数据**。
  - CDM：云服务提供的云迁移工具，实现不同云服务间的集群迁移能力。具体操作可以参考。
  - 备份与恢复：ES提供备份恢复能力，可以把一个集群的数据备份到OBS，在另一个集群恢复数据，完成集群间的数据迁移。具体操作可以参考**使用备份与恢复迁移集群数据**。
- **源端为Kafka/MQ的集群迁移**
- **源端为数据库的集群迁移**

迁移方案

CSS服务支持的迁移方案主要包含备份与恢复、Reindex API、Logstash+ESM和基于数据源同步的迁移方案，各方案的详细信息参见**表14-1**。

其中，基于数据源同步的迁移方案没有明显的限制，且迁移性能方面优于其他3个方案。另外基于数据源同步的迁移方案，在数据同步后随时可割接，更加方便灵活。

表 14-1 迁移方案说明

迁移方案	方案描述	方案限制	迁移性能
备份与恢复	需要有支持s3协议的共享存储，例如OBS桶。先在源端ES集群进行数据备份，再将快照同步到目标集群，最后在目标集群中恢复数据。	• 目的端ES版本≥源端ES版本 • 目的端ES的候选主节点数>源端ES的候选主节点数的一半 • 不支持增量数据同步，需停止更新再进行备份与恢复。	数据迁移速率可配置，理想状态下等于文件拷贝速率。
Reindex API	先配置源端ES与目的端ES互信，然后通过Reindex API进行数据迁移。	• 索引需开启“_source” • 不支持增量数据实时同步，需停止更新再执行API。	支持批量读写，但不支持slicing并发同步。
Logstash +ESM	先申请ECS部署并配置Logstash，然后启动数据迁移。	• 索引需开启“_source” • 不支持增量数据实时同步，需停止更新再启动Logstash。	支持批量读写，支持slicing并发同步。
基于数据源同步的迁移方案	存量数据采用Logstash迁移，增量数据通过流量复制或数据链路自动同步。	无	存量迁移速率同Logstash，增量迁移复用之前工具。

14.1.2 源端为 Elasticsearch

14.1.2.1 使用 Logstash 迁移集群数据

Logstash是ES官方提供的数据迁移工具。

步骤1 申请ECS虚拟机，虚拟机规格建议大于8u16g。

步骤2 在ECS中安装Logstash。

1. 由于Logstash依赖Java，需要先安装JDK。执行如下命令，使用yum安装JDK。
   ```
   yum install java
   yum install python
   ```
2. 下载Logstash。Logstash版本和ES版本不需要强制一致，一般选择和ES版本接近的Logstash版本。

   推荐使用Logstash 7.10.2 OSS版本，下载地址：**https://www.elastic.co/downloads/past-releases/logstash-oss-7-10-2**
3. 执行如下命令，使用yum安装Logstash。
   ```
   yum install logstash-oss-7.10.0-x86_64.rpm
   ```
   其中“*logstash-oss-7.10.0-x86_64.rpm*”替换成实际Logstash的安装包名称。

步骤3 修改Logstash的jvm配置，提升集群数据的迁移效率。

执行如下命令，修改jvm配置。Logstash默认的堆内存是1G，建议修改为集群节点内存的一半。

```
vim /etc/logstash/jvm.options
-Xms4g
-Xmx4g
```

步骤4 修改Logstash的conf配置文件，设置集群迁移配置。

1. 进入Logstash配置文件的目录下“/etc/logstash/conf.d/”。
   ```
   cd /etc/logstash/conf.d/
   ```
2. 创建“logstash-es-es-all.conf”文件。
   ```
   vim logstash-es-es-all.conf
   ```
3. 在文件“logstash-es-es-all.conf”里添加如下配置内容并保存。

   根据实际情况修改“hosts”、“user”、“password”、“index” 等字段。

   ```
   input{
       elasticsearch{
           # 源端集群地址。
           hosts =>  ["http://172.16.xxx.xxx:9200", "http://172.16.xxx.xxx:9200"]
           # 安全集群需要配置登录集群的用户名和密码，非安全集群可以使用“#”注释掉user和password。
           # user => "xxxx"
           # password => "xxxx"
           # 需要迁移的索引列表，以逗号“,”分隔，基于机器实际信息填写，“-.*”表示排除“.”开始的索引。
           index => "abmau_edi*,business_test,goods_deploy*, -.*"
           # 以下三项保持默认即可，包含线程数和迁移数据大小和logstash jvm配置相关。
           docinfo=>true
           # 默认不变，如果需要增加迁移速度可以适当调高以下两个参数，但是需要保证机器配置。
           slices => 3
           size => 3000
       }
   }

   filter {
     # 去掉一些logstash自己加的字段。
     mutate {
       remove_field => ["@timestamp", "@version"]
     }
   }

   output{
       elasticsearch{
           # 目的端集群地址。
           hosts => ["http://10.100.xx.xx:9200", "http://10.100.xx.xx:9200"]
           # 登录目标集群的用户名和密码，没有user和password可以使用“#”注释掉。
           user => "admin"
           password => "*****"
           # 目的端索引名称，以下配置为和源端保持一致。
           index => "%{[@metadata][_index]}"
           # 目的端索引type，以下配置为和源端保持一致。
           document_type => "%{[@metadata][_type]}"
           # 目标端数据的_id，如果不需要保留原_id，可以删除，删除后集群的性能会更好。
           document_id => "%{[@metadata][_id]}"
           ilm_enabled => false
           manage_template => false
       }

       # 调试信息，正式迁移时建议去掉调试信息。
       # stdout { codec => rubydebug { metadata => true }}
   }
   ```

步骤5 启动Logstash迁移集群数据。

1. 执行如下命令启动Logstash，开始迁移数据。
   ```
   /usr/share/logstash/bin/logstash --path.settings /etc/logstash
   ```


2. 查看Logstash日志文件，确认任务进展。Logstash日志目录是“/var/log/logstash/”。
3. 任务启动完毕后，等待数据迁移完成。

----结束

14.1.2.2 使用备份与恢复迁移集群数据

- **云ES之间的数据迁移**，请参见迁移集群。
- **自建ES到云ES之间的数据迁移**和**第三方友商ES到云ES之间的数据迁移**，可以参考本章操作指导。

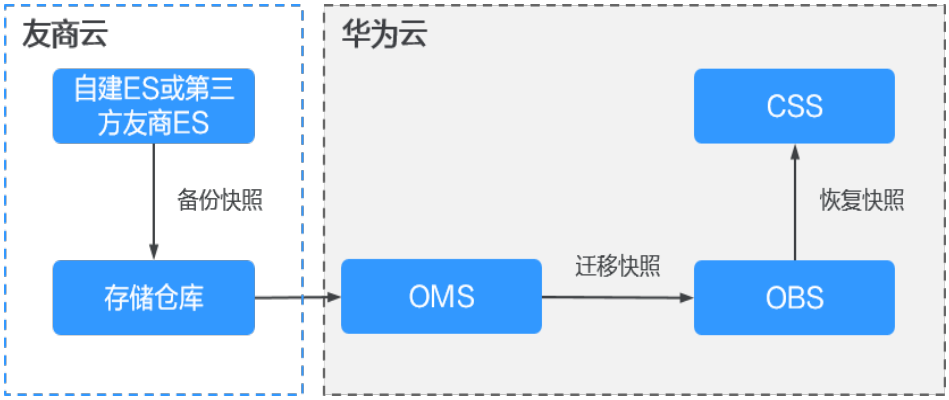
前提条件

- 使用备份与恢复时，需确认如下2点：
  - 目的端ES版本≥源端ES版本
  - 目的端ES的候选主节点数>源端ES的候选主节点数的一半
- 备份与恢复不支持增量数据同步，需停止数据更新后再进行备份。
- CSS服务中已创建好目的端Elasticsearch集群。

迁移流程

当源端是自建ES或第三方友商ES，目的端是CSS服务的ES集群时，集群迁移流程如图14-1所示。

图 14-1 使用备份与恢复迁移的集群迁移流程



操作步骤

步骤1 创建一个支持s3协议的共享存储仓库。

步骤2 在自建或第三方友商Elasticsearch中创建快照备份仓库，用于存放ES快照数据。

例如，在Elasticsearch中创建一个“my_backup”的备份仓库，关联到存储仓库OSS。

```
PUT _snapshot/my_backup
  {
     # 存储仓库类型。
  "type": "oss",
     "settings": {
     # 步骤1中存储仓库的内网访问域名。
     "endpoint": "http://oss-xxx.xxx.com",
```



```
    # 存储仓库的用户ID和密码。
    "access_key_id": "xxx",
    "secret_access_key": "xxx",
    # 步骤1创建的存储仓库的bucket名称。
    "bucket": "patent-esbak",
    # 是否打开快照文件的压缩功能。
    "compress": false,
    # 配置此参数可以限制快照数据的分块大小。当上传的快照数据超过这个数值，数据就会被分块上传到存储仓库中。
    "chunk_size": "1g",
    # 仓库的起始位置，默认是根目录。
    "base_path": "snapshot/"
    }
}
```

步骤3 为自建或第三方友商Elasticsearch创建快照。

- 为所有索引创建快照

  例如，创建一个名为“snapshot_1”的快照。

  ```
  PUT _snapshot/my_backup/snapshot_1?wait_for_completion=true
  ```

- 为指定索引创建快照

  例如，创建一个名为“snapshot_test”的快照，该快照包含索引“patent_analyse”和“patent”。

  ```
  PUT _snapshot/my_backup/snapshot_test
  {
  "indices": "patent_analyse,patent"
  }
  ```

步骤4 查看集群的快照创建进度。

- 执行如下命令可以查看所有快照信息：

  ```
  GET _snapshot/my_backup/_all
  ```

- 执行如下命令可以查看指定快照“snapshot_1”的信息：

  ```
  GET _snapshot/my_backup/snapshot_1
  ```

步骤5 将快照数据从存储仓库迁移到对象存储服务OBS中。

对象存储迁移服务（OMS）支持多种云服务商数据迁移到对象存储服务OBS中。

步骤6 在CSS服务的Elasticsearch集群中创建一个存储仓库关联到OBS，用于恢复自建或第三方友商Elasticsearch的快照数据。

例如，在集群中创建一个“my_backup_all”的存储仓库，关联上一步数据迁移目的端的OBS。

```
PUT _snapshot/my_backup_all/
{
  "type" : "obs",
  "settings" : {
    # OBS的内网访问域名。
    "endpoint" : "obs.xxx.xxx.com",
    "region" : "xxx",
    # 访问OBS的用户名和密码。
    "access_key": "xxx",
    "secret_key": "xxx",
    # OBS的桶名称，和上一步迁移目的端的OBS桶名保持一致。
    "bucket" : "esbak",
    "compress" : "false",
    "chunk_size" : "1g",
    # 注意“snapshot”后面没有/。
    "base_path" : "snapshot",
    "max_restore_bytes_per_sec": "100mb",
    "max_snapshot_bytes_per_sec": "100mb"
  }
}
```


步骤7 在CSS服务的Elasticsearch集群中恢复快照数据。

1. 查看所有快照信息。
   ```
   GET _snapshot
   ```
2. 恢复快照。
   - 恢复某一快照的所有索引。例如恢复名为“snapshot_1”的快照的所有索引数据。
     ```
     POST _snapshot/my_backup_all/snapshot_1/_restore?wait_for_completion=true
     ```
   - 恢复某一快照的部分索引。例如名为“snapshot_1”的快照中只恢复非“.”开头的索引。
     ```
     POST _snapshot/my_backup/snapshot_1/_restore
     {"indices":"*,-.monitoring*,-.security*,-.kibana*","ignore_unavailable":"true"}
     ```
   - 恢复某一快照中的指定索引，并重命名。例如在名为“snapshot_1”的快照中，将索引“index_1”恢复为“restored_index_1”，“index_2”恢复为“restored_index_2”。
     ```
     POST /_snapshot/my_backup/snapshot_1/_restore
     {
       # 只恢复索引“index_1”和“index_2”，忽略快照中的其他索引。
       "indices": "index_1,index_2"
       # 查找正在恢复的索引，该索引名称需要与提供的模板匹配。
       "rename_pattern": "index_(.+)",
       # 重命名查找到的索引。
       "rename_replacement": "restored_index_$1"
     }
     ```

步骤8 查看快照恢复结果。

- 查看所有快照的恢复结果：
  ```
  GET /_recovery/
  ```
- 查看指定索引的快照恢复结果： GET {index_name}/_recovery
  ```
  GET {index_name}/_recovery
  ```

----结束

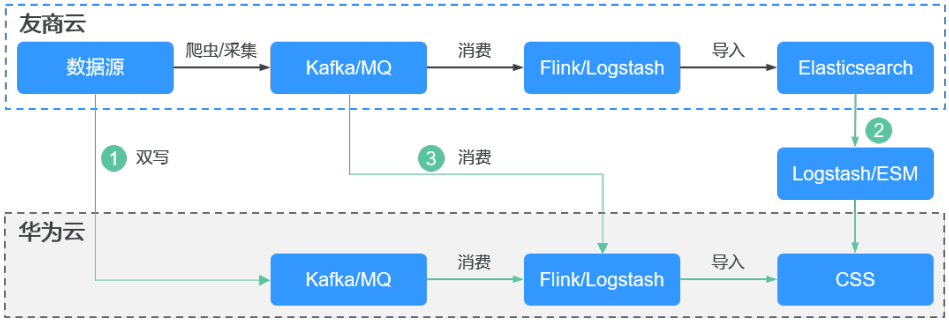
14.1.3 源端为 Kafka/MQ

迁移流程

在泛IOT、新闻、舆情、社交等数据量较大的行业，通常会引入消息中间件（kafka、MQ等）对流量进行削峰填谷，然后借助Flink/Logstash等工具消费数据并进行数据预处理，最终将数据导入到搜索引擎，对外提供搜索服务。

对于这种源端是kafka/MQ的集群，集群迁移流程如图14-2所示。

图 14-2 源端为 Kafka/MQ 的集群迁移流程



该迁移方案具有割接方便和通用性好的优点。

- 割接方便：一旦两个ES集群的数据保持一致后，随时可割接。
- 通用性好：两边均可同步对客户端的ES进行增删改查操作。

操作步骤

步骤1 订阅增量。在kafka/MQ中新建消费组，订阅增量数据。

步骤2 存量数据同步。使用Logstash等工具将源端ES集群中的数据迁移到CSS集群中。若使用Logstash进行数据迁移，可以参考**使用Logstash迁移集群数据**。

步骤3 增量数据同步。待存量数据同步完成之后开启增量消费组，利用ES对数据操作的幂等性，等新的消费组追上之前的消费组时，两边的数据就会追平。

说明

关于日志场景，源端ES集群中的数据并不需要迁移，即可跳过存量数据同步的步骤，待增量数据同步完成后，两边持续同步跑一段时间（例如3天或7天），然后直接割接即可。

----结束

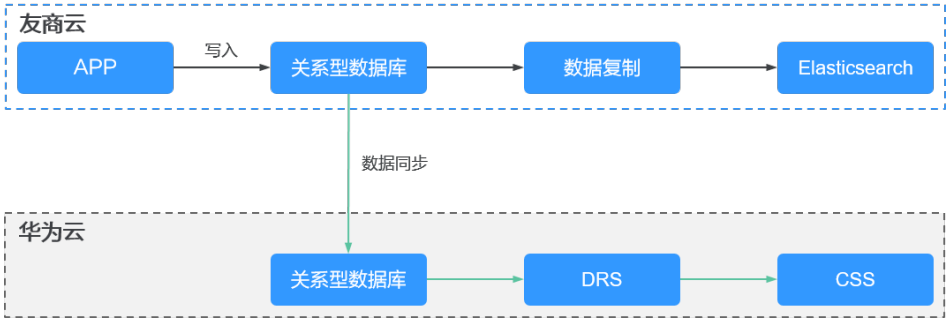
14.1.4 源端为数据库

迁移流程

因为Elasticsearch支持全文检索和Ad Hoc查询，所以经常作为关系型数据库（例如MySQL、GaussDB for MySQL等）的补充，以此提升数据库的全文检索能力和高并发的Ad Hoc查询能力。

对于这种源端是数据库的集群，集群迁移流程如**图14-3**所示。

图 14-3 源端为数据库的集群迁移流程



该迁移方案具有割接方便和通用性好的优点。

- 割接方便：当CSS进入增量同步状态，即可启动业务割接操作。
- 通用性好：两边均可同步对客户端的ES进行增删改查操作。

操作步骤

通过数据复制服务（DRS）可以实现MySQL等关系型数据库之间的数据迁移和同步，支持的数据库类型请参见。

步骤1 打通数据同步链路。在云建立数据库到CSS的同步链路。

步骤2 启动数据库同步。通过DRS或DataX等第三方工具，启动数据同步，将数据库的数据同步到CSS。

----结束

14.2 接入集群

14.2.1 方案概述

Elasticsearch集群支持多种连接方式，根据业务使用的编程语言可以自行选择接入方式。针对CSS服务的3种不同安全模式的集群（非安全模式的集群、安全模式+HTTP协议的集群、安全模式+HTTPS协议的集群），不同客户端的支持情况请参见**表14-2**。

- CSS提供了可视化的Kibana和Cerebro界面用于监控、使用集群。在CSS服务控制台，可以快速接入每个Elasticsearch集群的Kibana和Cerebro。
- 其他客户端也可以接入并使用Elasticsearch集群，如Curl命令行、Java客户端、Python客户端等形式，亦或是使用Hadoop提供的客户端实现更复杂的应用。Elasticsearch官方提供了的Java客户端，包括Rest High Level Client、Rest Low Level Client和Transport Client。建议使用对应Elasticsearch集群版本的Java客户端，否则可能存在兼容性问题。

表 14-2 不同客户端接入集群的支持情况

客户端	非安全模式的集群	安全模式+HTTP协议的集群	安全模式+HTTPS协议的集群
Kibana	3种安全模式的集群都支持，安全模式的集群登录Kibana时需要输入用户名和密码进行安全认证，接入集群的操作请参见**快速访问Elasticsearch集群**。		
Cerebro	3种安全模式的集群都支持，安全模式的集群登录Cerebro时需要输入用户名和密码进行安全认证，接入集群的操作请参见**快速访问Elasticsearch集群**。		
Curl	3种安全模式的集群都支持，接入命令有差别，具体请参见**通过Curl命令行接入集群**。		
Java（Rest High Level Client）	3种安全模式的集群都支持，接入命令有差别，具体请参见**通过Rest High Level Client接入集群**。		
Java（Rest Low Level Client）	3种安全模式的集群都支持，接入命令有差别，具体请参见**通过Rest Low Level Client接入集群**。		
Java（Transport Client）	只支持非安全模式的集群，具体请参见**通过Transport Client接入集群**。	不支持	不支持
Python	3种安全模式的集群都支持，接入命令有差别，具体请参见**通过Python接入集群**。		

客户端	非安全模式的集群	安全模式+HTTP协议的集群	安全模式+HTTPS协议的集群
ES-Hadoop	3种安全模式的集群都支持，接入命令有差别，具体请参见**通过ES-Hadoop实现Hive读写Elasticsearch数据**。		

14.2.2 通过 Curl 命令行接入集群

当CSS集群和弹性云服务器ECS在同一VPC内时，在此ECS中可以通过Curl命令直接访问Elasticsearch 集群，此方法多用于前期调试Elasticsearch的连接性，排查访问集群的客户端是否和Elasticserach节点的网络连通。

准备工作

- CSS集群处于可用状态。
- 已经具备满足如下要求的ECS：
  - ECS的VPC需要与CSS集群在同一个VPC中，即保证网络连通。
  - ECS的安全组需要和CSS集群的安全组相同。

    如果不同，请修改ECS安全组或者配置ECS安全组的出入规则，允许集群所有安全组的访问。修改操作请参见《虚拟私有云用户指南》。

  ECS的使用请参见《弹性云服务器用户指南》。

操作步骤

1. 获取集群的内网访问地址。访问集群时，需要输入内网访问地址。
   a. 在云搜索服务管理控制台，单击左侧导航栏的“集群管理”。
   b. 在集群管理列表页面，选择需要访问的集群，获取并记录“内网访问地址”，一般是“<host>:<port>”或“<host>:<port>,<host>:<port>”样式。

      如果集群只有一个节点，此处仅显示1个节点的IP地址和端口号，例如“10.62.179.32:9200”；如果集群有多个节点，此处显示所有节点的IP地址和端口号，例如“10.62.179.32:9200,10.62.179.33:9200”。
2. 在ECS中执行如下命令，访问集群。集群的安全模式不同，访问命令也不同。
   - 非安全模式的集群
     ```
     curl "http://<host>:<port>"
     ```
   - 安全模式+HTTP协议的集群
     ```
     curl -u <user>:<password> "http://<host>:<port>"
     ```
   - 安全模式+HTTPS协议的集群
     ```
     curl -u <user>:<password> -k "https://<host>:<port>"
     ```

