云搜索服务

最佳实践

文档版本 01

发布日期 2025-01-06

华为云计算技术有限公司

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云计算技术有限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，华为云计算技术有限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。

华为云计算技术有限公司

地址： 贵州省贵安新区黔中大道交兴功路华为云数据中心 邮编：550029

网址： https://www.huaweicloud.com/

目 录

1 Elasticsearch 数据迁移

1.1 Elasticsearch 集群数据迁移方案介绍

表 1-1 Elasticsearch 集群迁移方案

迁移场景	迁移工具	适用场景	使用限制	场景示例
华为云Elasticsearch集群间数据迁移	华为云Logstash	• 适用于低版本的华为云Elasticsearch集群到高版本的数据迁移。 • 适用于将多个华为云Elasticsearch集群的数据整合到一个Elasticsearch集群的场景。	在集群迁移期间，不要对源集群的索引进行增删改操作，避免迁移后源集群与目标集群数据不一致。	通过华为云Logstash实现Elasticsearch集群间数据迁移

迁移场景	迁移工具	适用场景	使用限制	场景示例
	备份与恢复	• 适用于同Region或跨Region、同账号或跨账号的华为云Elasticsearch集群间的数据迁移。 • 适用于低版本的华为云Elasticsearch集群到高版本的数据迁移。 • 适用于将多个华为云Elasticsearch集群的数据整合到一个Elasticsearch集群的场景。	• 目标集群的版本不能低于源集群的版本，版本兼容性分析请参见**Snapshot version compatibility**。 • 目标集群的节点数要大于源集群的节点数的一半，且不能小于源集群的shard副本数。 • 目标集群的CPU、MEM和Disk配置不能低于源集群的配置。	**通过备份与恢复实现华为云Elasticsearch集群间数据迁移**
	读写分离插件	• 适用于同Region或跨Region、同账号或跨账号的华为云Elasticsearch集群间的数据迁移。 • 适用于将多个华为云Elasticsearch集群的数据整合到一个Elasticsearch集群的场景。	源集群和目标集群的版本必须一致，都是7.6.2或7.10.2版本。	**通过读写分离插件实现华为云Elasticsearch集群间数据迁移**
	Reindex API	适用于将多个华为云Elasticsearch集群的数据整合到一个Elasticsearch集群的场景。	在集群迁移期间，不要对源集群的索引进行增删改操作，避免迁移后源集群与目标集群数据不一致。	**通过Reindex API实现Elasticsearch集群间数据迁移**

迁移场景	迁移工具	适用场景	使用限制	场景示例
	ESM	• 适用于低版本的华为云Elasticsearch集群到高版本的数据迁移。 • 适用于将多个华为云Elasticsearch集群的数据整合到一个Elasticsearch集群的场景。	在集群迁移期间，不要对源集群的索引进行增删改操作，避免迁移后源集群与目标集群数据不一致。	**通过ESM实现Elasticsearch集群间数据迁移**
自建Elasticsearch集群迁移至华为云	华为云Logstash	• 适用于低版本的自建Elasticsearch集群到高版本华为云Elasticsearch集群的数据迁移。 • 适用于将多个自建Elasticsearch集群的数据整合到一个华为云Elasticsearch集群的场景。 • 适用于将自建的Elasticsearch服务迁移到华为云平台的场景。	• 在集群迁移期间，不要对源集群的索引进行增删改操作，避免迁移后源集群与目标集群数据不一致。 • 集群间需要保证网络连通。 可以通过给自建Elasticsearch集群配置公网访问打通网络。	**通过华为云Logstash实现Elasticsearch集群间数据迁移**
	备份与恢复	• 适用于低版本的自建Elasticsearch集群到高版本华为云Elasticsearch集群的数据迁移。 • 适用于将多个自建Elasticsearch集群的数据整合到一个华为云Elasticsearch集群的场景。 • 适用于将自建的Elasticsearch服务迁移到华为云平台的场景。	• 目标集群的版本不能低于源集群的版本，版本兼容性分析请参见**Snapshot version compatibility**。 • 备份与恢复不支持动态增量数据同步，建议停止数据更新后再进行备份。 • 自建Elasticsearch集群需要配置公网访问权限才能备份快照到OBS。	**通过S3插件备份与恢复迁移自建Elasticsearch集群至华为云**

迁移场景	迁移工具	适用场景	使用限制	场景示例
	Reindex API	• 适用于将多个自建Elasticsearch集群的数据整合到一个华为云Elasticsearch集群的场景。 • 适用于将自建的Elasticsearch服务迁移到华为云平台的场景。	在集群迁移期间，不要对源集群的索引进行增删改操作，避免迁移后源集群与目标集群数据不一致。	**通过Reindex API实现Elasticsearch集群间数据迁移**
	ESM	• 适用于低版本的自建Elasticsearch集群到高版本华为云Elasticsearch集群的数据迁移。 • 适用于将多个自建Elasticsearch集群的数据整合到一个华为云Elasticsearch集群的场景。 • 适用于将自建的Elasticsearch服务迁移到华为云平台的场景。	在集群迁移期间，不要对源集群的索引进行增删改操作，避免迁移后源集群与目标集群数据不一致。	**通过ESM实现Elasticsearch集群间数据迁移**
	云数据迁移CDM	华为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。	• 需要建立企业内部数据中心到华为云的VPN通道或专线。 • 集群迁移过程中，不能删除源集群的索引数据，否则迁移的数据可能出现异常。	**Elasticsearch整库迁移到云搜索服务**