表 14-3 变量说明

变量名	说明
<host>	集群中各节点的IP地址，当集群包含多个节点时，会存在多个IP地址，可以任选其中一个发送。
<port>	集群节点的访问端口号，一般为9200。

变量名	说明
<user>	访问集群的用户名。
<password>	用户名对应的密码。

访问示例如下：

```
curl "http://10.62.176.32:9200"
```

返回结果如下：

```
HTTP/1.1 200 OK
content-type: application/json; charset=UTF-8
content-length: 513

{
  "name" : "xxx-1",
  "cluster_name" : "xxx",
  "cluster_uuid" : "xxx_uuid",
  "version" : {
    "number" : "7.10.2",
    "build_flavor" : "oss",
    "build_type" : "tar",
    "build_hash" : "unknown",
    "build_date" : "unknown",
    "build_snapshot" : true,
    "lucene_version" : "8.7.0",
    "minimum_wire_compatibility_version" : "6.7.0",
    "minimum_index_compatibility_version" : "6.0.0-beta1"
  },
  "tagline" : "You Know, for Search"
}
```

说明

更多命令，请参见**Elasticsearch官方文档**。

14.2.3 通过 Java 接入集群

14.2.3.1 通过 Rest High Level Client 接入集群

Elasticsearch官方提供了SDK（Rest High level Client）方式连接集群，Rest Client客户端对Elasticsearch的API进行了封装，用户只需要构造对应的结构即可对ES集群进行访问。Rest Client的详细使用请参考官方文档：**https://www.elastic.co/guide/en/elasticsearch/client/java-api-client/master/index.html**

本文介绍通过Rest High level Client访问CSS集群的配置说明。Rest High level Client接入集群有3种方式：

- **通过Rest High Level Client连接非安全集群**：适用于非安全模式的集群
- **通过Rest High Level Client连接安全集群（不使用安全证书）**：适用于安全模式+HTTP协议的集群、安全模式+HTTPS协议的集群（忽略证书）
- **通过Rest High Level Client连接安全集群（使用安全证书）**：适用于安全模式+HTTPS协议的集群

注意事项

建议Rest High Level Client的版本和Elasticsearch的版本保持一致，例如需要访问的ES集群版本是7.6.2，则使用的Rest High Level Client客户端版本建议也是7.6.2。若您

使用相比Elasticsearch集群更高版本的Java Rest High Level Client且存在少量请求的兼容性问题，您可以使用“RestHighLevelClient.getLowLevelClient()”方式直接获取Low Level Client，实现自定义的Elasticsearch请求内容。

准备工作

- CSS集群处于可用状态。
- 确保运行Java代码的服务器与CSS集群的网络是互通的。
- 确认服务器已安装JDK1.8，JDK1.8官网下载地址：**http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html**。
- 引入Java依赖。

  其中*7.6.2*为Elasticsearch Java客户端的版本号。

  - Maven方式引入：

    ```
    <dependency>
        <groupId>org.elasticsearch.client</groupId>
        <artifactId>elasticsearch-rest-high-level-client</artifactId>
        <version>7.6.2</version>
    </dependency>
    <dependency>
        <groupId>org.elasticsearch</groupId>
        <artifactId>elasticsearch</artifactId>
        <version>7.6.2</version>
    </dependency>
    ```

  - Gradle方式引入：

    ```
    compile group: 'org.elasticsearch.client', name: 'elasticsearch-rest-high-level-client', version: '7.6.2'
    ```

通过 Rest High Level Client 连接非安全集群

通过Rest High Level Client连接非安全集群，并查询*test*索引是否存在。代码示例如下：

```
import org.apache.http.HttpHost;
import org.elasticsearch.client.RequestOptions;
import org.elasticsearch.client.RestClient;
import org.elasticsearch.client.RestClientBuilder;
import org.elasticsearch.client.RestHighLevelClient;
import org.elasticsearch.client.indices.GetIndexRequest;

import java.io.IOException;
import java.util.Arrays;
import java.util.List;

/**
 * Rest Hive Level 连接非安全集群
 */
public class Main {
    public static void main(String[] args) throws IOException {
        List<String> host = Arrays.asList("x.x.x.x", "x.x.x.x");
        RestClientBuilder builder = RestClient.builder(constructHttpHosts(host, 9200, "http"));
        final RestHighLevelClient client = new RestHighLevelClient(builder);
        GetIndexRequest indexRequest = new GetIndexRequest("test");
        boolean exists = client.indices().exists(indexRequest, RequestOptions.DEFAULT);
        System.out.println(exists);
        client.close();
    }

    /**
     * constructHttpHosts函数转换host集群节点ip列表。
     */
    public static HttpHost[] constructHttpHosts(List<String> host, int port, String protocol) {
        return host.stream().map(p -> new HttpHost(p, port, protocol)).toArray(HttpHost[]::new);
```



```
    }
}
```

其中，*host*为集群各个节点的IP地址列表，当存在多个IP地址时，中间用“,”隔开；*test*为查询的索引名称。

通过 Rest High Level Client 连接安全集群（不使用安全证书）

该场景适用于连接2种集群：安全模式+HTTP协议的集群、安全模式+HTTPS协议的集群（忽略证书）。

代码示例如下：

```
import org.apache.http.HttpHost;
import org.apache.http.auth.AuthScope;
import org.apache.http.auth.UsernamePasswordCredentials;
import org.apache.http.client.CredentialsProvider;
import org.apache.http.impl.client.BasicCredentialsProvider;
import org.apache.http.impl.nio.client.HttpAsyncClientBuilder;
import org.apache.http.nio.conn.ssl.SSLIOSessionStrategy;
import org.apache.logging.log4j.LogManager;
import org.apache.logging.log4j.Logger;
import org.elasticsearch.action.admin.cluster.health.ClusterHealthRequest;
import org.elasticsearch.action.admin.cluster.health.ClusterHealthResponse;
import org.elasticsearch.client.RequestOptions;
import org.elasticsearch.client.RestClient;
import org.elasticsearch.client.RestClientBuilder;
import org.elasticsearch.client.RestHighLevelClient;
import org.elasticsearch.client.indices.GetIndexRequest;
import org.elasticsearch.common.Nullable;

import java.io.IOException;
import java.security.KeyManagementException;
import java.security.NoSuchAlgorithmException;
import java.security.SecureRandom;
import java.security.cert.CertificateException;
import java.security.cert.X509Certificate;
import java.util.Arrays;
import java.util.List;
import java.util.Objects;

import javax.net.ssl.HostnameVerifier;
import javax.net.ssl.SSLContext;
import javax.net.ssl.SSLSession;
import javax.net.ssl.TrustManager;
import javax.net.ssl.X509TrustManager;

/**
 * Rest High Level连接安全集群（不使用证书）
 */
public class Main {
    /**
     * 创建客户端的类，定义create函数用于创建客户端。
     */
    public static RestHighLevelClient create(List<String> host, int port, String protocol, int connectTimeout,
int connectionRequestTimeout, int socketTimeout, String username, String password) throws IOException{
        final CredentialsProvider credentialsProvider = new BasicCredentialsProvider();
        credentialsProvider.setCredentials(AuthScope.ANY, new UsernamePasswordCredentials(username,
password));
        SSLContext sc = null;
        try {
            sc = SSLContext.getInstance("SSL");
            sc.init(null, trustAllCerts, new SecureRandom());
        } catch (KeyManagementException | NoSuchAlgorithmException e) {
            e.printStackTrace();
        }
        SSLIOSessionStrategy sessionStrategy = new SSLIOSessionStrategy(sc, new NullHostNameVerifier());
        SecuredHttpClientConfigCallback httpClientConfigCallback = new
```



```
SecuredHttpClientConfigCallback(sessionStrategy,
            credentialsProvider);


        RestClientBuilder builder = RestClient.builder(constructHttpHosts(host, port, protocol))
            .setRequestConfigCallback(requestConfig -> requestConfig.setConnectTimeout(connectTimeout)
                .setConnectionRequestTimeout(connectionRequestTimeout)
                .setSocketTimeout(socketTimeout))
            .setHttpClientConfigCallback(httpClientConfigCallback);
        final RestHighLevelClient client = new RestHighLevelClient(builder);
        logger.info("es rest client build success {} ", client);

        ClusterHealthRequest request = new ClusterHealthRequest();
        ClusterHealthResponse response = client.cluster().health(request, RequestOptions.DEFAULT);
        logger.info("es rest client health response {} ", response);
        return client;
    }

    /**
     * constructHttpHosts函数转换host集群节点ip列表。
     */
    public static HttpHost[] constructHttpHosts(List<String> host, int port, String protocol) {
        return host.stream().map(p -> new HttpHost(p, port, protocol)).toArray(HttpHost[]::new);
    }

    /**
     * trustAllCerts忽略证书配置。
     */
    public static TrustManager[] trustAllCerts = new TrustManager[] {
        new X509TrustManager() {
            @Override
            public void checkClientTrusted(X509Certificate[] chain, String authType) throws
CertificateException {
            }

            @Override
            public void checkServerTrusted(X509Certificate[] chain, String authType) throws
CertificateException {
            }

            @Override
            public X509Certificate[] getAcceptedIssuers() {
                return null;
            }
        }
    };

    private static final Logger logger = LogManager.getLogger(Main.class);

    static class SecuredHttpClientConfigCallback implements RestClientBuilder.HttpClientConfigCallback {
        @Nullable
        private final CredentialsProvider credentialsProvider;
        /**
         * The {@link SSLIOSessionStrategy} for all requests to enable SSL / TLS encryption.
         */
        private final SSLIOSessionStrategy sslStrategy;
        /**
         * Create a new {@link SecuredHttpClientConfigCallback}.
         *
         * @param credentialsProvider The credential provider, if a username/password have been supplied
         * @param sslStrategy         The SSL strategy, if SSL / TLS have been supplied
         * @throws NullPointerException if {@code sslStrategy} is {@code null}
         */
        SecuredHttpClientConfigCallback(final SSLIOSessionStrategy sslStrategy,
            @Nullable final CredentialsProvider credentialsProvider) {
            this.sslStrategy = Objects.requireNonNull(sslStrategy);
            this.credentialsProvider = credentialsProvider;
        }
        /**
```



```
        * Get the {@link CredentialsProvider} that will be added to the HTTP client.
        *
        * @return Can be {@code null}.
        */
       @Nullable
       CredentialsProvider getCredentialsProvider() {
           return credentialsProvider;
       }
       /**
        * Get the {@link SSLIOSessionStrategy} that will be added to the HTTP client.
        *
        * @return Never {@code null}.
        */
       SSLIOSessionStrategy getSSLStrategy() {
           return sslStrategy;
       }
       /**
        * Sets the {@linkplain HttpAsyncClientBuilder#setDefaultCredentialsProvider(CredentialsProvider)
credential provider},
        *
        * @param httpClientBuilder The client to configure.
        * @return Always {@code httpClientBuilder}.
        */
       @Override
       public HttpAsyncClientBuilder customizeHttpClient(final HttpAsyncClientBuilder httpClientBuilder) {
           // enable SSL / TLS
           httpClientBuilder.setSSLStrategy(sslStrategy);
           // enable user authentication
           if (credentialsProvider != null) {
               httpClientBuilder.setDefaultCredentialsProvider(credentialsProvider);
           }
           return httpClientBuilder;
       }
   }

   public static class NullHostNameVerifier implements HostnameVerifier {
       @Override
       public boolean verify(String arg0, SSLSession arg1) {
           return true;
       }
   }

   /**
    * main函数参考如下，调用上面的create函数创建客户端，查询“test”索引是否存在。
    */
   public static void main(String[] args) throws IOException {
       RestHighLevelClient client = create(Arrays.asList("x.x.x.x", "x.x.x.x"), 9200, "https", 1000, 1000, 1000,
"username", "password");
       GetIndexRequest indexRequest = new GetIndexRequest("test");
       boolean exists = client.indices().exists(indexRequest, RequestOptions.DEFAULT);
       System.out.println(exists);
       client.close();
   }
}
```

表 14-4 函数中的变量说明

参数	描述
host	ES集群各个节点（或者独立Client节点）的IP地址列表，当存在多个IP地址时，中间用“,”隔开。
port	ES集群的连接端口，默认是“9200”。

参数	描述
protocol	连接协议，“http”或者“https”，根据集群实际情况填写。
connectTimeout	socket连接超时时间。
connectionRequestTimeout	socket连接请求超时时间。
socketTimeout	socket请求超时时间。
username	访问集群的用户名。
password	用户名对应的密码。

通过 Rest High Level Client 连接安全集群（使用安全证书）

该场景适用于使用安全证书连接安全模式+HTTPS协议的集群。

1. 获取安全证书（CloudSearchService.cer）。
   a. 登录云搜索服务控制台。
   b. 选择“集群管理”进入集群列表。
   c. 单击对应集群的名称，进入集群基本信息页面。
   d. 在“基本信息”页面，单击“HTTPS访问”后面的“下载证书”。
2. 转换安全证书（CloudSearchService.cer）。将下载的安全证书上传到客户端机器上，使用keytool工具将“.cer”证书转换成Java可以读取的“.jks”证书格式。
   - 在Linux系统中，执行如下命令转换证书。
     ```
     keytool -import -alias newname -keystore ./truststore.jks -file ./CloudSearchService.cer
     ```
   - 在Windows系统中，执行如下命令转换证书。
     ```
     keytool -import -alias newname -keystore .\truststore.jks -file .\CloudSearchService.cer
     ```

   其中，*newname*是由用户自定义的证书名称。

   该命令执行后，会提示设置证书密码，并确认密码。请保存该密码，后续接入集群会使用。
3. 接入集群。代码示例如下：

```
import org.apache.http.HttpHost;
import org.apache.http.auth.AuthScope;
import org.apache.http.auth.UsernamePasswordCredentials;
import org.apache.http.client.CredentialsProvider;
import org.apache.http.conn.ssl.NoopHostnameVerifier;
import org.apache.http.impl.client.BasicCredentialsProvider;
import org.apache.http.impl.nio.client.HttpAsyncClientBuilder;
import org.apache.http.nio.conn.ssl.SSLIOSessionStrategy;
import org.apache.logging.log4j.LogManager;
import org.apache.logging.log4j.Logger;
import org.elasticsearch.action.admin.cluster.health.ClusterHealthRequest;
import org.elasticsearch.action.admin.cluster.health.ClusterHealthResponse;
import org.elasticsearch.client.RequestOptions;
import org.elasticsearch.client.RestClient;
import org.elasticsearch.client.RestClientBuilder;
import org.elasticsearch.client.RestHighLevelClient;
import org.elasticsearch.client.indices.GetIndexRequest;
import org.elasticsearch.common.Nullable;

import java.io.File;
import java.io.FileInputStream;
```



```
import java.io.IOException;
import java.io.InputStream;
import java.security.KeyStore;
import java.security.SecureRandom;
import java.security.cert.CertificateException;
import java.security.cert.X509Certificate;
import java.util.Arrays;
import java.util.List;
import java.util.Objects;

import javax.net.ssl.SSLContext;
import javax.net.ssl.TrustManager;
import javax.net.ssl.TrustManagerFactory;
import javax.net.ssl.X509TrustManager;

/**
 * Rest Hive Level连接安全集群（使用https证书）
 */
public class Main {
    public static RestHighLevelClient create(List<String> host, int port, String protocol, int
connectTimeout, int connectionRequestTimeout, int socketTimeout, String username, String password,
String cerFilePath,
        String cerPassword) throws IOException {

        final CredentialsProvider credentialsProvider = new BasicCredentialsProvider();
        credentialsProvider.setCredentials(AuthScope.ANY, new
UsernamePasswordCredentials(username, password));
        SSLContext sc = null;
        try {
            TrustManager[] tm = {new MyX509TrustManager(cerFilePath, cerPassword)};
            sc = SSLContext.getInstance("SSL", "SunJSSE");
            //也可以使用SSLContext sslContext = SSLContext.getInstance("TLSv1.2");
            sc.init(null, tm, new SecureRandom());
        } catch (Exception e) {
            e.printStackTrace();
        }

        SSLIOSessionStrategy sessionStrategy = new SSLIOSessionStrategy(sc, new
NoopHostnameVerifier());
        SecuredHttpClientConfigCallback httpClientConfigCallback = new
SecuredHttpClientConfigCallback(sessionStrategy,
            credentialsProvider);

        RestClientBuilder builder = RestClient.builder(constructHttpHosts(host, port, protocol))
            .setRequestConfigCallback(requestConfig ->
requestConfig.setConnectTimeout(connectTimeout)
                .setConnectionRequestTimeout(connectionRequestTimeout)
                .setSocketTimeout(socketTimeout))
            .setHttpClientConfigCallback(httpClientConfigCallback);
        final RestHighLevelClient client = new RestHighLevelClient(builder);
        logger.info("es rest client build success {} ", client);

        ClusterHealthRequest request = new ClusterHealthRequest();
        ClusterHealthResponse response = client.cluster().health(request, RequestOptions.DEFAULT);
        logger.info("es rest client health response {} ", response);
        return client;
    }

    /**
     * constructHttpHosts函数转换host集群节点ip列表。
     */
    public static HttpHost[] constructHttpHosts(List<String> host, int port, String protocol) {
        return host.stream().map(p -> new HttpHost(p, port, protocol)).toArray(HttpHost[]::new);
    }

    /**
     * SecuredHttpClientConfigCallback类定义。
     */
    static class SecuredHttpClientConfigCallback implements
```



```
RestClientBuilder.HttpClientConfigCallback {
        @Nullable
        private final CredentialsProvider credentialsProvider;

        private final SSLIOSessionStrategy sslStrategy;

        SecuredHttpClientConfigCallback(final SSLIOSessionStrategy sslStrategy,
            @Nullable final CredentialsProvider credentialsProvider) {
            this.sslStrategy = Objects.requireNonNull(sslStrategy);
            this.credentialsProvider = credentialsProvider;
        }

        @Nullable
        CredentialsProvider getCredentialsProvider() {
            return credentialsProvider;
        }

        SSLIOSessionStrategy getSSLStrategy() {
            return sslStrategy;
        }

        @Override
        public HttpAsyncClientBuilder customizeHttpClient(final HttpAsyncClientBuilder
httpClientBuilder) {
            httpClientBuilder.setSSLStrategy(sslStrategy);
            if (credentialsProvider != null) {
                httpClientBuilder.setDefaultCredentialsProvider(credentialsProvider);
            }
            return httpClientBuilder;
        }
    }

    private static final Logger logger = LogManager.getLogger(Main.class);

    public static class MyX509TrustManager implements X509TrustManager {
        X509TrustManager sunJSSEX509TrustManager;

        MyX509TrustManager(String cerFilePath, String cerPassword) throws Exception {
            File file = new File(cerFilePath);
            if (!file.isFile()) {
                throw new Exception("Wrong Certification Path");
            }
            System.out.println("Loading KeyStore " + file + "...");
            InputStream in = new FileInputStream(file);
            KeyStore ks = KeyStore.getInstance("JKS");
            ks.load(in, cerPassword.toCharArray());
            TrustManagerFactory tmf = TrustManagerFactory.getInstance("SunX509", "SunJSSE");
            tmf.init(ks);
            TrustManager[] tms = tmf.getTrustManagers();
            for (TrustManager tm : tms) {
                if (tm instanceof X509TrustManager) {
                    sunJSSEX509TrustManager = (X509TrustManager) tm;
                    return;
                }
            }
            throw new Exception("Couldn't initialize");
        }

        @Override
        public void checkClientTrusted(X509Certificate[] chain, String authType) throws
CertificateException {

        }

        @Override
        public void checkServerTrusted(X509Certificate[] chain, String authType) throws
CertificateException {

        }
```



```
        @Override
        public X509Certificate[] getAcceptedIssuers() {
            return new X509Certificate[0];
        }
    }

    /**
     * main函数参考如下，调用上面的create函数创建客户端，查询“test”索引是否存在。
     */
    public static void main(String[] args) throws IOException {
        RestHighLevelClient client = create(Arrays.asList("xxx.xxx.xxx.xxx", "xxx.xxx.xxx.xxx"), 9200,
"https", 1000, 1000, 1000, "username", "password", "cerFilePath", "cerPassword");
        GetIndexRequest indexRequest = new GetIndexRequest("test");
        boolean exists = client.indices().exists(indexRequest, RequestOptions.DEFAULT);
        System.out.println(exists);
        client.close();
    }
}
```

表 14-5 函数中的参数说明

参数	描述
host	ES集群各个节点（或者独立Client节点）的IP地址列表，当存在多个IP地址时，中间用“,”隔开。
port	ES集群的连接端口，默认是“9200”。
protocol	连接协议，此处填写“https”。
connectTimeout	socket连接超时时间。
connectionRequestTimeout	socket连接请求超时时间。
socketTimeout	socket请求超时时间。
username	访问集群的用户名。
password	用户名对应的密码。
cerFilePath	证书路径。
cerPassword	证书密码。

14.2.3.2 通过 Rest Low Level Client 接入集群

High Level Client是在Low Level Client基础上进行封装的，如果High Level Client中的方法调用（例如.search，.bulk）不能满足使用需求，或存在兼容性问题，可以选择使用Low Level Client方式，甚至可以使用“HighLevelClient.getLowLevelClient()”方式直接获取Low Level Client。使用Low Level Client发送请求时需要自己定义请求结构，使用上更为灵活，能满足所有Elasticsearch支持的请求格式，例如GET、POST、DELETE、HEAD等。

本文介绍通过Rest Low Level Client访问CSS集群的配置说明。Rest Low Level Client接入集群有3种方式，每一种方式又分为直接创建Rest Client（即Rest Low Level Client）和通过创建High Level Client再调用getLowLevelClient()获取Low Level Client。

- **通过Rest Low Level Client连接非安全集群**：适用于非安全模式的集群
- **通过Rest Low Level Client连接安全集群（不使用安全证书）**：适用于安全模式+HTTP协议的集群、安全模式+HTTPS协议的集群（忽略证书）
- **通过Rest Low Level Client连接安全集群（使用安全证书）**：适用于安全模式+HTTPS协议的集群

注意事项

建议Rest Low Level Client的版本和Elasticsearch的版本保持一致，例如需要访问的ES集群版本是7.6.2，则使用的Rest Low Level Client客户端版本建议也是7.6.2。

准备工作

- CSS集群处于可用状态。
- 确保运行Java代码的服务器与CSS集群的网络是互通的。
- 确认服务器已安装JDK1.8，JDK1.8官网下载地址：**http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html**。
- 通过Maven方式引入apache版本。代码示例以7.6.2版本为例。

  其中*7.6.2*为Elasticsearch Java客户端的版本号。

```
<dependency>
    <groupId>org.elasticsearch.client</groupId>
    <artifactId>elasticsearch-rest-client</artifactId>
    <version>7.6.2</version>
</dependency>
<dependency>
    <groupId>org.elasticsearch</groupId>
    <artifactId>elasticsearch</artifactId>
    <version>7.6.2</version>
</dependency>
```

通过 Rest Low Level Client 连接非安全集群

- **方式一：直接创建Rest Low Level Client**

```
import org.apache.http.HttpHost;
import org.elasticsearch.client.Request;
import org.elasticsearch.client.Response;
import org.elasticsearch.client.RestClient;
import org.elasticsearch.client.RestClientBuilder;

import java.io.IOException;
import java.util.Arrays;
import java.util.List;

public class Main {

    public static void main(String[] args) throws IOException {
        List<String> host = Arrays.asList("xxx.xxx.xxx.xxx", "xxx.xxx.xxx.xxx");
        RestClientBuilder builder = RestClient.builder(constructHttpHosts(host, 9200, "http"));
        /**
         * 创建Rest Low Level Client。
         */
        RestClient lowLevelClient = builder.build();
        /**
         * 查询“test”索引是否存在。当索引存在时返回200，不存在时返回404。
         */
        Request request = new Request("HEAD", "/test");
        Response response = lowLevelClient.performRequest(request);
        System.out.println(response.getStatusLine().getStatusCode());
        lowLevelClient.close();
    }
```



```
    /**
     * constructHttpHosts函数转换host集群节点ip列表。
     */
    public static HttpHost[] constructHttpHosts(List<String> host, int port, String protocol) {
        return host.stream().map(p -> new HttpHost(p, port, protocol)).toArray(HttpHost[]::new);
    }
}
```

- **方式二：创建High Level Client再调用getLowLevelClient()获取Low Level Client**

```
import org.apache.http.HttpHost;
import org.elasticsearch.client.Request;
import org.elasticsearch.client.Response;
import org.elasticsearch.client.RestClient;
import org.elasticsearch.client.RestClientBuilder;
import org.elasticsearch.client.RestHighLevelClient;

import java.io.IOException;
import java.util.Arrays;
import java.util.List;

public class Main {

    public static void main(String[] args) throws IOException {
        List<String> host = Arrays.asList("xxx.xxx.xxx.xxx", "xxx.xxx.xxx.xxx");
        RestClientBuilder builder = RestClient.builder(constructHttpHosts(host, 9200, "http"));
        final RestHighLevelClient restHighLevelClient = new RestHighLevelClient(builder);
        /**
         * 创建High Level Client再调用getLowLevelClient()获取Low Level Client，即client创建仅下面这一行代码存在差别。
         */
        final RestClient lowLevelClient = restHighLevelClient.getLowLevelClient();
        /**
         * 查询“test”索引是否存在。当索引存在时返回200，不存在时返回404。
         */
        Request request = new Request("HEAD", "/test");
        Response response = lowLevelClient.performRequest(request);
        System.out.println(response.getStatusLine().getStatusCode());
        lowLevelClient.close();
    }

    /**
     * constructHttpHosts函数转换host集群节点ip列表。
     */
    public static HttpHost[] constructHttpHosts(List<String> host, int port, String protocol) {
        return host.stream().map(p -> new HttpHost(p, port, protocol)).toArray(HttpHost[]::new);
    }
}
```

其中，*host*为集群各个节点的IP地址列表，当存在多个IP地址时，中间用“,”隔开；*test*为查询的索引名称。

通过 Rest Low Level Client 连接安全集群（不使用安全证书）

- **方式一：直接创建Rest Low Level Client**

```
import org.apache.http.HttpHost;
import org.apache.http.HttpResponse;
import org.apache.http.auth.AuthScope;
import org.apache.http.auth.UsernamePasswordCredentials;
import org.apache.http.client.CredentialsProvider;
import org.apache.http.impl.client.BasicCredentialsProvider;
import org.apache.http.impl.client.DefaultConnectionKeepAliveStrategy;
import org.apache.http.impl.nio.client.HttpAsyncClientBuilder;
import org.apache.http.nio.conn.ssl.SSLIOSessionStrategy;
import org.apache.http.protocol.HttpContext;
import org.apache.logging.log4j.LogManager;
import org.apache.logging.log4j.Logger;
```



```
import org.elasticsearch.client.Request;
import org.elasticsearch.client.Response;
import org.elasticsearch.client.RestClient;
import org.elasticsearch.client.RestClientBuilder;
import org.elasticsearch.common.Nullable;

import java.io.IOException;
import java.security.KeyManagementException;
import java.security.NoSuchAlgorithmException;
import java.security.SecureRandom;
import java.security.cert.CertificateException;
import java.security.cert.X509Certificate;
import java.util.Arrays;
import java.util.List;
import java.util.Objects;
import java.util.concurrent.TimeUnit;

import javax.net.ssl.HostnameVerifier;
import javax.net.ssl.SSLContext;
import javax.net.ssl.SSLSession;
import javax.net.ssl.TrustManager;import javax.net.ssl.X509TrustManager;

public class Main {

    /**
     * 创建客户端的类，定义create函数用于创建客户端。
     */
    public static RestClient create(List<String> host, int port, String protocol, int connectTimeout, int
connectionRequestTimeout, int socketTimeout,  String username, String password) throws
IOException {
        final CredentialsProvider credentialsProvider = new BasicCredentialsProvider();
        credentialsProvider.setCredentials(AuthScope.ANY, new
UsernamePasswordCredentials(username, password));
        SSLContext sc = null;
        try {
            sc = SSLContext.getInstance("SSL");
            sc.init(null, trustAllCerts, new SecureRandom());
        } catch (KeyManagementException | NoSuchAlgorithmException e) {
            e.printStackTrace();
        }
        SSLIOSessionStrategy sessionStrategy = new SSLIOSessionStrategy(sc, new
NullHostNameVerifier());
        SecuredHttpClientConfigCallback httpClientConfigCallback = new
SecuredHttpClientConfigCallback(sessionStrategy,
            credentialsProvider);

        RestClientBuilder builder = RestClient.builder(constructHttpHosts(host, port, protocol))
            .setRequestConfigCallback(requestConfig ->
requestConfig.setConnectTimeout(connectTimeout)
                .setConnectionRequestTimeout(connectionRequestTimeout)
                .setSocketTimeout(socketTimeout))
            .setHttpClientConfigCallback(httpClientConfigCallback);
        final RestClient client = builder.build();
        logger.info("es rest client build success {} ", client);
        return client;
    }

    /**
     * constructHttpHosts函数转换host集群节点ip列表。
     */
    public static HttpHost[] constructHttpHosts(List<String> host, int port, String protocol) {
        return host.stream().map(p -> new HttpHost(p, port, protocol)).toArray(HttpHost[]::new);
    }

    /**
     * trustAllCerts忽略证书配置。
     */
    public static TrustManager[] trustAllCerts = new TrustManager[] {
        new X509TrustManager() {
```



```
            @Override
            public void checkClientTrusted(X509Certificate[] chain, String authType) throws
CertificateException {
            }

            @Override
            public void checkServerTrusted(X509Certificate[] chain, String authType) throws
CertificateException {
            }

            @Override
            public X509Certificate[] getAcceptedIssuers() {
                return null;
            }
        }
    };

    /**
     * CustomConnectionKeepAliveStrategy函数设置连接的保活时间，主要应对大量短连接的情况和数据
请求不高的场景。
     */
    public static class CustomConnectionKeepAliveStrategy extends
DefaultConnectionKeepAliveStrategy {
        public static final CustomConnectionKeepAliveStrategy INSTANCE = new
CustomConnectionKeepAliveStrategy();

        private CustomConnectionKeepAliveStrategy() {
            super();
        }

        /**
         * 最大keep alive的时间（分钟）
         * 这里默认为10分钟，可以根据实际情况设置。可以观察客户端机器状态为TIME_WAIT的TCP连接
数，如果太多，可以增大此值。
         */
        private final long MAX_KEEP_ALIVE_MINUTES = 10;

        @Override
        public long getKeepAliveDuration(HttpResponse response, HttpContext context) {
            long keepAliveDuration = super.getKeepAliveDuration(response, context);
            // <0 为无限期keepalive
            // 将无限期替换成一个默认的时间
            if (keepAliveDuration < 0) {
                return TimeUnit.MINUTES.toMillis(MAX_KEEP_ALIVE_MINUTES);
            }
            return keepAliveDuration;
        }
    }

    private static final Logger logger = LogManager.getLogger(Main.class);

    static class SecuredHttpClientConfigCallback implements
RestClientBuilder.HttpClientConfigCallback {
        @Nullable
        private final CredentialsProvider credentialsProvider;
        /**
         * The {@link SSLIOSessionStrategy} for all requests to enable SSL / TLS encryption.
         */
        private final SSLIOSessionStrategy sslStrategy;
        /**
         * Create a new {@link SecuredHttpClientConfigCallback}.
         *
         * @param credentialsProvider The credential provider, if a username/password have been
supplied
         * @param sslStrategy         The SSL strategy, if SSL / TLS have been supplied
         * @throws NullPointerException if {@code sslStrategy} is {@code null}
         */
        SecuredHttpClientConfigCallback(final SSLIOSessionStrategy sslStrategy,
            @Nullable final CredentialsProvider credentialsProvider) {
```



```
            this.sslStrategy = Objects.requireNonNull(sslStrategy);
            this.credentialsProvider = credentialsProvider;
        }
        /**
         * Get the {@link CredentialsProvider} that will be added to the HTTP client.
         *
         * @return Can be {@code null}.
         */
        @Nullable
        CredentialsProvider getCredentialsProvider() {
            return credentialsProvider;
        }
        /**
         * Get the {@link SSLIOSessionStrategy} that will be added to the HTTP client.
         *
         * @return Never {@code null}.
         */
        SSLIOSessionStrategy getSSLStrategy() {
            return sslStrategy;
        }
        /**
         * Sets the {@linkplain
HttpAsyncClientBuilder#setDefaultCredentialsProvider(CredentialsProvider) credential provider},
         *
         * @param httpClientBuilder The client to configure.
         * @return Always {@code httpClientBuilder}.
         */
        @Override
        public HttpAsyncClientBuilder customizeHttpClient(final HttpAsyncClientBuilder
httpClientBuilder) {
            // enable SSL / TLS
            httpClientBuilder.setSSLStrategy(sslStrategy);
            // enable user authentication
            if (credentialsProvider != null) {
                httpClientBuilder.setDefaultCredentialsProvider(credentialsProvider);
            }
            return httpClientBuilder;
        }
    }

    public static class NullHostNameVerifier implements HostnameVerifier {
        @Override
        public boolean verify(String arg0, SSLSession arg1) {
            return true;
        }
    }

    /**
     * main函数参考如下，调用create函数创建Rest Low Level Client客户端，查询“test”索引是否存在。
     */
    public static void main(String[] args) throws IOException {
        RestClient lowLevelClient = create(Arrays.asList("xxx.xxx.xxx.xxx", "xxx.xxx.xxx.xxx"), 9200, "http",
1000, 1000, 1000, "username", "password");
        Request request = new Request("HEAD", "/test");
        Response response = lowLevelClient.performRequest(request);
        System.out.println(response.getStatusLine().getStatusCode());
        lowLevelClient.close();
    }
}
```

- **方式二：创建High Level Client再调用getLowLevelClient()获取Low Level Client**

```
import org.apache.http.HttpHost;
import org.apache.http.HttpResponse;
import org.apache.http.auth.AuthScope;
import org.apache.http.auth.UsernamePasswordCredentials;
import org.apache.http.client.CredentialsProvider;
import org.apache.http.impl.client.BasicCredentialsProvider;
import org.apache.http.impl.client.DefaultConnectionKeepAliveStrategy;
import org.apache.http.impl.nio.client.HttpAsyncClientBuilder;
```



```
import org.apache.http.nio.conn.ssl.SSLIOSessionStrategy;
import org.apache.http.protocol.HttpContext;
import org.apache.logging.log4j.LogManager;
import org.apache.logging.log4j.Logger;
import org.elasticsearch.client.Request;
import org.elasticsearch.client.Response;
import org.elasticsearch.client.RestClient;
import org.elasticsearch.client.RestClientBuilder;
import org.elasticsearch.common.Nullable;

import java.io.IOException;
import java.security.KeyManagementException;
import java.security.NoSuchAlgorithmException;
import java.security.SecureRandom;
import java.security.cert.CertificateException;
import java.security.cert.X509Certificate;
import java.util.Arrays;
import java.util.List;
import java.util.Objects;
import java.util.concurrent.TimeUnit;

import javax.net.ssl.HostnameVerifier;
import javax.net.ssl.SSLContext;
import javax.net.ssl.SSLSession;
import javax.net.ssl.TrustManager;import javax.net.ssl.X509TrustManager;

import org.elasticsearch.client.RestHighLevelClient;

public class Main13 {

    /**
     * 创建客户端的类，定义create函数用于创建客户端。
     */
    public static RestHighLevelClient create(List<String> host, int port, String protocol, int
connectTimeout, int connectionRequestTimeout, int socketTimeout,  String username, String
password) throws IOException {

        final CredentialsProvider credentialsProvider = new BasicCredentialsProvider();
        credentialsProvider.setCredentials(AuthScope.ANY, new
UsernamePasswordCredentials(username, password));
        SSLContext sc = null;
        try {
            sc = SSLContext.getInstance("SSL");
            sc.init(null, trustAllCerts, new SecureRandom());
        } catch (KeyManagementException | NoSuchAlgorithmException e) {
            e.printStackTrace();
        }
        SSLIOSessionStrategy sessionStrategy = new SSLIOSessionStrategy(sc, new
NullHostNameVerifier());
        SecuredHttpClientConfigCallback httpClientConfigCallback = new
SecuredHttpClientConfigCallback(sessionStrategy,
            credentialsProvider);

        RestClientBuilder builder = RestClient.builder(constructHttpHosts(host, port, protocol))
            .setRequestConfigCallback(requestConfig ->
requestConfig.setConnectTimeout(connectTimeout)
                .setConnectionRequestTimeout(connectionRequestTimeout)
                .setSocketTimeout(socketTimeout))
            .setHttpClientConfigCallback(httpClientConfigCallback);
        final RestHighLevelClient client = new RestHighLevelClient(builder);
        logger.info("es rest client build success {} ", client);
        return client;
    }

    /**
     * constructHttpHosts函数转换host集群节点ip列表。
     */
    public static HttpHost[] constructHttpHosts(List<String> host, int port, String protocol) {
        return host.stream().map(p -> new HttpHost(p, port, protocol)).toArray(HttpHost[]::new);
```



```
    }

    /**
     * trustAllCerts忽略证书配置。
     */
    public static TrustManager[] trustAllCerts = new TrustManager[] {
        new X509TrustManager() {
            @Override
            public void checkClientTrusted(X509Certificate[] chain, String authType) throws
CertificateException {
            }

            @Override
            public void checkServerTrusted(X509Certificate[] chain, String authType) throws
CertificateException {
            }

            @Override
            public X509Certificate[] getAcceptedIssuers() {
                return null;
            }
        }
    };

    /**
     * CustomConnectionKeepAliveStrategy函数设置连接的保活时间，主要应对大量短连接的情况和数据
请求不高的场景。
     */
    public static class CustomConnectionKeepAliveStrategy extends
DefaultConnectionKeepAliveStrategy {
        public static final CustomConnectionKeepAliveStrategy INSTANCE = new
CustomConnectionKeepAliveStrategy();

        private CustomConnectionKeepAliveStrategy() {
            super();
        }

        /**
         * 最大keep alive的时间（分钟）
         * 这里默认为10分钟，可以根据实际情况设置。可以观察客户端机器状态为TIME_WAIT的TCP连接
数，如果太多，可以增大此值。
         */
        private final long MAX_KEEP_ALIVE_MINUTES = 10;

        @Override
        public long getKeepAliveDuration(HttpResponse response, HttpContext context) {
            long keepAliveDuration = super.getKeepAliveDuration(response, context);
            // <0 为无限期keepalive
            // 将无限期替换成一个默认的时间
            if (keepAliveDuration < 0) {
                return TimeUnit.MINUTES.toMillis(MAX_KEEP_ALIVE_MINUTES);
            }
            return keepAliveDuration;
        }
    }

    private static final Logger logger = LogManager.getLogger(Main.class);

    static class SecuredHttpClientConfigCallback implements
RestClientBuilder.HttpClientConfigCallback {
        @Nullable
        private final CredentialsProvider credentialsProvider;
        /**
         * The {@link SSLIOSessionStrategy} for all requests to enable SSL / TLS encryption.
         */
        private final SSLIOSessionStrategy sslStrategy;
        /**
         * Create a new {@link SecuredHttpClientConfigCallback}.
         *
```



```
        * @param credentialsProvider The credential provider, if a username/password have been
supplied
        * @param sslStrategy         The SSL strategy, if SSL / TLS have been supplied
        * @throws NullPointerException if {@code sslStrategy} is {@code null}
        */
       SecuredHttpClientConfigCallback(final SSLIOSessionStrategy sslStrategy,
           @Nullable final CredentialsProvider credentialsProvider) {
           this.sslStrategy = Objects.requireNonNull(sslStrategy);
           this.credentialsProvider = credentialsProvider;
       }
       /**
        * Get the {@link CredentialsProvider} that will be added to the HTTP client.
        *
        * @return Can be {@code null}.
        */
       @Nullable
       CredentialsProvider getCredentialsProvider() {
           return credentialsProvider;
       }
       /**
        * Get the {@link SSLIOSessionStrategy} that will be added to the HTTP client.
        *
        * @return Never {@code null}.
        */
       SSLIOSessionStrategy getSSLStrategy() {
           return sslStrategy;
       }
       /**
        * Sets the {@linkplain
HttpAsyncClientBuilder#setDefaultCredentialsProvider(CredentialsProvider) credential provider},
        *
        * @param httpClientBuilder The client to configure.
        * @return Always {@code httpClientBuilder}.
        */
       @Override
       public HttpAsyncClientBuilder customizeHttpClient(final HttpAsyncClientBuilder
httpClientBuilder) {
           // enable SSL / TLS
           httpClientBuilder.setSSLStrategy(sslStrategy);
           // enable user authentication
           if (credentialsProvider != null) {
               httpClientBuilder.setDefaultCredentialsProvider(credentialsProvider);
           }
           return httpClientBuilder;
       }
   }

   public static class NullHostNameVerifier implements HostnameVerifier {
       @Override
       public boolean verify(String arg0, SSLSession arg1) {
           return true;
       }
   }

   /**
    * main函数参考如下，调用create函数创建High Level Client再调用getLowLevelClient()获取Low
Level Client，并查询“test”索引是否存在。
    */
   public static void main(String[] args) throws IOException {
       RestHighLevelClient client = create(Arrays.asList(”xxx.xxx.xxx.xxx”, ”xxx.xxx.xxx.xxx”), 9200,
"http", 1000, 1000, 1000, "username", "password");
       RestClient lowLevelClient = client.getLowLevelClient();
       Request request = new Request("HEAD", "test");
       Response response = lowLevelClient.performRequest(request);
       System.out.println(response.getStatusLine().getStatusCode());
       lowLevelClient.close();
   }
}
```


表 14-6 函数中的变量说明

参数	描述
host	ES集群各个节点（或者独立Client节点）的IP地址列表，当存在多个IP地址时，中间用“,”隔开。
port	ES集群的连接端口，默认是“9200”。
protocol	连接协议，“http”或者“https”，根据集群实际情况填写。
connectTimeout	socket连接超时时间。
connectionRequestTimeout	socket连接请求超时时间。
socketTimeout	socket请求超时时间。
username	访问集群的用户名。
password	用户名对应的密码。

通过 Rest Low Level Client 连接安全集群（使用安全证书）

- **方式一：直接创建Rest Low Level Client**

```
import org.apache.http.HttpHost;
import org.apache.http.auth.AuthScope;
import org.apache.http.auth.UsernamePasswordCredentials;
import org.apache.http.client.CredentialsProvider;
import org.apache.http.conn.ssl.NoopHostnameVerifier;
import org.apache.http.impl.client.BasicCredentialsProvider;
import org.apache.http.impl.nio.client.HttpAsyncClientBuilder;
import org.apache.http.nio.conn.ssl.SSLIOSessionStrategy;
import org.apache.logging.log4j.LogManager;
import org.apache.logging.log4j.Logger;
import org.elasticsearch.client.Request;
import org.elasticsearch.client.Response;
import org.elasticsearch.client.RestClient;
import org.elasticsearch.client.RestClientBuilder;
import org.elasticsearch.common.Nullable;

import java.io.File;
import java.io.FileInputStream;
import java.io.IOException;
import java.io.InputStream;
import java.security.KeyStore;
import java.security.SecureRandom;
import java.security.cert.CertificateException;
import java.security.cert.X509Certificate;
import java.util.Arrays;
import java.util.List;
import java.util.Objects;

import javax.net.ssl.SSLContext;import javax.net.ssl.TrustManager;
import javax.net.ssl.TrustManagerFactory;
import javax.net.ssl.X509TrustManager;

public class Main13 {

    private static final Logger logger = LogManager.getLogger(Main.class);

    /**
     * 创建客户端的类，定义create函数用于创建客户端。
```



```
    */
    public static RestClient create(List<String> host, int port, String protocol, int connectTimeout, int
connectionRequestTimeout, int socketTimeout, String username, String password, String cerFilePath,
String cerPassword) throws IOException {
        final CredentialsProvider credentialsProvider = new BasicCredentialsProvider();
        credentialsProvider.setCredentials(AuthScope.ANY, new
UsernamePasswordCredentials(username, password));
        SSLContext sc = null;
        try {
            TrustManager[] tm = {new MyX509TrustManager(cerFilePath, cerPassword)};
            sc = SSLContext.getInstance("SSL", "SunJSSE");
            //也可以使用SSLContext sslContext = SSLContext.getInstance("TLSv1.2");
            sc.init(null, tm, new SecureRandom());
        } catch (Exception e) {
            e.printStackTrace();
        }

        SSLIOSessionStrategy sessionStrategy = new SSLIOSessionStrategy(sc, new
NoopHostnameVerifier());
        SecuredHttpClientConfigCallback httpClientConfigCallback = new
SecuredHttpClientConfigCallback(sessionStrategy,
            credentialsProvider);

        RestClientBuilder builder = RestClient.builder(constructHttpHosts(host, port, protocol))
            .setRequestConfigCallback(requestConfig ->
requestConfig.setConnectTimeout(connectTimeout)
                .setConnectionRequestTimeout(connectionRequestTimeout)
                .setSocketTimeout(socketTimeout))
            .setHttpClientConfigCallback(httpClientConfigCallback);
        final RestClient client = builder.build();
        logger.info("es rest client build success {} ", client);
        return client;
    }

    /**
     * constructHttpHosts函数转换host集群节点ip列表。
     */
    public static HttpHost[] constructHttpHosts(List<String> host, int port, String protocol) {
        return host.stream().map(p -> new HttpHost(p, port, protocol)).toArray(HttpHost[]::new);}

    static class SecuredHttpClientConfigCallback implements
RestClientBuilder.HttpClientConfigCallback {
        @Nullable
        private final CredentialsProvider credentialsProvider;

        private final SSLIOSessionStrategy sslStrategy;

        SecuredHttpClientConfigCallback(final SSLIOSessionStrategy sslStrategy,
            @Nullable final CredentialsProvider credentialsProvider) {
            this.sslStrategy = Objects.requireNonNull(sslStrategy);
            this.credentialsProvider = credentialsProvider;
        }

        @Nullable
        CredentialsProvider getCredentialsProvider() {
            return credentialsProvider;
        }

        SSLIOSessionStrategy getSSLStrategy() {
            return sslStrategy;
        }

        @Override
        public HttpAsyncClientBuilder customizeHttpClient(final HttpAsyncClientBuilder
httpClientBuilder) {
            httpClientBuilder.setSSLStrategy(sslStrategy);
            if (credentialsProvider != null) {
                httpClientBuilder.setDefaultCredentialsProvider(credentialsProvider);
            }
```



```
            return httpClientBuilder;
        }}

    public static class MyX509TrustManager implements X509TrustManager {
        X509TrustManager sunJSSEX509TrustManager;

        MyX509TrustManager(String cerFilePath, String cerPassword) throws Exception {
            File file = new File(cerFilePath);
            if (!file.isFile()) {
                throw new Exception("Wrong Certification Path");
            }
            System.out.println("Loading KeyStore " + file + "...");
            InputStream in = new FileInputStream(file);
            KeyStore ks = KeyStore.getInstance("JKS");
            ks.load(in, cerPassword.toCharArray());
            TrustManagerFactory tmf = TrustManagerFactory.getInstance("SunX509", "SunJSSE");
            tmf.init(ks);
            TrustManager[] tms = tmf.getTrustManagers();
            for (TrustManager tm : tms) {
                if (tm instanceof X509TrustManager) {
                    sunJSSEX509TrustManager = (X509TrustManager) tm;
                    return;
                }
            }
            throw new Exception("Couldn't initialize");
        }

        @Override
        public void checkClientTrusted(X509Certificate[] chain, String authType) throws
CertificateException {

        }

        @Override
        public void checkServerTrusted(X509Certificate[] chain, String authType) throws
CertificateException {

        }

        @Override
        public X509Certificate[] getAcceptedIssuers() {
            return new X509Certificate[0];
        }
    }

    /**
     * main函数参考如下，调用create函数创建Rest Low Level Client，查询“test”索引是否存在。
     */
    public static void main(String[] args) throws IOException {
        RestClient lowLevelClient = create(Arrays.asList("xxx.xxx.xxx.xxx", "xxx.xxx.xxx.xxx"), 9200,
"https", 1000, 1000, 1000, "username", "password", "cerFilePath", "cerPassword");
        Request request = new Request("HEAD", "test");
        Response response = lowLevelClient.performRequest(request);
        System.out.println(response.getStatusLine().getStatusCode());
        lowLevelClient.close();
    }
}
```

- **方式二：创建High Level Client再调用getLowLevelClient()获取Low Level Client**

```
import org.apache.http.HttpHost;
import org.apache.http.auth.AuthScope;
import org.apache.http.auth.UsernamePasswordCredentials;
import org.apache.http.client.CredentialsProvider;
import org.apache.http.conn.ssl.NoopHostnameVerifier;
import org.apache.http.impl.client.BasicCredentialsProvider;
import org.apache.http.impl.nio.client.HttpAsyncClientBuilder;
import org.apache.http.nio.conn.ssl.SSLIOSessionStrategy;
import org.apache.logging.log4j.LogManager;
import org.apache.logging.log4j.Logger;
```



```
import org.elasticsearch.action.admin.cluster.health.ClusterHealthRequest;
import org.elasticsearch.action.admin.cluster.health.ClusterHealthResponse;
import org.elasticsearch.client.Request;
import org.elasticsearch.client.RequestOptions;
import org.elasticsearch.client.Response;
import org.elasticsearch.client.RestClient;
import org.elasticsearch.client.RestClientBuilder;
import org.elasticsearch.client.RestHighLevelClient;
import org.elasticsearch.common.Nullable;

import java.io.File;
import java.io.FileInputStream;
import java.io.IOException;
import java.io.InputStream;
import java.security.KeyStore;
import java.security.SecureRandom;
import java.security.cert.CertificateException;
import java.security.cert.X509Certificate;
import java.util.Arrays;
import java.util.List;
import java.util.Objects;

import javax.net.ssl.SSLContext;import javax.net.ssl.TrustManager;
import javax.net.ssl.TrustManagerFactory;
import javax.net.ssl.X509TrustManager;

public class Main {

    private static final Logger logger = LogManager.getLogger(Main.class);

    /**
     * 创建客户端的类，定义create函数用于创建客户端。
     */
    public static RestHighLevelClient create(List<String> host, int port, String protocol, int
connectTimeout, int connectionRequestTimeout, int socketTimeout, String username, String password,
String cerFilePath, String cerPassword) throws IOException {
        final CredentialsProvider credentialsProvider = new BasicCredentialsProvider();
        credentialsProvider.setCredentials(AuthScope.ANY, new
UsernamePasswordCredentials(username, password));
        SSLContext sc = null;
        try {
            TrustManager[] tm = {new MyX509TrustManager(cerFilePath, cerPassword)};
            sc = SSLContext.getInstance("SSL", "SunJSSE");
            //也可以使用SSLContext sslContext = SSLContext.getInstance("TLSv1.2");
            sc.init(null, tm, new SecureRandom());
        } catch (Exception e) {
            e.printStackTrace();
        }

        SSLIOSessionStrategy sessionStrategy = new SSLIOSessionStrategy(sc, new
NoopHostnameVerifier());
        SecuredHttpClientConfigCallback httpClientConfigCallback = new
SecuredHttpClientConfigCallback(sessionStrategy,
            credentialsProvider);

        RestClientBuilder builder = RestClient.builder(constructHttpHosts(host, port, protocol))
            .setRequestConfigCallback(requestConfig ->
requestConfig.setConnectTimeout(connectTimeout)
                .setConnectionRequestTimeout(connectionRequestTimeout)
                .setSocketTimeout(socketTimeout))
            .setHttpClientConfigCallback(httpClientConfigCallback);
        final RestHighLevelClient client = new RestHighLevelClient(builder);
        logger.info("es rest client build success {} ", client);

        ClusterHealthRequest request = new ClusterHealthRequest();
        ClusterHealthResponse response = client.cluster().health(request, RequestOptions.DEFAULT);
        logger.info("es rest client health response {} ", response);
        return client;
    }
```



```
    /**
     * constructHttpHosts函数转换host集群节点ip列表。
     */
    public static HttpHost[] constructHttpHosts(List<String> host, int port, String protocol) {
        return host.stream().map(p -> new HttpHost(p, port, protocol)).toArray(HttpHost[]::new);}

    static class SecuredHttpClientConfigCallback implements
RestClientBuilder.HttpClientConfigCallback {
        @Nullable
        private final CredentialsProvider credentialsProvider;

        private final SSLIOSessionStrategy sslStrategy;

        SecuredHttpClientConfigCallback(final SSLIOSessionStrategy sslStrategy,
            @Nullable final CredentialsProvider credentialsProvider) {
            this.sslStrategy = Objects.requireNonNull(sslStrategy);
            this.credentialsProvider = credentialsProvider;
        }

        @Nullable
        CredentialsProvider getCredentialsProvider() {
            return credentialsProvider;
        }

        SSLIOSessionStrategy getSSLStrategy() {
            return sslStrategy;
        }

        @Override
        public HttpAsyncClientBuilder customizeHttpClient(final HttpAsyncClientBuilder
httpClientBuilder) {
            httpClientBuilder.setSSLStrategy(sslStrategy);
            if (credentialsProvider != null) {
                httpClientBuilder.setDefaultCredentialsProvider(credentialsProvider);
            }
            return httpClientBuilder;
        }}

    public static class MyX509TrustManager implements X509TrustManager {
        X509TrustManager sunJSSEX509TrustManager;

        MyX509TrustManager(String cerFilePath, String cerPassword) throws Exception {
            File file = new File(cerFilePath);
            if (!file.isFile()) {
                throw new Exception("Wrong Certification Path");
            }
            System.out.println("Loading KeyStore " + file + "...");
            InputStream in = new FileInputStream(file);
            KeyStore ks = KeyStore.getInstance("JKS");
            ks.load(in, cerPassword.toCharArray());
            TrustManagerFactory tmf = TrustManagerFactory.getInstance("SunX509", "SunJSSE");
            tmf.init(ks);
            TrustManager[] tms = tmf.getTrustManagers();
            for (TrustManager tm : tms) {
                if (tm instanceof X509TrustManager) {
                    sunJSSEX509TrustManager = (X509TrustManager) tm;
                    return;
                }
            }
            throw new Exception("Couldn't initialize");
        }

        @Override
        public void checkClientTrusted(X509Certificate[] chain, String authType) throws
CertificateException {

        }
```



```
        @Override
        public void checkServerTrusted(X509Certificate[] chain, String authType) throws
CertificateException {

        }

        @Override
        public X509Certificate[] getAcceptedIssuers() {
            return new X509Certificate[0];
        }
    }

    /**
     * main函数参考如下，调用create函数创建High Level Client再调用getLowLevelClient()获取Low
Level Client，并查询“test”索引是否存在。
     */
    public static void main(String[] args) throws IOException {
        RestHighLevelClient client = create(Arrays.asList("xxx.xxx.xxx.xxx", "xxx.xxx.xxx.xxx"), 9200,
"https", 1000, 1000, 1000, "username", "password", "cerFilePath", "cerPassword");
        RestClient lowLevelClient = client.getLowLevelClient();
        Request request = new Request("HEAD", "test");
        Response response = lowLevelClient.performRequest(request);
        System.out.println(response.getStatusLine().getStatusCode());
        lowLevelClient.close();
    }
}
```

表 14-7 函数中的参数说明

参数	描述
host	ES集群各个节点（或者独立Client节点）的IP地址列表，当存在多个IP地址时，中间用“,”隔开。
port	ES集群的连接端口，默认是“9200”。
protocol	连接协议，此处填写“https”。
connectTimeout	socket连接超时时间。
connectionRequestTimeout	socket连接请求超时时间。
socketTimeout	socket请求超时时间。
username	访问集群的用户名。
password	用户名对应的密码。
cerFilePath	证书路径。
cerPassword	证书密码。

14.2.3.3 通过 Transport Client 接入集群

本文介绍通过Transport Client访问CSS服务非安全集群的配置说明。若是安全模式的集群，建议通过Rest High Level Client访问Elasticsearch集群。

注意事项

建议Transport Client的版本和Elasticsearch的版本保持一致，例如需要访问的ES集群版本是7.6.2，则使用的Transport Client客户端版本建议也是7.6.2。

准备工作

- CSS集群处于可用状态。
- 确保运行Java代码的服务器与CSS集群的网络是互通的。
- 确认服务器已安装JDK1.8，JDK1.8官网下载地址：**http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html**。
- 引入Java依赖：

  其中*7.6.2*为Elasticsearch Java客户端的版本号。

```
<dependency>
    <groupId>org.elasticsearch.client</groupId>
    <artifactId>transport</artifactId>
    <version>7.6.2</version>
</dependency>
<dependency>
    <groupId>org.elasticsearch</groupId>
    <artifactId>elasticsearch</artifactId>
    <version>7.6.2</version>
</dependency>
```

操作步骤

以下介绍Transport Client连接Elasticsearch集群并查询*test*索引是否存在的代码示例。

```
import org.elasticsearch.action.ActionFuture;
import org.elasticsearch.action.admin.indices.exists.indices.IndicesExistsRequest;
import org.elasticsearch.action.admin.indices.exists.indices.IndicesExistsResponse;
import org.elasticsearch.client.transport.TransportClient;
import org.elasticsearch.common.settings.Settings;
import org.elasticsearch.common.transport.TransportAddress;
import org.elasticsearch.transport.client.PreBuiltTransportClient;

import java.net.InetAddress;
import java.net.UnknownHostException;
import java.util.concurrent.ExecutionException;

public class Main {
    public static void main(String[] args) throws ExecutionException, InterruptedException,
UnknownHostException {
        String cluster_name = "xxx";
        String host1 = "x.x.x.x";
        String host2 = "y.y.y.y";
        Settings settings = Settings.builder()
            .put("client.transport.sniff",false)
            .put("cluster.name", cluster_name)
            .build();
        TransportClient client = new PreBuiltTransportClient(settings)
            .addTransportAddress(new TransportAddress(InetAddress.getByName(host1), 9300))
            .addTransportAddress(new TransportAddress(InetAddress.getByName(host2), 9300));
        IndicesExistsRequest indicesExistsRequest = new IndicesExistsRequest("test");
        ActionFuture<IndicesExistsResponse> exists = client.admin().indices().exists(indicesExistsRequest);
        System.out.println(exists.get().isExists());
    }
}
```

其中，*cluster_name*为集群的名称；*host1*、*host2*为集群节点IP地址，，可通过**GET _cat/nodes**命令查看节点的IP地址。

14.2.4 通过 Python 接入集群

本文介绍通过Python语言访问CSS集群的配置说明。

准备工作

- CSS集群处于可用状态。
- 确保运行Python代码的服务器与CSS集群的网络是互通的。

操作步骤

1. 安装Elasticsearch Python客户端，建议和Elasticsearch的版本保持一致，例如需要访问的集群版本是7.6.2，则安装7.6的Elasticsearch Python客户端。
   ```
   pip install Elasticsearch==7.6.2
   ```
2. 创建Elasticsearch客户端并查看是否存在索引“test”。根据集群安全模式参考对应的示例代码。
   - 非安全模式的集群
     ```
     from elasticsearch import Elasticsearch

     class ElasticFactory(object):

         def __init__(self, host: list, port: str, username: str, password: str):
             self.port = port
             self.host = host
             self.username = username
             self.password = password

         def create(self) -> Elasticsearch:
             addrs = []
             for host in self.host:
                 addr = {'host': host, 'port': self.port}
                 addrs.append(addr)

             if self.username and self.password:
                 elasticsearch = Elasticsearch(addrs, http_auth=(self.username, self.password))
             else:
                 elasticsearch = Elasticsearch(addrs)
             return elasticsearch

     es = ElasticFactory(["xxx.xxx.xxx.xxx"], "9200", None, None).create()
     print(es.indices.exists(index='test'))
     ```
   - 安全模式+HTTP协议的集群
     ```
     from elasticsearch import Elasticsearch

     class ElasticFactory(object):

         def __init__(self, host: list, port: str, username: str, password: str):
             self.port = port
             self.host = host
             self.username = username
             self.password = password

         def create(self) -> Elasticsearch:
             addrs = []
             for host in self.host:
                 addr = {'host': host, 'port': self.port}
                 addrs.append(addr)

             if self.username and self.password:
                 elasticsearch = Elasticsearch(addrs, http_auth=(self.username, self.password))
             else:
                 elasticsearch = Elasticsearch(addrs)
             return elasticsearch
     ```



```
es = ElasticFactory(["xxx.xxx.xxx.xxx"], "9200", "username", "password").create()
print(es.indices.exists(index='test'))
```

- 安全模式+HTTPS协议的集群

```
from elasticsearch import Elasticsearch
import ssl

class ElasticFactory(object):

    def __init__(self, host: list, port: str, username: str, password: str):
        self.port = port
        self.host = host
        self.username = username
        self.password = password

    def create(self) -> Elasticsearch:
        context = ssl._create_unverified_context()

        addrs = []
        for host in self.host:
            addr = {'host': host, 'port': self.port}
            addrs.append(addr)

        if self.username and self.password:
            elasticsearch = Elasticsearch(addrs, http_auth=(self.username, self.password),
scheme="https", ssl_context=context)
        else:
            elasticsearch = Elasticsearch(addrs)
        return elasticsearch

es = ElasticFactory(["xxx.xxx.xxx.xxx"], "9200", "username", "password").create()
print(es.indices.exists(index='test'))
```

表 14-8 函数中的变量说明

参数	描述
host	ES集群各个节点（或者独立Client节点）的IP地址列表，当存在多个IP地址时，中间用“,”隔开。
port	ES集群的连接端口，填写“9200”。
username	访问集群的用户名。
password	用户名对应的密码。

3. 通过Elasticsearch客户端创建集群索引。

```
mappings = {
    "settings": {
        "index": {
            "number_of_shards": number_of_shards,
            "number_of_replicas": 1,
        },
    },
    "mappings": {
        properties
    }
}
result = es.indices.create(index=index, body=mappings)
```

4. 通过Elasticsearch客户端查询上一步创建的索引。

```
body = {
    "query": {
        "match": {
```



```
            "查询字段": "查询内容"
        }
    }
}
result = es.search(index=index, body=body)
```

14.2.5 通过 ES-Hadoop 实现 Hive 读写 Elasticsearch 数据

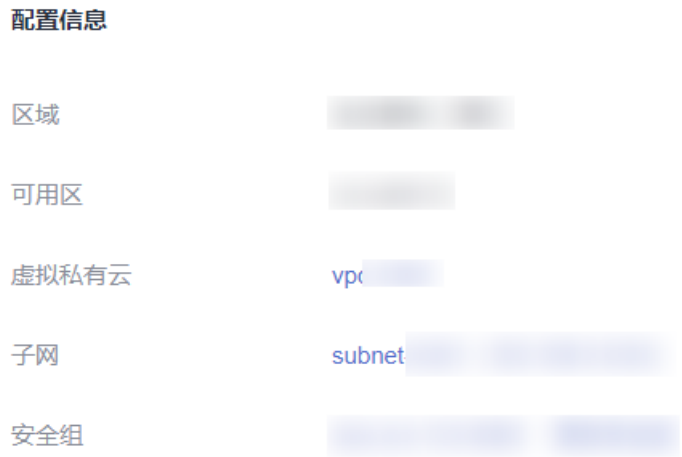
Elasticsearch-Hadoop (ES-Hadoop) 连接器将 Hadoop 海量的数据存储和深度加工能力与 Elasticsearch 实时搜索和分析功能结合在一起。它能够让您快速深入了解大数据，并让您在 Hadoop 生态系统中更好地开展工作。

本文通过MRS的ES-Hadoop与CSS集群连接作为示例，你可以配置其他任何需要使用ES集群的应用。如有需要，也可以参考本文在其他服务中使用Elasticsearch，前提是要保证客户端与Elasticsearch集群网络连通。

准备工作

- CSS集群处于可用状态。
- 确保客户端与CSS集群的网络是互通的。
- 确保CSS集群和MRS集群在同一个区域、可用区、虚拟私有云和子网。

图 14-4 CSS 集群的区域等信息



操作步骤

1. 获取集群的内网访问地址。访问集群时，需要输入内网访问地址。
   a. 在云搜索服务管理控制台，单击左侧导航栏的“集群管理”。
   b. 在集群管理列表页面，选择需要访问的集群，获取并记录“内网访问地址”，一般是“<host>:<port>”或“<host>:<port>,<host>:<port>”样式。
      如果集群只有一个节点，此处仅显示1个节点的IP地址和端口号，例如“10.62.179.32:9200”；如果集群有多个节点，此处显示所有节点的IP地址和端口号，例如“10.62.179.32:9200,10.62.179.33:9200”。
2. 登录MRS集群节点，操作步骤请参见。
3. 在MRS集群节点上通过curl命令检查网络连通性，需要确保MRS集群的每个节点都能连通CSS集群。
   - 非安全模式的集群
     ```
     curl -X GET http://<host>:<port>
     ```


- 安全模式+HTTP协议的集群
  ```
  curl -X GET http://<host>:<port> -u <user>:<password>
  ```
- 安全模式+HTTPS协议的集群
  ```
  curl -X GET https://<host>:<port> -u <user>:<password> -ik
  ```

表 14-9 变量说明

变量名	说明
<host>	集群中各节点的IP地址，当集群包含多个节点时，会存在多个IP地址，可以任选其中一个发送。
<port>	集群节点的访问端口号，一般为9200。
<user>	访问集群的用户名。
<password>	用户名对应的密码。

4. 下载**ES-Hadoop的lib包**，并解压zip包获取“elasticsearch-hadoop-x.x.x.jar”文件。版本需要与CSS集群版本一致，例如CSS集群是7.6.2版本，则建议下载elasticsearch-hadoop-7.6.2.zip。
5. 下载httpclient依赖包**commons-httpclient:commons-httpclient-3.1.jar**，其中*3.1*为版本号，建议用户根据实际需要选择。
6. 安装MRS客户端，如果已经安装可以跳过该步骤，未安装的请参见。
7. 登录MRS客户端，将下载的ES-Hadoop和httpclient的jar依赖包上传到MRS客户端。
8. 在MRS客户端创建HDFS目录，将ES-Hadoop lib包和httpclient依赖包上传到该目录下。
   ```
   hadoop fs -mkdir /tmp/hadoop-es
   hadoop fs -put elasticsearch-hadoop-x.x.x.jar /tmp/hadoop-es
   hadoop fs -put commons-httpclient-3.1.jar /tmp/hadoop-es
   ```
9. 从MRS客户端登录到Hive客户端，具体操作请参见。
10. 在Hive客户端，添加ES-Hadoop lib包和httpclient依赖包。该命令只对当前会话有效。

    输入**beeline**或**hive**进入到执行界面，执行如下命令：

    ```
    add jar hdfs:///tmp/hadoop-es/commons-httpclient-3.1.jar;
    add jar hdfs:///tmp/hadoop-es/elasticsearch-hadoop-x.x.x.jar;
    ```
11. 在Hive客户端，创建Hive外表。
    - 非安全模式的集群
      ```
      CREATE EXTERNAL table IF NOT EXISTS student(
        id BIGINT,
        name STRING,
        addr STRING
      )
      STORED BY 'org.elasticsearch.hadoop.hive.EsStorageHandler'
      TBLPROPERTIES(
         'es.nodes' = 'xxx.xxx.xxx.xxx:9200',
         'es.port' = '9200',
         'es.net.ssl' = 'false',
         'es.nodes.wan.only' = 'false',
         'es.nodes.discovery'='false',
         'es.input.use.sliced.partitions'='false',
         'es.resource' = 'student/_doc'
      );
      ```
    - 安全模式+HTTP协议的集群


```
CREATE EXTERNAL table IF NOT EXISTS student(
  id BIGINT,
  name STRING,
  addr STRING
)
STORED BY 'org.elasticsearch.hadoop.hive.EsStorageHandler'
TBLPROPERTIES(
   'es.nodes' = 'xxx.xxx.xxx.xxx:9200',
   'es.port' = '9200',
   'es.net.ssl' = 'false',
   'es.nodes.wan.only' = 'false',
   'es.nodes.discovery'='false',
   'es.input.use.sliced.partitions'='false',
   'es.nodes.client.only'='true',
   'es.resource' = 'student/_doc',
   'es.net.http.auth.user' = 'username',
   'es.net.http.auth.pass' = 'password'
);
```

- 安全模式+HTTPS协议的集群

  i. 获取安全证书“CloudSearchService.cer”。

     1) 登录云搜索服务控制台。
     2) 选择“集群管理”进入集群列表。
     3) 单击对应集群的名称，进入集群基本信息页面。
     4) 在“基本信息”页面，单击“HTTPS访问”后面的“下载证书”。

  ii. 转换安全证书（CloudSearchService.cer）。将下载的安全证书上传到客户端机器上，使用keytool工具将“.cer”证书转换成Java可以读取的“.jks”证书格式。

     - 在Linux系统中，执行如下命令转换证书。

       ```
       keytool -import -alias newname -keystore ./truststore.jks -file ./
       CloudSearchService.cer
       ```

     - 在Windows系统中，执行如下命令转换证书。

       ```
       keytool -import -alias newname -keystore .\truststore.jks -file .
       \CloudSearchService.cer
       ```

     其中，*newname*是由用户自定义的证书名称。

     该命令执行后，会提示设置证书密码，并确认密码。请保存该密码，后续接入集群会使用。

  iii. 将“.jks”文件分发到MRS集群的每个节点的相同路径，如“/tmp”，可以使用**scp**命令进行文件传输。同时，要确保**omm**用户有权限读取该文件，设置权限可以参考如下命令：

  ```
  chown -R omm truststore.jks
  ```

  iv. 创建Hive外表。

  ```
  CREATE EXTERNAL table IF NOT EXISTS student(
    id BIGINT,
    name STRING,
    addr STRING
  )
  STORED BY 'org.elasticsearch.hadoop.hive.EsStorageHandler'
  TBLPROPERTIES(
     'es.nodes' = 'https://xxx.xxx.xxx.xxx:9200',
     'es.port' = '9200',
     'es.net.ssl' = 'true',
     'es.net.ssl.truststore.location' = 'cerFilePath',
     'es.net.ssl.truststore.pass' = 'cerPassword',
     'es.nodes.wan.only' = 'false',
     'es.nodes.discovery'='false',
     'es.nodes.client.only'='true',
     'es.input.use.sliced.partitions'='false',
     'es.resource' = 'student/_doc',
  ```



```
  'es.net.http.auth.user' = 'username',
  'es.net.http.auth.pass' = 'password'
);
```

表 14-10 ES-Hadoop 参数说明

参数	默认值	描述
es.nodes	localhost	CSS集群的访问地址，可以集群列表页面查看“内网访问地址”。
es.port	9200	集群的访问端口号，一般为“9200”。
es.nodes.wan.only	false	是否进行节点嗅探。
es.nodes.discovery	true	是否禁用节点发现。
es.input.use.sliced.partitions	true	是否使用slice分区： ● true：使用。 ● false：不使用。 **说明** 设置为true，可能会导致索引在预读阶段的时间明显变长，有时会远远超出查询数据所耗费的时间。建议设置为false，以提高查询效率。
es.resource	NA	指定要读写的Index和type。
es.net.http.auth.user	NA	访问集群的用户名，只有启用了安全模式才需要配置此值。
es.net.http.auth.pass	NA	用户名所对应的密码，只有启用了安全模式才需要配置此值。
es.net.ssl	false	是否启用SSL，启用后需要配置安全证书信息。
es.net.ssl.truststore.location	NA	“.jks”证书文件的路径，如“file:///tmp/truststore.jks”。
es.nodes.client.only	false	是否配置了独立的Client节点的IP地址给es.nodes（即创建Elasticsearch集群时是否“启用Client节点”）。如果是，则需要将该值改为“true”，否则会报错找不到data节点。
es.net.ssl.truststore.pass	NA	“.jks”证书文件的密码。

更多ES-Hadoop配置项说明请参见**官方配置说明**。

12. 在Hive客户端，插入数据。
```
INSERT INTO TABLE student VALUES (1, "Lucy", "address1"), (2, "Lily", "address2");
```
13. 在Hive客户端，执行查询。
```
select * from student;
```

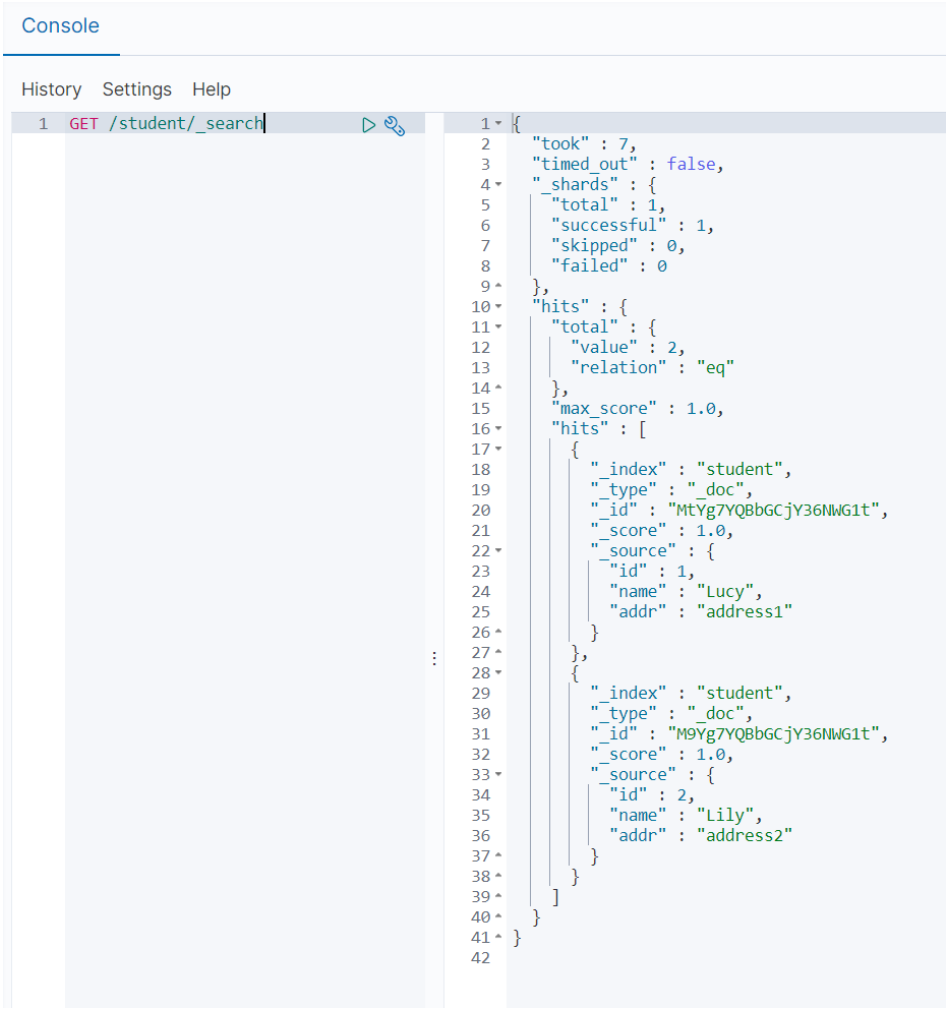

查询结果如下：

```
+-------------+---------------+---------------+
| student.id  | student.name  | student.addr  |
+-------------+---------------+---------------+
| 1           | Lucy          | address1      |
| 2           | Lily          | address2      |
+-------------+---------------+---------------+
2 rows selected (0.116 seconds)
```

14. 登录CSS控制台，在“集群管理”页面，单击集群操作列“Kibana”，登录Kibana界面。
15. 进入Kibana的“Dev Tools”页面执行查询命令，查看查询结果。

```
GET /student/_search
```

图 14-5 kibana 查询结果



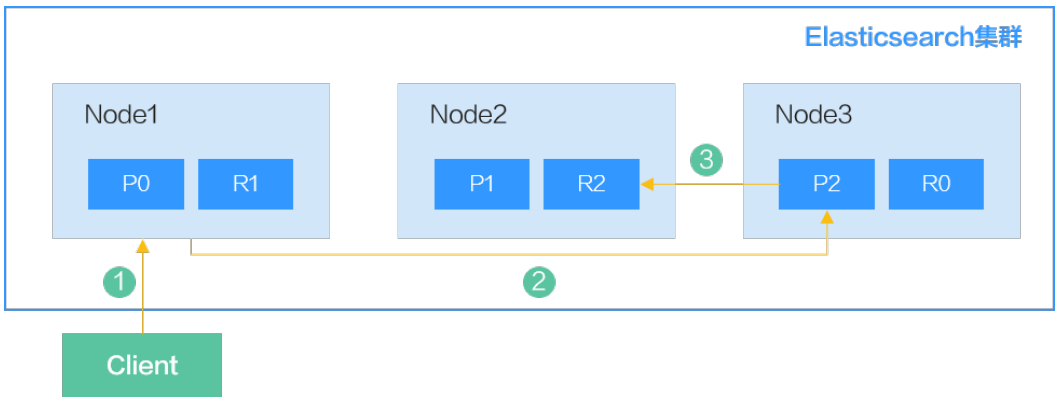
14.3 优化集群性能

14.3.1 写入性能优化

CSS集群在使用前，建议参考本文进行集群的写入性能优化，便于提高集群的写入性能，提升使用效率。

数据写入流程

图 14-6 数据写入流程

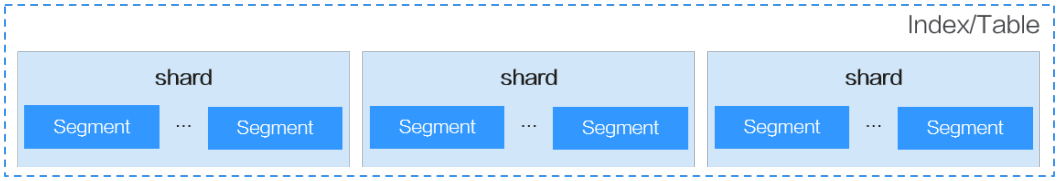


当从客户端往Elasticsearch中写入数据时，写入流程如下：

1. 客户端向Node1发送写数据请求，此时Node1为协调节点。
2. 节点Node1根据数据的_id将数据路由到分片2，此时请求会被转发到Node3，并执行写操作。
3. 当主分片写入成功后，它将请求转发到Node2的副本分片上。当副本写入成功后，Node3将向协调节点报告写入成功，协调节点向客户端报告写入成功。

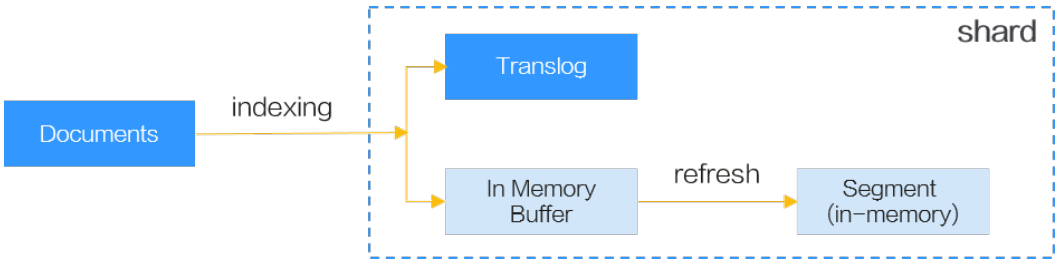
Elasticsearch中的单个索引由一个或多个分片(shard)组成，每个分片包含多个段（Segment），每一个Segment都是一个倒排索引。

图 14-7 Elasticsearch 的索引组成



将文档插入Elasticsearch时，文档首先会被写入缓冲区中，然后在刷新时定期从该缓冲区刷新到Segment中。刷新频率由refresh_interval参数控制，默认每1秒刷新一次。

图 14-8 文档插入 Elasticsearch 的流程



写入性能优化

基于Elasticsearch的数据写入流程分析，有以下几种性能优化方案。

表 14-11 写入性能优化

序号	优化方案	方案说明
1	使用SSD盘或升级集群配置	使用SSD盘可以大幅提升数据写入与merge操作的速度，对应到CSS服务，建议选择“超高IO型”存储，或者超高IO型主机。
2	采用Bulk API	客户端采用批量数据的写入方式，每次批量写入的数据建议在1~10MB之间。
3	随机生成_id	如果采用指定_id的写入方式，数据写入时会先触发一次查询操作，进而影响数据写入性能。对于不需要通过_id检索数据的场景，建议使用随机生成的_id。
4	设置合适的分片数	分片数建议设置为集群数据节点的倍数，且分片的大小控制在50GB以内。

5	关闭副本	数据写入与查询错峰执行，在数据写入时关闭数据副本，待数据写入完成后再开启副本。
		Elasticsearch 7.x版本中关闭副本的命令如下：
		PUT *{index}*/_settings
		{
		"number_of_replicas": 0
		}

6	调整索引的刷新频率	数据批量写入时，可以将索引的刷新频率“refresh_interval”设置为更大的值或者设置为“-1”（表示不刷新），通过减少分片刷新次数提高写入性能。
		Elasticsearch 7.x版本中，将更新时间设置为15s的命令如下：
		PUT *{index}*/_settings
		{
		"refresh_interval": "*15s*"
		}

7	优化写入线程数与写入队列大小	为应对突发流量，可以适当地提升写入线程数与写入队列的大小，防止突发流量导致出现错误状态码为429的情况。 Elasticsearch 7.x版本中，可以修改如下自定义参数实现写入优化：thread_pool.write.size，thread_pool.write.queue_size；

8	设置合适的字段类型	指定集群中各字段的类型，防止Elasticsearch默认将字段猜测为keyword和text的组合类型，增加不必要的数据量。其中keyword用于关键词搜索，text用于全文搜索。
		对于不需要索引的字段，建议“index”设置为“false”。
		Elasticsearch 7.x版本中，将字段“field1”设置为不建构索引的命令如下：
		PUT *{index}*
		{
		"mappings": {
		"properties": {
		"field1":{
		"type": "text",
		"index": false
		}
		}
		}
		}

序号	优化方案	方案说明

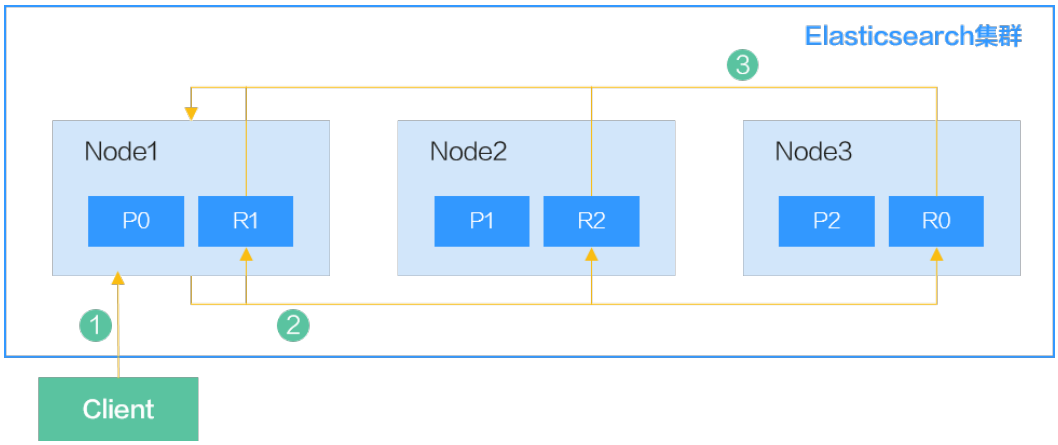
9	优化shard均衡策略	Elasticsearch默认采用基于磁盘容量大小的Load balance策略，多节点时，尤其是在新扩容的节点上，可能出现shard在各节点上分配不均的问题。为避免这类问题，可以通过设置索引级别的参数“routing.allocation.total_shards_per_node”控制索引分片在各节点的分布情况。此参数可以在索引模板中配置，也可以修改已有索引的setting生效。
		修改已有索引的setting的命令如下：
		PUT *{index}*/_settings
		{
		"index": {
		"routing.allocation.total_shards_per_node": 2
		}
		}

14.3.2 查询性能优化

CSS集群在使用前，建议参考本文进行集群的查询性能优化，便于提高集群的查询性能，提升使用效率。

数据查询流程

图 14-9 数据查询流程

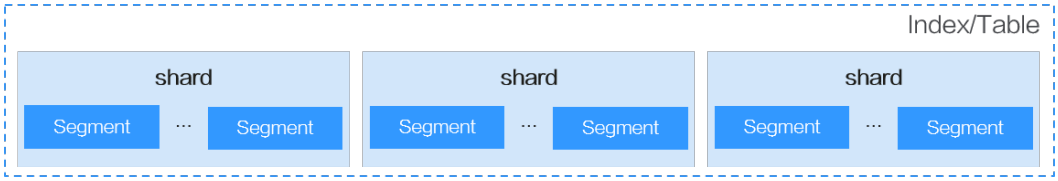


当从客户端往Elasticsearch发送查询请求时，查询流程如下：

1. 客户端向Node1发送查询请求，此时Node1为协调节点。
2. 节点Node1根据查询请求的索引以及其分片分布，进行分片选择；然后将请求转发到Node1、Node2、Node3。
3. 各分片分别执行查询任务；当各分片查询成功后，将查询结果汇聚到Node1，然后协调节点向客户端返回查询结果。

对于某个查询请求，其在节点上默认可并行查询5个分片，多于5个分片时将分批进行查询；在单个分片内，通过逐个遍历各个Segment的方式进行查询。

图 14-10 Elasticsearch 的索引组成



查询性能优化

基于Elasticsearch的数据查询流程分析，有以下几种性能优化方案。

表 14-12 查询性能优化

序号	优化方案	方案说明

1	通过_routing减少检索扫描的分片数	在数据入库时指定routing值，将数据路由到某个特定的分片，查询时通过该routing值将请求转发到某个特定的分片，而不是相关索引的所有分片，进而提升集群整体的吞吐能力。
		Elasticsearch 7.x版本中，设置命令如下：
		• 指定routing值插入数据
		`PUT /{index}/_doc/1?routing=user1`
		`{`
		`"title": "This is a document"`
		`}`
		• 根据routing值去查询数据
		`GET /{index}/_doc/1?routing=user1`

2	采用index sorting减少检索扫描的Segments数	当请求落到某个分片时，会逐个遍历其Segments，通过使用index sorting，可以使得范围查询、或者排序查询在段内提前终止(early-terminate)。
		Elasticsearch 7.x 版本中，示例命令如下：
		`//假设需要频繁使用字段date做范围查询。`
		`PUT {index}`
		`{`
		`"settings": {`
		`"index": {`
		`"sort.field": "date",`
		`"sort.order": "desc"`
		`}`
		`},`
		`"mappings": {`
		`"properties": {`
		`"date": {`
		`"type": "date"`
		`}`
		`}`
		`}`
		`}`

3	增加query cache提升缓存命中的概率	当filter请求在段内执行时，会通过bitset保留其刷选结果，当下一个类似的查询过来时，就可以复用之前查询的结果，以此减少重复查询。 增加query cache可以通过修改集群的参数配置实现，将自定义缓存参数“indices.queries.cache.size”设置为更大的值。具体操作请参见，修改参数配置后一定要重启集群使参数生效。

序号	优化方案	方案说明
4	提前Forcemerge，减小需要扫描的Segments数	对于定期滚动后的只读索引，可以定期执行forcemerge，将小的Segments合并为大的Segments，并将标记为“deleted”状态的索引彻底删除，提升查询效率。 Elasticsearch 7.x 版本中，配置示例如下： //假设配置索引forcemerge后segments数量为10个。 POST /*{index}*/_forcemerge?max_num_segments=10

14.4 实践案例

14.4.1 使用 CSS 加速数据库的查询分析

方案架构

关系型数据库（例如MySQL、GaussDB for MySQL等）受限于全文检索和Ad Hoc查询能力，因此会将Elasticsearch作为关系型数据库的补充，以此提升数据库的全文检索能力和高并发的Ad Hoc查询能力。

本章主要介绍，如何将MySQL数据库中的数据同步到云搜索服务CSS，通过CSS实现数据库的全文检索与Ad Hoc查询分析加速。方案架构图如图14-11所示。

图 14-11 CSS 加速数据库的查询分析方案

1. 用户业务数据存储到MySQL。
2. 通过数据复制服务DRS将MySQL中的数据实时同步到CSS。
3. 通过CSS进行全文检索与数据查询分析。

前提条件

- 已创建好安全模式的CSS集群和MySQL数据库，且两者在同一个VPC与安全组内。
- MySQL数据库中已经有待同步的数据。本章以如下表结构和初始数据举例。
  a. MySQL中创建一个学生信息表：
  ```
  CREATE TABLE `student` (
    `dsc` varchar(100) COLLATE utf8mb4_general_ci DEFAULT NULL,
    `age` smallint unsigned DEFAULT NULL,
    `name` varchar(32) COLLATE utf8mb4_general_ci NOT NULL,
    `id` int unsigned NOT NULL,
    PRIMARY KEY (`id`)
  ) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;
  ```
  b. MySQL中插入3个学生的初始数据：
  ```
  INSERT INTO student (id,name,age,dsc)
  VALUES
  ('1','Jack Ma Yun','50','Jack Ma Yun is a business magnate, investor and philanthropist.'),
  ('2','will smith','22','also known by his stage name the Fresh Prince, is an American actor, rapper, and producer.'),
  ('3','James Francis Cameron','68','the director of avatar');
  ```

- CSS集群中已完成索引创建，与MySQL中表相对应。

  本章集群的索引示例如下：

```
PUT student
{
  "settings": {
    "number_of_replicas": 0,
    "number_of_shards": 3
    },
  "mappings": {
    "properties": {
      "id": {
        "type": "keyword"
        },
      "name": {
        "type": "short"
        },
      "age": {
        "type": "short"
        },
      "desc": {
        "type": "text"
        }
    }
  }
}
```

  其中的number_of_shards与number_of_replicas需根据具体业务场景进行配置。

操作步骤

步骤1 通过DRS将MySQL数据实时同步到CSS。具体操作步骤请参见。

在本章案例中，**表14-13**中的同步任务配置参数需要按建议填写。

表 14-13 同步任务参数说明

配置模块	参数名称	填写建议
同步实例 > 同步实例信息	“网络类型”	选择“VPC网络”。
	“源数据库实例”	选择需要同步的RDS for MySQL实例，即存储用户业务数据的MySQL。
	“同步实例所在子网”	选择同步实例所在的子网，建议跟数据库实例以及CSS集群所在子网保持一致。
源库及目标库 > 目标库信息	“VPC”和“子网”	选择和CSS集群一致的VPC与子网。
	“IP地址或域名”	填写CSS集群的IP地址，获取方式请参见**获取CSS集群的IP地址**。
	“数据库用户名”和“数据库密码”	填写CSS集群的管理员帐户名（admin）和密码。
	“加密证书”	选择CSS集群的安全证书，如果未启用“SSL安全链接”，则不用选择。获取方式请参见**获取CSS集群的安全证书**。
设置同步	“流速模式”	选择“不限速”。

配置模块	参数名称	填写建议
	“同步对象类型”	不勾选“同步表结构”，因为已经预先在CSS集群中创建了与MySQL中表相对应的索引。
	“同步对象”	选择“表级同步”，选择与CSS对应的数据库以及表名。 **说明** 配置项中type名称需要与索引名称一样，都是“_doc”，若不一致请修改。
数据加工	-	直接“下一步”。

启动同步任务后，等待任务“状态”从“全量同步”变成“增量同步”，表示数据进入实时同步状态。

步骤2 验证数据库的同步状态。

1. 全量数据同步验证。

   在CSS的Kibana中执行如下命令，确认全量数据是否同步到CSS。

   ```
   GET student/_search
   ```

2. 源端插入新数据，验证数据是否会同步到CSS。

   例如，端源插入“id”为“4”的新数据。

   ```
   INSERT INTO student (id,name,age,dsc)
   VALUES
   ('4','Bill Gates','50','Gates III is an American business magnate, software developer, investor, author, and philanthropist.')
   ```

   在CSS的Kibana中执行如下命令，确认新数据是否同步到CSS。

   ```
   GET student/_search
   ```

3. 源端更新数据，验证数据是否会同步更新到CSS。

   例如，更新“id”为“4”这条数据的“age”字段，从“50”改成“55”。

   ```
   UPDATE student set age='55' WHERE id=4;
   ```

   在CSS的Kibana中执行如下命令，确认数据是否同步更新到CSS。

   ```
   GET student/_search
   ```

4. 源端删除数据，验证CSS里的数据是否同步删除。

   例如，删除“id”为“4”的数据。

   ```
   DELETE FROM student WHERE id=4;
   ```

   在CSS的Kibana中执行如下如下命令，确认CSS里的数据是否被同步删除。

   ```
   GET student/_search
   ```

步骤3 验证数据库的全文检索能力。

例如，在CSS查询“dsc”中包含“avatar”的数据。

```
GET student/_search
{
  "query": {
    "match": {
      "dsc": "avatar"
    }
  }
}
```


步骤4 验证数据库的Ad Hoc查询能力。

例如，在CSS查询年龄大于40的philanthropist。

```
GET student/_search
{
  "query": {
    "bool": {
      "must": [
        {
          "match": {
            "dsc": "philanthropist"
          }
        },
        {
          "range": {
            "age": {
              "gte": 40
            }
          }
        }
      ]
    }
  }
}
```

步骤5 验证数据库的统计分析能力。

例如，在CSS统计所有人的年龄分布。

```
GET student/_search
{
  "size": 0,
  "query": {
    "match_all": {}
  },
  "aggs": {
    "age_count": {
      "terms": {
        "field": "age",
        "size": 10
      }
    }
  }
}
```

----结束

其他操作

- **获取CSS集群的IP地址**
  a. 在云搜索服务管理控制台，单击左侧导航栏的“集群管理”。
  b. 在集群管理列表页面，选择需要访问的集群，在“内网访问地址”列获取CSS集群的IP地址，一般是“<host>:<port>”或“<host>:<port>,<host>:<port>”样式。

     如果集群只有一个节点，此处仅显示1个节点的IP地址和端口号，例如“10.62.179.32:9200”；如果集群有多个节点，此处显示所有节点的IP地址和端口号，例如“10.62.179.32:9200,10.62.179.33:9200”。
- **获取CSS集群的安全证书**
  a. 登录云搜索服务控制台。
  b. 选择“集群管理”进入集群列表。

c. 单击对应集群的名称，进入集群基本信息页面。
d. 在“基本信息”页面，单击“HTTPS访问”后面的“下载证书”。

14.4.2 使用 CSS 搭建统一日志管理平台

使用CSS搭建的统一日志管理平台可以实时地、统一地、方便地管理日志，让日志驱动运维、运营等，提升服务管理效率。

方案架构

ELKB（Elasticsearch、Logstash、Kibana、Beats）提供了一整套日志场景解决方案，是目前主流的一种日志系统。其框架如图所示。

图 14-12 统一日志管理平台框架

- **Beats**是一个轻量级日志采集器，包括Filebeat、Metricbeat等。
- **Logstash**用于对日志进行搜集与预处理，支持多种数据源与ETL处理方式。
- **Elasticsearch**是个开源分布式搜索引擎，提供搜集、分析、存储数据等主要功能，CSS云搜索服务可以创建Elasticsearch集群。
- **Kibana**是一个可视化工具，可以基于Kibana进行Web可视化查询，并制作BI报表。

本章以CSS、Filebeat、Logstash和Kibana为例，搭建一个统一日志管理平台。使用Filebeat采集ECS中的日志，发送到Logstash进行数据处理，再存储到CSS中，最后通过Kibana进行日志的可视化查询与分析。

ELKB系统中各组件的版本兼容性请参见 **https://www.elastic.co/support/matrix#matrix_compatibility**。

前提条件

- 已创建好非安全模式的CSS集群。
- 已申请了ECS虚拟机，并安装了Java环境。

操作步骤

步骤1 部署并配置Filebeat。

1. 下载Filebeat，版本建议选择7.6.2。下载地址：**https://www.elastic.co/downloads/past-releases#filebeat-oss**
2. 配置Filebeat的配置文件“filebeat.yml”。
   例如，采集“/root/”目录下以“log”结尾的所有文件，配置文件“filebeat.yml”中的内容如下：

```
filebeat.inputs:
- type: log
  enabled: true
  # 采集的日志文件路径。
  paths:
    - /root/*.log
```

```
filebeat.config.modules:
  path: ${path.config}/modules.d/*.yml
  reload.enabled: false
# Logstash的hosts信息
output.logstash:
  hosts: ["192.168.0.126:5044"]

processors:
```

步骤2 部署并配置Logstash。

说明

为了获得更优的性能，Logstash中的JVM参数建议配置为ECS/docker中内存的一半。

1. 下载Logstash，版本建议选择7.6.2。下载地址：**https://www.elastic.co/downloads/past-releases#logstash-oss**
2. 确保Logstash与CSS集群的网络互通。
3. 配置Logstash的配置文件“logstash-sample.conf”。中的内容如下：

   配置文件“logstash-sample.conf”中的内容如下：

```
input {
  beats {
    port => 5044
  }
}
# 对数据的切割与截取信息，
filter {
    grok {
        match => {
            "message" => '\[%{GREEDYDATA:timemaybe}\] \[%{WORD:level}\] %{GREEDYDATA:content}'
        }
    }
    mutate {
      remove_field => ["@version","tags","source","input","prospector","beat"]
    }
}
# CSS集群的信息
output {
  elasticsearch {
    hosts => ["http://192.168.0.4:9200"]
    index => "%{[@metadata][beat]}-%{+YYYY.MM.dd}"
    #user => "xxx"
    #password => "xxx"
  }
}
```

说明

Logstash的“filter”进行模式配置时，可以借助Grok Debugger（**http://grokdebug.herokuapp.com/**）。

步骤3 在Kibana上或通过API配置CSS集群的索引模板。

例如，创建一个索引模板，配置索引默认采用3分片、0副本，索引中定义了@timestamp、content、host.name、level、log.file.path、message、timemaybe等字段。

```
PUT _template/filebeat
{
  "index_patterns": ["filebeat*"],
  "settings": {
    # 定义分片数。
    "number_of_shards": 3,
    # 定义副本数。
    "number_of_replicas": 0,
    "refresh_interval": "5s"
```



```
},
# 定义字段。
"mappings": {
      "properties": {
        "@timestamp": {
          "type": "date"
        },
        "content": {
          "type": "text"
        },
        "host": {
          "properties": {
            "name": {
              "type": "text"
            }
          }
        },
        "level": {
          "type": "keyword"
        },
        "log": {
          "properties": {
            "file": {
              "properties": {
                "path": {
                  "type": "text"
                }
              }
            }
          }
        },
        "message": {
          "type": "text"
        },
        "timemaybe": {
          "type": "date",
          "format": "yyyy-MM-dd HH:mm:ss||epoch_millis"
        }
      }
   }
}
```

步骤4 在ECS上准备测试数据。

执行如下命令，生成测试数据，并将数据写到“/root/tmp.log”中：

```
bash -c 'while true; do echo [$(date)] [info] this is the test message; sleep 1; done;' >> /root/tmp.log &
```

生成的测试数据样例如下：

```
[Thu Feb 13 14:01:16 CST 2020] [info] this is the test message
```

步骤5 执行如下命令，启动Logstash。

```
nohup ./bin/logstash -f /opt/pht/logstash-6.8.6/logstash-sample.conf &
```

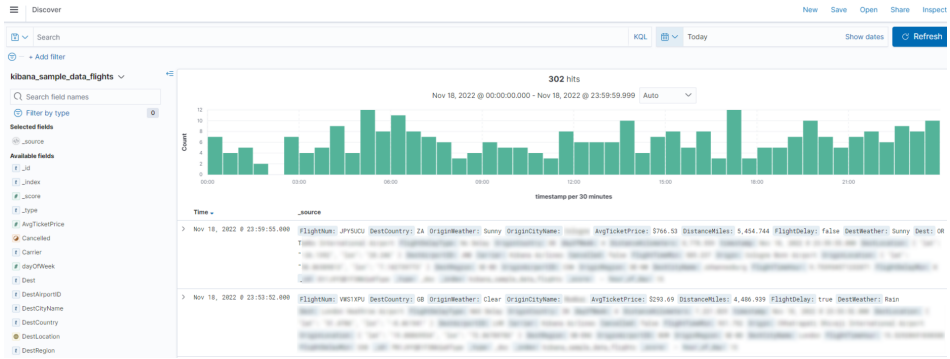
步骤6 执行如下命令，启动Filebeat。

```
./filebeat
```

步骤7 通过Kibana进行查询并制作报表。

1. 进入CSS集群的Kibana页面。
2. 选择“Discover”进行查询与分析，类似的效果如下：

图 14-13 Discover 界面示例



----结束

14.4.3 使用 Elasticsearch 集群自定义评分查询

通过Elasticsearch集群可以对匹配的文档进行评分。本文介绍如何进行自定义评分查询。

方案概述

自定义评分查询有两种方式。

- 用**绝对好评率**计算总分，按照总分由高到低的顺序排列出查询结果。
  总分 = 匹配得分 *（好评率 * 绝对因子）
  - 匹配得分：根据查询结果计分，内容匹配记1分，否则记0分，得分之和即为匹配得分。
  - 好评率：从匹配项的数据内容中获取好评率的值，一般指单条数据的评分。
  - 绝对因子：自定义的好评比例。
- 用**相对好评率**计算总分，按照总分由高到低的顺序排列查询结果。
  总分 = 匹配得分 *（好评率 * 相对分数）
  - 匹配得分：根据查询结果计分，内容匹配记1分，否则记0分，得分之和即为匹配得分。
  - 好评率：从匹配项的数据内容中获取好评率的值，一般指单条数据的评分。
  - 相对分数：自定义一个好评率阈值，当好评率大于阈值时，返回一个自定义的相对分数；当好评率小于等于阈值时，返回另一个自定义的相对分数。通过这种方式可以避免异常好评率对查询结果的影响。

前提条件

已经在云搜索服务管理控制台创建好Elasticsearch集群，且集群处于可用状态。

操作步骤

说明

本文的代码示例仅适用于Elasticsearch 7.x及以上版本的集群。

1. 登录云搜索服务管理控制台。
2. 在左侧导航栏，选择“集群管理”，进入Elasticsearch集群列表页面。

3. 在集群列表页面中，单击集群操作列的“Kibana”登录Kibana页面。
4. 在Kibana的左侧导航中选择“Dev Tools”，进入命令执行页面。
5. 创建索引，并指定自定义映射来定义数据类型。

   例如，数据文件“tv.json”的内容如下所示。

   ```
   {
   "tv":[
   { "name": "tv1", "description": "USB, DisplayPort", "vote": 0.98 }
   { "name": "tv2", "description": "USB, HDMI", "vote": 0.99 }
   { "name": "tv3", "description": "USB", "vote": 0.5 }
   { "name": "tv4", "description": "USB, HDMI, DisplayPort", "vote": 0.7 }
   ]
   }
   ```

   可以执行如下命令，创建索引“mall”，并指定自定义映射来定义数据类型。

   ```
   PUT /mall?pretty
   {
     "mappings": {
       "properties": {
         "name": {
           "type": "text",
           "fields": {
             "keyword": {
               "type": "keyword"
             }
           }
         },
         "description": {
           "type": "text",
           "fields": {
             "keyword": {
               "type": "keyword"
             }
           }
         },
         "vote": {
           "type": "float"
         }
       }
     }
   }
   ```

6. 导入数据。

   执行如下命令，将“tv.json”文件中的数据导入到“mall”索引中。

   ```
   POST /mall/_bulk?pretty
   { "index": {"_id": "1"}}
   { "name": "tv1", "description": "USB, DisplayPort", "vote": 0.98 }
   { "index": {"_id": "2"}}
   { "name": "tv2", "description": "USB, HDMI", "vote": 0.99 }
   { "index": {"_id": "3"}}
   { "name": "tv3", "description": "USB", "vote": 0.5 }
   { "index": {"_id": "4"}}
   { "name": "tv4", "description": "USB, HDMI, DisplayPort", "vote": 0.7 }
   ```

7. 自定义评分查询数据。分别列举了绝对好评率和相对好评率查询方式。

   假设用户想要查询有USB接口、HDMI接口、DisplayPort接口的电视机，并根据好评率计算各款电视机的总分，根据总分由高到低的顺序排列结果。

   - 用绝对好评率计算总分

     总分的计算公式为“new_score = query_score * (vote * factor)”，执行的命令如下：

     ```
     GET /mall/_doc/_search?pretty
     {
       "query":{
         "function_score":{
           "query":{
     ```



```
      "bool":{
        "should":[
          {"match": {"description": "USB"}},
          {"match": {"description": "HDMI"}},
          {"match": {"description": "DisplayPort"}}
        ]
      }
    },
    "field_value_factor":{
      "field":"vote",
      "factor":1
    },
    "boost_mode":"multiply",
    "max_boost":10
  }
 }
}
```

返回结果如下所示，按照总分由高到低的顺序排列查询结果。

```
{
 "took" : 4,
 "timed_out" : false,
 "_shards" : {
  "total" : 1,
  "successful" : 1,
  "skipped" : 0,
  "failed" : 0
 },
 "hits" : {
  "total" : {
   "value" : 4,
   "relation" : "eq"
  },
  "max_score" : 0.8388366,
  "hits" : [
   {
    "_index" : "mall",
    "_type" : "_doc",
    "_id" : "4",
    "_score" : 0.8388366,
    "_source" : {
     "name" : "tv4",
     "description" : "USB, HDMI, DisplayPort",
     "vote" : 0.7
    }
   },
   {
    "_index" : "mall",
    "_type" : "_doc",
    "_id" : "2",
    "_score" : 0.7428025,
    "_source" : {
     "name" : "tv2",
     "description" : "USB, HDMI",
     "vote" : 0.99
    }
   },
   {
    "_index" : "mall",
    "_type" : "_doc",
    "_id" : "1",
    "_score" : 0.7352994,
    "_source" : {
     "name" : "tv1",
     "description" : "USB, DisplayPort",
     "vote" : 0.98
    }
   },
   {
    "_index" : "mall",
```



```
      "_type" : "_doc",
      "_id" : "3",
      "_score" : 0.03592815,
      "_source" : {
        "name" : "tv3",
        "description" : "USB",
        "vote" : 0.5
      }
    }
  ]
 }
}
```

- 用相对好评率计算总分

  总分的计算公式为“new_score = query_score * inline”，本示例中设置的好评率阈值为0.8，当vote>0.8时，inline取值为1；当vote≤0.8时，inline取值为0.5。执行命令如下：

```
GET /mall/_doc/_search?pretty
{
  "query":{
    "function_score":{
      "query":{
        "bool":{
          "should":[
            {"match":{"description":"USB"}},
            {"match":{"description":"HDMI"}},
            {"match":{"description":"DisplayPort"}}
          ]
        }
      },
      "script_score": {
        "script": {
          "params": {
            "threshold": 0.8
          },
          "inline": "if (doc[\"vote\"].value > params.threshold) {return 1;} return 0.5;"
        }
      },
      "boost_mode":"multiply",
      "max_boost":10
    }
  }
}
```

  返回结果如下所示，按照总分由高到低的顺序排列查询结果。

```
{
  "took" : 4,
  "timed_out" : false,
  "_shards" : {
    "total" : 1,
    "successful" : 1,
    "skipped" : 0,
    "failed" : 0
  },
  "hits" : {
    "total" : {
      "value" : 4,
      "relation" : "eq"
    },
    "max_score" : 0.75030553,
    "hits" : [
      {
        "_index" : "mall",
        "_type" : "_doc",
        "_id" : "1",
        "_score" : 0.75030553,
        "_source" : {
          "name" : "tv1",
```



```
        "description" : "USB, DisplayPort",
        "vote" : 0.98
      }
    },
    {
      "_index" : "mall",
      "_type" : "_doc",
      "_id" : "2",
      "_score" : 0.75030553,
      "_source" : {
        "name" : "tv2",
        "description" : "USB, HDMI",
        "vote" : 0.99
      }
    },
    {
      "_index" : "mall",
      "_type" : "_doc",
      "_id" : "4",
      "_score" : 0.599169,
      "_source" : {
        "name" : "tv4",
        "description" : "USB, HDMI, DisplayPort",
        "vote" : 0.7
      }
    },
    {
      "_index" : "mall",
      "_type" : "_doc",
      "_id" : "3",
      "_score" : 0.03592815,
      "_source" : {
        "name" : "tv3",
        "description" : "USB",
        "vote" : 0.5
      }
    }
  ]
 }
}
```


15 常见问题

15.1 产品咨询类

15.1.1 什么是区域和可用区

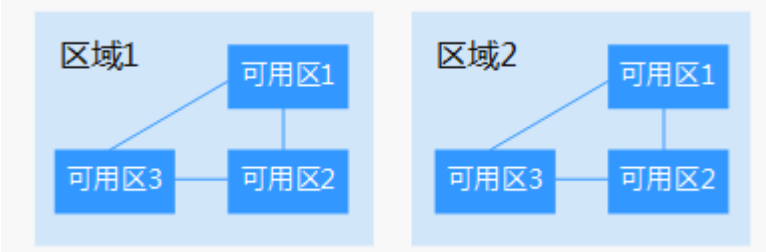
什么是区域、可用区?

区域和可用区用来描述数据中心的位置，您可以在特定的区域、可用区创建资源。

- 区域（Region）指物理的数据中心。每个区域完全独立，这样可以实现最大程度的容错能力和稳定性。资源创建成功后不能更换区域。
- 可用区（AZ，Availability Zone）是同一区域内，电力和网络互相隔离的物理区域，一个可用区不受其他可用区故障的影响。一个区域内可以有多个可用区，不同可用区之间物理隔离，但内网互通，既保障了可用区的独立性，又提供了低价、低时延的网络连接。

图15-1阐明了区域和可用区之间的关系。

图 15-1 区域和可用区



如何选择区域?

建议就近选择靠近您或者您的目标用户的区域，这样可以减少网络时延，提高访问速度。

如何选择可用区?

是否将资源放在同一可用区内，主要取决于您对容灾能力和网络时延的要求。

- 如果您的应用需要较高的容灾能力，建议您将资源部署在同一区域的不同可用区内。
- 如果您的应用要求实例之间的网络延时较低，则建议您将资源创建在同一可用区内。

区域和终端节点

当您通过API使用资源时，您必须指定其区域终端节点。更多信息，请参考《 云搜索服务API参考 》的“终端节点”章节。

15.1.2 云搜索服务如何保证数据和业务运行安全

云搜索服务主要从以下几个方面保障数据和业务运行安全：

- 网络隔离

  整个网络划分为2个平面，即业务平面和管理平面。两个平面采用物理隔离的方式进行部署，保证业务、管理各自网络的安全性。

  - 业务平面：主要是集群的网络平面，支持为用户提供业务通道，对外提供数据定义、索引、搜索能力。
  - 管理平面：主要是管理控制台，用于管理云搜索服务。
- 主机安全

  云搜索服务提供如下安全措施：

  - 通过VPC安全组来确保VPC内主机的安全。
  - 通过网络访问控制列表（ ACL ），可以允许或拒绝进入和退出各个子网的网络流量。
  - 内部安全基础设施（ 包括网络防火墙、入侵检测和防护系统 ）可以监视通过IPsec VPN连接进入或退出VPC的所有网络流量。
- 数据安全

  在云搜索服务中，通过多副本、集群跨az部署、索引数据第三方（ OBS ）备份功能保证用户的数据安全。

15.1.3 用户平时需要关注云搜索服务的哪些监控指标

用户需要关注的监控指标为磁盘使用率和集群健康状态。用户可以登录到云监控服务，根据实际应用场景配置告警提示，当收到告警，可采取相应措施消除告警。

配置示例：

- 如果在某段时间内（ 如5min ），磁盘使用率出现多次（ 如5次 ）不低于某特定值（ 如85% ）的情况，则发出相应告警。
- 如果在某段时间内（ 如5min ），集群健康状态出现多次（ 如5次 ）大于0的情况，则发出相应告警。

采取措施：

- 收到与磁盘使用率有关的告警时，可以调查磁盘空间消耗，查看是否可以从集群节点中删除数据或是将数据存档到其他系统以释放空间，或者扩容磁盘。

- 收到与集群健康状态有关的告警时，可以查看集群的分片分配是否正常以及Shard是否已丢失，在Cerebro上查看进程是否发生重启。

15.1.4 云搜索服务有哪些存储选项

云搜索服务采用EVS和本地磁盘存储用户的索引。在集群创建过程中，用户可指定EVS的类型及规格（即卷大小）。

- 支持EVS类型有普通I/O、高I/O、超高I/O。
- 针对不同的ECS，其对应的EVS卷大小限制根据创建集群选择的节点规格而定。

15.1.5 云搜索服务存储容量的上限是多少

创建集群过程中，最少可创建1个节点，最多可创建200个节点，其中每个节点（对应一个ECS）可挂载一定数量的EVS。可参考不同ECS挂载EVS卷大小的不同，计算出云搜索服务存储容量的总大小，EVS卷大小根据创建集群选择的节点规格而定。

15.1.6 有哪些工具可以使用云搜索服务

管理云搜索服务，或使用其搜索引擎的API，提供了如下三种方式。可以基于已构建好的请求消息发起请求。

- curl

  curl是一个命令行工具，用来执行各种URL操作和信息传输。curl充当的是HTTP客户端，可以发送HTTP请求给服务端，并接收响应消息。curl适用于接口调试。关于curl详细信息请参见**https://curl.haxx.se/**。

- 编码

  通过编码调用接口，组装请求消息，并发送处理请求消息。

- REST客户端

  Mozilla Firefox、Google Chrome都为REST提供了图形化的浏览器插件，发送处理请求消息。

  – 针对Firefox，请参见**Firefox REST Client**。

  – 针对Chrome，请参见**Postman**。

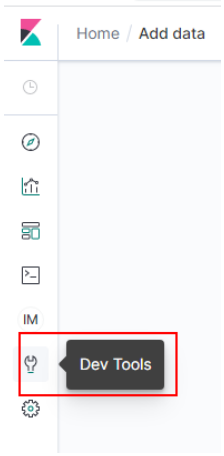
15.1.7 申请的集群节点磁盘空间会有哪些开销

占用集群节点磁盘空间的日志及文件如下所示：

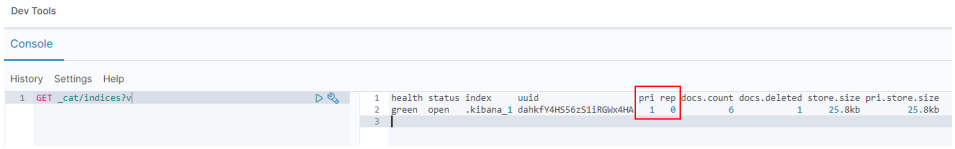
- 日志文件：Elasticsearch日志
- 数据文件：Elasticsearch索引文件
- 其他文件：集群配置文件
- 操作系统：默认余留5%的存储空间

15.1.8 在 CSS 的 console 界面怎么查看集群的分片数以及副本数?

1. 登录console控制台。
2. 在集群管理页面，选择需要查看的集群操作列的Kibana。
3. 登录Kibana界面，选择**Dev Tools**。



4. 在**Dev Tools**的Console界面中执行**GET _cat/indices?v**命令，查询集群分片数和副本数。如图，pri列表示该索引分片数，rep列表示副本数。索引一旦创建，pri无法修改的，rep可以动态修改。



15.1.9 云搜索服务使用的数据压缩算法是什么？

云搜索服务支持的数据压缩算法有两种：一种是Elasticsearch默认的**LZ4算法**，另一种是**best_compression算法**。

- **LZ4算法**

  lz4算法是Elasticsearch的默认压缩算法，该算法对数据的解压/压缩效率很快，但压缩率较低一些。

  压缩算法的实现流程：压缩过程以至少4个bytes为扫描窗口查找匹配，每次移动1byte进行扫描，遇到重复的就进行压缩。该算法适用于读取量大、写入量小的场景。

- **best_compression算法**

  除了默认的LZ4算法，云搜索服务还支持自定义best_compression算法。该算法适用于写入量大、索引存储成本高的场景，例如日志场景、时序分析场景等，可以大大降低索引的存储成本。

执行如下命令，可以将默认压缩算法（LZ4算法）切换为best_compression算法：

```
PUT index-1
{
   "settings": {
      "index": {
         "codec": "best_compression"
      }
   }
}
```

两者比较，LZ4算法在解压/压缩速率方面更快一些，而best_compression算法在压缩率和解压率方面则更优秀一些。

15.2 功能使用相关

15.2.1 Elasticsearch 是否支持不同 VPC 之间的数据迁移?

Elasticsearch不支持直接迁移不同VPC之间的数据，但是可以通过以下2种方式进行迁移。

方法一：

可以使用备份与恢复功能迁移集群数据。

方法二：

1. 打通VPC网络，建立对等链接。
2. 打通网络后，使用Logstash进行数据迁移。

15.2.2 如何跨 Region 迁移 CSS 集群?

CSS集群不支持直接迁移，但可以通过OBS桶备份和恢复的方式进行数据迁移实现集群迁移。

- 如果OBS桶在同一个区域，请参考备份与恢复索引进行集群迁移。
- 如果OBS桶跨区域，请先参考配置跨区域复制复制进行跨区域复制OBS桶，再参考备份与恢复索引进行集群迁移。

说明

- 在跨区域复制之前，要保证目标集群设置的快照文件夹为空，否则无法将快照信息刷新到目标集群的快照列表中。
- 每次迁移都需要将文件夹置空。

15.2.3 如何设置云搜索服务的慢查询日志的阈值?

云搜索服务的慢查询日志设置和elasticsearch 保持一致，通过 _settings接口设置。例如，您可以在Kibana中执行如下样例，设置索引级别。

```
PUT /my_index/_settings
{
    "index.search.slowlog.threshold.query.warn": "10s",
    "index.search.slowlog.threshold.fetch.debug": "500ms",
    "index.indexing.slowlog.threshold.index.info": "5s"
}
```

- 查询慢于10秒输出一个WARN日志。
- 获取慢于500毫秒输出一个DEBUG日志。
- 索引慢于5秒输出一个INFO日志。

详细可参考官网：https://www.elastic.co/guide/cn/elasticsearch/guide/current/logging.html

15.2.4 如何更新 CSS 生命周期策略?

CSS生命周期实现使用的是Open Distro的ISM。此处简单介绍不涉及ISM template的策略更新步骤，若要配置有关ISM template的策略可以参考**Open Distro文档**。

1. 当创建一个policy时，系统会往**.opendistro-ism-config**索引中写入一条数据，这条数据的“_id”就是policy的名字，内容是policy的定义。

图 15-2 写入一条数据

2. 将policy和索引绑定以后，系统会再往**.opendistro-ism-config**索引中写入一条数据。这条数据的初始状态如下图所示。

图 15-3 数据初始状态

3. 执行explain命令，此时返回的内容只有一条policy的id。

```
GET _opendistro/_ism/explain/data2
{
  "data2" : {
    "index.opendistro.index_state_management.policy_id" : "policy1"
```



```
 }
}
```

之后Open Distro会执行一个初始化的流程，将policy的内容填到这条数据中，初始化以后的数据如下图所示。

图 15-4 初始化后数据

初始化结束后，policy中的min_index_age都会被复制过来。

说明

如果此时去更新policy的内容，已经完成初始化流程的索引是完全不感知的，因为他已经将老的policy的内容复制了一份，更新policy的时候不会去更新复制的那部分内容。

4. 修改完policy以后，执行change_policy API完成策略更新，如下所示。

```
POST _opendistro/_ism/change_policy/data1
{
  "policy_id": "policy1"
}
```

15.2.5 如何批量设置索引副本数为 0?

1. 登录集群Kibana界面，在Kibana的左侧导航中选择“Dev Tools”。
2. 执行命令**PUT /*/_settings{"number_of_replicas":0}** 。

说明

*可能会匹配安全索引，不建议执行。建议执行批量操作需要的对应索引。如：**PUT /test*/_settings{"number_of_replicas":0}**。

15.2.6 为什么新创建的索引分片全部被分配到一个 node 节点上?

原因分析

新建索引分片被集中分配于一个node节点上可能有以下原因:

- 之前索引的分配导致某个节点上的shards数量过少，新建索引shards分配被balance.shard参数主导，为了平衡所有索引的全部分片，将shards集中分配在数量过少的节点上。
- 节点扩容，当新节点加入时新节点上的shards数量为0，此时集群会自动进行rebalance，但是rebalance需要时间，此时新建索引很容易会被balande.shard参数主导，平衡所有索引的分片，即都分配在新节点上看起来更加平衡。

涉及集群平衡性shard分配主要有两个配置参数:

cluster.routing.allocation.balance.index（默认值0.45f）

cluster.routing.allocation.balance.shard（默认值0.55f）

说明

- “balance.index”：值越大，shard分配越倾向于使得每个索引的所有分片在节点上均匀分布，如a索引共有6个shards，数据节点有3个，该配置值倾向于让a索引2、2、2平衡分配
- “balance.shard”：值越大，shard分配越倾向于使得所有分片（所有索引的）在节点上平衡，如索引a有2个shards，索引b有4个shards，该配置倾向于所有6个分片进行2、2、2平衡分配。
- balance.index和balance.shard共同负责shards分配。

解决方案

当新建的索引分片被全部分配在一个node节点上时，有以下2种解决办法:

1. 扩容集群需要新建索引时，按照如下所示设置对应参数。
   ```
   "index.routing.allocation.total_shards_per_node": 2
   ```
   即单个索引在每个节点上最多分配2个shards。其中，具体每个节点最多分配多少个shards，请根据集群数据节点个数、索引分片（主、副）的数量自行决定。
2. 如果是shards集中分配在数量过少的节点上导致索引shards分配到同一个节点上，可以使用POST _cluster/reroute的move命令迁移分片到其他节点，rebalance模块会自动分配其他更合适的分片与其交换节点。根据具体业务使用场景可以适当调节balance.index，balance.shard配置。

15.2.7 如何查询快照信息?

前提条件

集群开启了快照，并且设置了快照信息。

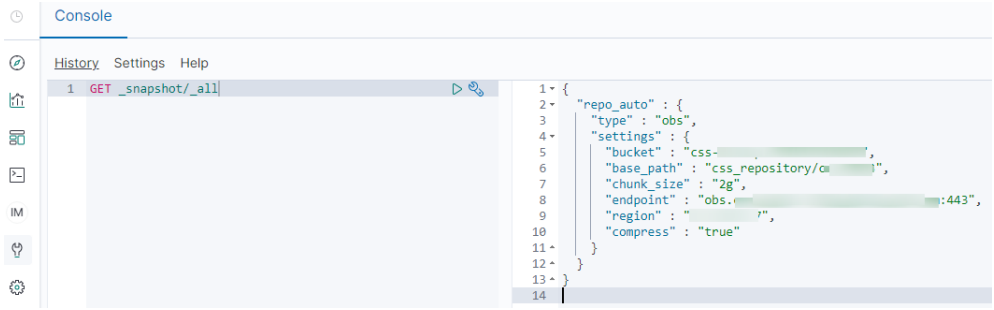
快照查询

1. 在云搜索服务的“集群管理”页面上，单击集群“操作”列的“Kibana”访问集群。
2. 在Kibana的左侧导航中选择“Dev Tools”，单击“Get to work”，进入Console界面。

Console左侧区域为输入框，右侧为结果输出区域， 为执行命令按钮。

3. 执行命令**GET _snapshot/_all**，查询所有仓库信息，如下图所示。

图 15-5 查询所有仓库信息



- bucket：OBS桶名。
- base_path：路径名称。前缀默认固定，后面是集群名称。
- endpoint：OBS域名。
- region：所在region。

4. 查询指定快照信息。
   a. 执行**GET _snapshot/repo_auto/_all**命令，查询当前仓库下面所有的快照列表。

图 15-6 快照信息

   - snapshot：快照名称。
   - state：快照状态。
   - start_time、start_time_in_millis、end_time、end_time_in_millis：快照时间。
   - shards：shards个数。total表示总共的个数。failed表示失败的个数。successful表示成功的个数。

   b. 执行**GET _snapshot/repo_auto/$snapshot-xxx**，查询指定快照信息。

- **$snapshot-xxx**需根据实际情况替换为具体的快照名称。
- **repo_auto**后面跟快照名称，也可以跟通配符。

5. （可选）删除指定快照信息。

   如果要删除指定的快照，执行**DELETE _snapshot/repo_auto/$snapshot-xxx**。

   $snapshot-xxx需根据实际情况替换为具体的快照名称。

15.2.8 购买的低版本集群是否可以升级为高版本集群

不支持直接升级集群版本。但是可以通过购买高版本的集群，进行集群迁移，实现集群版本升级。

1. 创建集群：在同一个region下，创建一个更高版本的集群。
2. 迁移集群：通过备份与恢复索引功能可实现集群迁移。

15.2.9 集群被删除后是否还能恢复?

如果被删除的集群启用过快照功能，且OBS桶中创建的快照并未被删除，则可以通过OBS桶中存储的快照信息恢复集群。否则，被删除的集群无法被恢复，因此请谨慎操作删除任务。

通过OBS桶中存储的快照信息恢复被删除集群的操作步骤：

1. 登录云搜索服务管理控制台。
2. 单击右上角的“创建集群”新建一个集群，并启用集群快照功能。其中，“OBS桶”和“备份路径”填写被删除集群存放快照信息的OBS桶和路径。

   如果要在其他已创建好的集群上恢复被删集群的数据，也需要将已创建好的集群快照的OBS桶”和“备份路径”参数配置为被删除集群存放快照信息的OBS桶和路径。

   须知

   新集群和被删集群要在同一个region下，集群的版本要等于或高于被删集群，新集群的节点数至少要大于被删集群节点数的一半，否则集群可能恢复失败。

3. 当新建集群的“集群状态”会变为“可用”时，单击集群名称进入“基本信息”页面。
4. 在左侧导航栏选择“集群快照”，进入“集群快照”管理页面。

   在快照管理列表中，可以看到被删除集群的快照信息。若没有显示，请等待几分钟后刷新查看。
5. 单击快照“操作”列的“恢复”，弹出“恢复”页面。
6. 在“恢复”页面配置集群的恢复参数。

   “索引”：指定需要进行恢复的索引名称，默认为空。如保持默认值，即不指定索引名称，则表示恢复所有的索引数据。0～1024个字符，不能包含空格和大写字母，且不能包含"\<|>/?特殊字符。

   “索引名称匹配模式”：在恢复时，可以根据文本框中定义的过滤条件去恢复符合条件的索引，过滤条件请使用正则表达式。默认值“index_(.+)”表示所有的索引。0～1024个字符，不能包含空格和大写字母，且不能包含"\<|>/?,特殊字符。

   “索引名称替换模式”：索引重命名的规则。默认值“restored_index_$1”表示在所有恢复的索引名称前面加上“restored_”。0～1024个字符，不能包含空格

和大写字母，且不能包含"\<|>/?,特殊字符。在设置“索引名称替换模式”时，“索引名称匹配模式”和“索引名称替换模式”需要同时设置才会生效。

“集群”：选择需要进行恢复的集群名称，可选择当前集群或者其他集群。只能选择处于“可用”状态的集群，如果快照所属的集群处于“不可用”状态，那么也无法将快照恢复到本集群。恢复到其他集群时，目标集群中的Elasticsearch版本不低于本集群。如果已选择其他集群，且该集群中存在同名的索引，则恢复完成后，该同名的索引中的数据将会被覆盖，请谨慎操作。

7. 单击“确定”开始恢复。恢复成功，快照列表中“任务状态”将变更为“恢复成功”，索引数据将根据快照信息重新生成。

15.2.10 如何修改 Elasticsearch 集群的 TLS 算法?

目前CSS在7.6.2及以上版本支持修改TLS算法。

1. 登录云搜索服务控制台。
2. 选择“集群管理”进入集群列表。
3. 选择需要修改的集群，单击集群名称，进入集群基本信息页面。
4. 选择“参数配置”，单击“编辑”，展开“自定义”，单击“添加”。

   在自定义参数中添加“参数名”称为**opendistro_security.ssl.http.enabled_ciphers**，“参数值”为**['TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256', 'TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384']**

   说明

   如果“参数值”为多个的算法协议，需要一个中括号包围。如果“参数值”为单个的算法协议，需要单引号引起来。

5. 修改完成后，单击上方的“提交”弹出“提交配置”窗口，确认参数无误后勾选“参数修改后需要手动重启才能生效”，单击“确定”。

   当下方的参数修改列表显示“作业状态”为“成功”时，表示修改保存成功。

6. 返回集群列表，单击集群操作列的“更多 > 重启”重启集群，使修改的配置生效。

15.2.11 Elasticsearch 集群如何设置 search.max_buckets 参数?

问题描述

CSS默认限制查找条目为10000，如果查询结果在分片上找到的条目超过了限定的10000个，需要调大search.max_buckets参数值。

解决方案

在kibana的“Dev Tools”页面执行如下命令：

```
PUT _cluster/settings
{
   "persistent": {
      "search.max_buckets": 20000
   }
}
```


15.2.12 Elasticsearch 集群中某个客户端节点的 node.roles 为 i 表示该节点是 ingest 节点吗?

问题描述

集群某个客户端节点的“node.roles”为“i”表示该节点是ingest节点吗?

- 如果客户端节点是ingest节点，那么集群中是否存在Coordinating only node，所有节点都是Coordinating node分摊来客户端请求吗?
- 如果没有ingest业务时，那么客户端节点是不是就处于空闲状态?

解决方案

集群节点的“node.roles”为“i”时，表示集群的客户端节点上启用了ingest节点模式。

- Elasticsearch的“coordinating only node”在CSS服务中称为“client node”，如果集群中没有设置client node，则所有节点都是client node共同分摊客户端请求。
- ingest节点相当于一套ELK，用于数据转换，当没有ingest业务时，客户端节点也不会闲置。

15.2.13 Elasticsearch 7.x 集群如何在 index 下创建 type?

在Elasticsearch 7.x版本中，去掉了type概念，在7.x及以后的版本中，index都不再支持创建type。

若需要强制使用，可以在命令中添加“include_type_name=true”强制使用type类型。

```
PUT _template/urldialinfo_template?include_type_name=true
```

执行命令后，界面会有提示：

```
“#! Deprecation: [types removal] Specifying include_type_name in put index template requests is deprecated. The parameter will be removed in the next major version. ”
```

15.3 安全模式集群相关

15.3.1 filebeat 版本与集群版本的关系

- 非安全模式集群：不限制。
- 安全模式集群：需使用跟集群版本配套的filebeat oss版本，请参考https://www.elastic.co/cn/downloads/past-releases#filebeat-oss进行下载。

15.3.2 如何获取 CSS 服务的安全证书?

CSS服务只有启用HTTPS访问的安全集群才能下载安全证书（CloudSearchService.cer）。

1. 登录云搜索服务控制台。
2. 选择“集群管理”进入集群列表。

3. 单击对应集群的名称，进入集群基本信息页面。
4. 在“基本信息”页面，单击“安全模式”后面的“下载证书”。

图 15-7 下载证书

15.3.3 如何转换 CER 安全证书的格式?

启用了HTTPS访问的安全集群可以下载CSS服务安全证书（CloudSearchService.cer）。而大多数软件支持“.pem”或“.jks”格式的证书，因此要对安全证书进行格式转换。

- 将安全证书从“.cer”格式转换为“.pem”格式。
  ```
  openssl x509 -inform der -in CloudSearchService.cer –out newname.pem
  ```
- 将安全证书从“.cer”格式转换为“.jks”格式。
  ```
  keytool -import -alias newname -keystore ./truststore.jks -file ./CloudSearchService.cer
  ```

其中，*newname*是由用户自定义的证书名称。

执行命令后，会提示设置证书密码，并确认密码。请保存该密码，后续接入集群会使用。

15.4 资源使用和更改相关

15.4.1 如何使用 Elasticsearch 清理过期数据，释放磁盘存储空间?

- 删除单条索引数据命令。

  curl -XDELETE http://IP:9200/索引名

  说明

  IP：任意一个集群节点的IP地址。
- 删除某一天logstash的所有数据命令，例如删除19号所有数据。

  非安全模式集群：**curl -XDELETE 'http://IP:9200/logstash-2017.06.19*'**

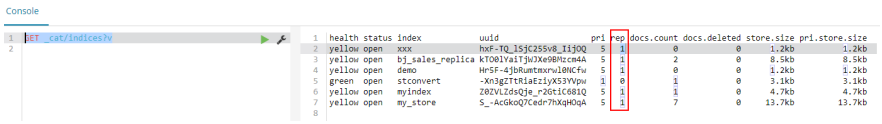
  安全模式集群：**curl -XDELETE -u username:password 'https://IP:9200/logstash-2017.06.19' -k**

说明

- username：管理员帐户名默认为admin。
- password：创建集群时设置的密码。
- IP：任意一个集群节点的IP地址。

15.4.2 如何配置 CSS 集群双副本?

1. 在kibana里执行**GET _cat/indices?v**命令确认集群副本的数目。如果rep参数列为1，说明是双副本。



2. 如果不是，可以如下执行命令设置副本数。

   PUT /index/_settings

   {

   "number_of_replicas" : 1 //表示需要设置的副本数

   }

   说明

   index为需要修改的索引名称，需根据实际情况进行修改。

15.4.3 如何清理索引数据?

- 手动清理：在kibana里执行**DELETE /my_index**命令。
- 自动化定期清理：可以写定时任务调用清理索引的请求，定期执行。CSS支持Opendistro Index State Management。详见：https://opendistro.github.io/for-elasticsearch-docs/docs/im/ism/

15.4.4 json 里设置了 1 个分片，是否可以通过修改配置，达到 4 分片，2 副本的效果

索引一旦创建成功，主shards数量不可变。

修改副本数，可通过在kibana中执行以下命令：

```
PUT /indexname/_settings
{
"number_of_replicas" : 1      //表示需要设置的副本数
}
```

说明

index为需要修改的索引名称，需根据实际情况进行修改。

15.4.5 Elasticsearch 集群分片过多会有哪些影响

1. 集群创建分片的速度随着集群分片数量增多而逐渐减低。
2. 触发Elasticsearch自动创建index时，创建速度变慢会导致大量写入请求堆积在内存中，严重时可导致集群崩溃。

3. 分片过多时，如果不能及时掌控业务的变化，可能经常遇到单分片记录超限、写入拒绝等问题。

15.4.6 Elasticsearch 集群设置默认分页返回最大条数

解决方案

- 方法1：

  打开kibana，在devtools界面执行如下命令：

  ```
  PUT _all/_settings?preserve_existing=true
  {
  "index.max_result_window" : "10000000"
  }
  ```

- 方法2：

  后台执行如下命令进行设置：

  ```
  curl –XPUT ‘http://localhost:9200/_all/_setting?preserve_existing=true’-d
  {
  "index.max_result_window":"1000000"
  }
  ```

⚠ 注意

该配置会相应的消耗内存与CPU，请谨慎设置。

15.4.7 使用 delete_by_query 命令删除数据后，为什么磁盘使用率反而增加?

使用**delete_by_query**命令删除数据并不是真正意义上的物理删除，它仅仅是对数据增加了删除标记。当再次搜索的时，会搜索全部数据后再过滤掉带有删除标记的数据。

因此，该索引所占的空间并不会因为执行磁盘删除命令后马上释放掉，只有等到下一次段合并时才真正的被物理删除，这个时候磁盘空间才会释放。

相反，在查询带有删除数据时需要占用磁盘空间，这时执行磁盘删除命令不但没有被释放磁盘空间，反而磁盘使用率上升了。

15.4.8 CSS 集群如何清理缓存?

- **清理fielddata**

  进行聚合和排序时，会使用**fielddata**数据结构，会占用较大内存。

  a. 在Kibana执行如下命令，查看索引的**fielddata**占用情况。

  ```
  DELETE /_search/scroll
  {
  "scroll_id" :
  "DXF1ZXJ5QW5kRmV0Y2gBAAAAAAAAAD4WYm9laVYtZndUQlNsdDcwakFMNjU1QQ=="
  }
  ```

  b. 当**fielddata**占用内存过高时，可以执行如下命令清理**fielddata**。

  ```
  POST /test/_cache/clear?fielddata=true
  ```

  “test”为fielddata占用内存较高的索引名称。

- **清理segment**

  每个segment的FST结构都会被加载到内存中，并且这些内存是不会被垃圾回收的。因此如果索引的segment数量过大，会导致内存使用率较高，建议定期进行清理。

  a. 在Kibana执行如下命令，查看各节点的segment数量和占用内存大小。
  ```
  GET /_cat/nodes?v&h=segments.count,segments.memory&s=segments.memory:desc
  ```
  b. 若segment占用内存过高时，可以通过删除部分不用的索引、关闭索引或定期合并不再更新的索引等方式释放内存。

- **清理cache**

  在Kibana执行如下命令清理cache。
  ```
  POST _cache/clear
  ```

15.4.9 Elasticsearch 集群平均已用内存比例达到 98%

问题现象

查看集群监控发现，ES集群“平均已用内存比例”一直处于98%，用户担心内存比例过高是否对集群有影响。

问题原因

在ES集群中，Elasticsearch会占用50%内存，另外50%内存会被Lucene用于缓存文件，因此节点内存占用会一直很高，平均已用内存比例达到98%是正常现象，请您放心使用。

解决方案

您可以关注“最大JVM堆使用率”和“平均JVM堆使用率”这两个指标来监控集群内存使用情况。

15.5 组件使用

15.5.1 云搜索服务是否支持 SearchGuard 插件的安装?

不支持。

云搜索服务提供了安全模式集群与SearchGuard插件功能一样。

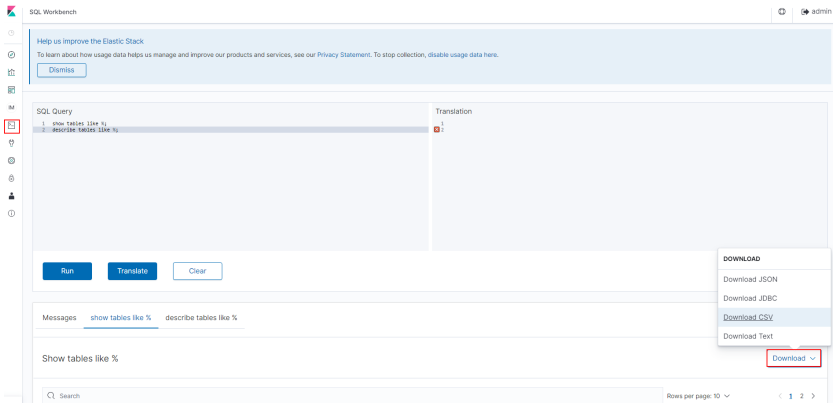
15.6 Kibana 使用相关

15.6.1 Kibana 是否支持导出数据功能?

Kibana导出数据需要依赖SQL Workbench插件，目前云搜索服务只有Elasticsearch 7.6.2及以上的版本支持。

在Kibana的SQL Workbench里，输入Elasticsearch SQL语句可以查询数据，也可以“Download”导出数据，支持自定义导出1~200条数据，缺省导出200条数据。

图 15-8 SQL Workbench



15.6.2 Elasticsearch 集群在 kibana 如何查询索引数据

在kibana可以通过API查询索引数据，命令如下：

```
GET indexname/_search
```

返回数据如下图所示：

图 15-9 返回数据

```
{
  "took": 5,
  "timed_out": false,
  "_shards": {
    "total": 5,
    "successful": 5,
    "skipped": 0,
    "failed": 0
  },
  "hits": {
    "total": 3,
    "max_score": 2.0794415,
    "hits": [
      {
        "_index": "book",
        "_type": "novel",
        "_id": "7",
        "_score": 2.0794415,
        "_source": {
          "author": "孙七",
          "title": "Elasticsearch入门",
          "word_count": 3000,
          "publish_date": "2017-10-01"
        }
      },
      {
```


说明

- “took”：耗时几毫秒。
- “time_out”：是否超时。
- “_shard”：数据被拆到了5个分片上，搜索时使用了5个分片，5个分片都成功地返回了数据，失败了0个，跳过了0个。
- “hits.total”：查询结果的数量，3个document。
- “max_score”：就是document对于一个search的相关度的匹配分数，越相关，就越匹配，分数也越高。
- “hits.hits”：包含了匹配搜索的document的详细数据。

15.7 访问集群相关

15.7.1 ECS 无法连接到集群

遇到该问题，请按照如下操作步骤排查解决。

1. 先确认ECS实例和集群是否在同一个VPC。
   - 如果在，执行步骤2。
   - 如果不在，需要重新创建ECS实例，使之和集群在同一个VPC下。
2. 查看集群的安全组的出方向和入方向是否已允许9200端口（TCP协议），或者允许的端口范围已包含9200端口（TCP协议）。
   - 如果是，执行步骤3。
   - 如果不是，请前往VPC页面，设置“安全组”的出方向和入方向已允许9200端口或允许的端口范围已包含9200端口。
3. 查看ECS实例是否添加安全组。
   - 如果有，检查安全组的配置规则是否满足要求，在集群“基本信息”页面，可以查看“安全组信息”。然后执行步骤4。
   - 如果没有，从ECS的实例详情页面，进入VPC页面，选择“安全组”，添加安全组。
4. 在ECS实例上，测试是否可以正常连接到集群。

   ***ssh** <节点的内网访问地址和端口号>*

   说明

   当集群包含多个节点时，需要逐个节点测试是否可以正常连接到该集群中的每个节点。

   - 如果可以通信，说明网络是正常的。
   - 如果端口不通，请联系技术支持协助排查。

15.7.2 新建的集群是否可以使用老集群的 IP 地址?

原集群的IP地址无法更换为新集群的IP地址。

15.7.3 CSS 集群想通过外网访问，可以绑定自己的弹性 IP 吗?

不可以使用自己的EIP。如果需要通过公网访问集群，可以参考公网访问。

15.7.4 CSS 集群是否支持采用 x-pack-sql-jdbc 进行客户端连接并查询?

不支持，目前云搜索服务没有集成x-pack组件。

15.8 端口使用

15.8.1 9200 和 9300 端口是否都开放?

都开放。9200端口为外部访问es集群端口，9300为节点之间通讯端口。

访问9300端口有以下几种方式：

- 如果是同VPC同子网内可直接访问。
- 如果是同VPC下跨子网访问，需要单独申请路由配置。
- 如果是不同的VPC不同的子网访问，需要先通过对等连接，打通两个VPC网络，然后单独申请路由配置，联通两个子网。

16 修订记录

发布日期	修订记录
2023-06-20	新增： • 7.10.2版本。 • **产品组件** • **配额说明** • **创建用户并授权使用CSS** • **部署跨AZ集群** • **形态变更概述** • **扩容** • **缩容** • **使用CDM从OBS导入数据到Elasticsearch** • **使用DIS导入本地数据到Elasticsearch** • **向量检索** • **使用Kibana相关操作** • **存算分离** • **流量控制** • **索引监控** • **监控增强** • **最佳实践**
2022-08-23	• 更新优化公共内容： – **约束与限制** – **基本概念** – **安全集群简介** – **快速开始使用Elasticsearch搜索引擎** – **使用Kibana或API导入数据到Elasticsearch** • 删除自定义词库。

发布日期	修订记录
2022-10-09	优化内容：**查询Elasticsearch SQL**
2022-06-30	第一次正式发布。