迁移场景	迁移工具	适用场景	使用限制	场景示例
第三方Elasticsearch集群迁移至华为云	华为云Logstash	• 适用于Elasticsearch集群版本跨度较大的迁移场景，例如从6.X版本迁移至7.X版本。 • 适用于将多个第三方Elasticsearch集群的数据整合到一个华为云Elasticsearch集群的场景。 • 适用于将第三方Elasticsearch服务迁移到华为云平台的场景。	• 在集群迁移期间，不要对源集群的索引进行增删改操作，避免迁移后源集群与目标集群数据不一致。 • 集群间需要保证网络连通。需要建立企业内部数据中心到华为云的VPN通道或专线。	**通过华为云Logstash实现Elasticsearch集群间数据迁移**
	备份与恢复	• 适用于低版本的第三方Elasticsearch集群到高版本华为云Elasticsearch集群的数据迁移。 • 适用于将多个第三方Elasticsearch集群的数据整合到一个华为云Elasticsearch集群的场景。 • 适用于将第三方Elasticsearch服务迁移到华为云平台的场景。	• 目标集群的版本不能低于源集群的版本，版本兼容性分析请参见**Snapshot version compatibility**。 • 备份与恢复不支持动态增量数据同步，建议停止数据更新后再进行备份。 • 第三方存储仓库要配置公网访问才能迁移快照数据。	**通过备份与恢复迁移第三方Elasticsearch集群至华为云**
	Reindex API	• 适用于将多个第三方Elasticsearch集群的数据整合到一个华为云Elasticsearch集群的场景。 • 适用于将第三方Elasticsearch服务迁移到华为云平台的场景。	在集群迁移期间，不要对源集群的索引进行增删改操作，避免迁移后源集群与目标集群数据不一致。	**通过Reindex API实现Elasticsearch集群间数据迁移**

迁移场景	迁移工具	适用场景	使用限制	场景示例
	ESM	• 适用于低版本的第三方Elasticsearch集群到高版本华为云Elasticsearch集群的数据迁移。 • 适用于将多个第三方Elasticsearch集群的数据整合到一个华为云Elasticsearch集群的场景。 • 适用于将第三方Elasticsearch服务迁移到华为云平台的场景。	在集群迁移期间，不要对源集群的索引进行增删改操作，避免迁移后源集群与目标集群数据不一致。	通过ESM实现Elasticsearch集群间数据迁移
	云数据迁移CDM	华为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。	• 需要建立企业内部数据中心到华为云的VPN通道或专线。 • 集群迁移过程中，不能删除源集群的索引数据，否则迁移的数据可能出现异常。	Elasticsearch整库迁移到云搜索服务
华为云Elasticsearch集群迁移至OpenSearch集群	跨引擎升级	适用于CSS服务的Elasticsearch 7.10.2升级至OpenSearch 1.3.6。	• 建议在业务低峰期进行升级操作，避免影响业务。 • 待升级的Elasticsearch集群不能存在正在进行中的任务。	升级Elasticsearch集群版本

1.2 通过华为云 Logstash 实现 Elasticsearch 集群间数据迁移

使用华为云CSS服务的Logstash集群可以实现Elasticsearch集群间的数据迁移。

应用场景

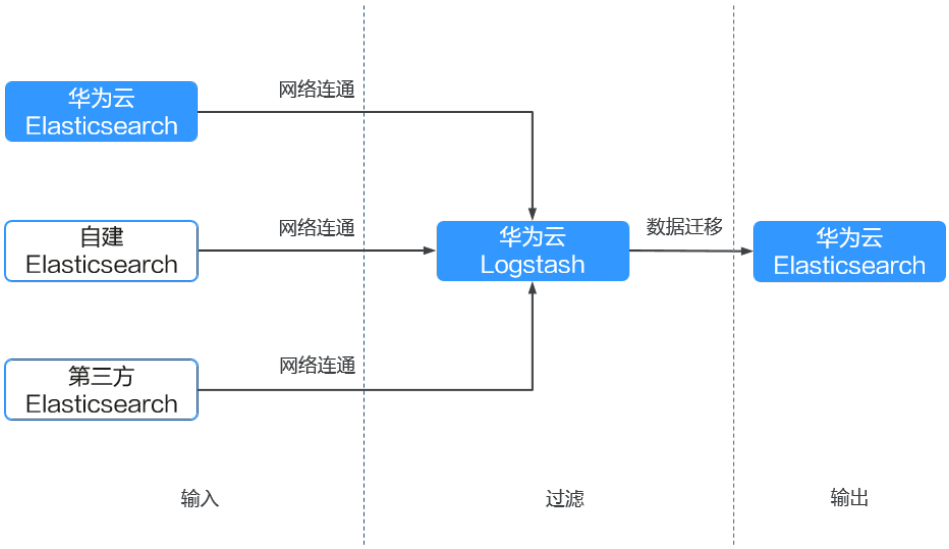
华为云Logstash是一款全托管的数据接入处理服务，兼容开源Logstash的能力，支持用于Elasticsearch集群间数据迁移。

通过华为云Logstash可以实现华为云Elasticsearch、自建Elasticsearch或第三方Elasticsearch迁移至华为云Elasticsearch，该方案常用于以下场景：

- 跨版本迁移：利用Logstash的兼容性和灵活性，实现不同版本间的数据迁移，确保数据在新版本中的可用性和一致性。适用于Elasticsearch集群版本跨度较大的迁移场景，例如从6.X版本迁移至7.X版本。
- 集群合并：使用Logstash进行数据迁移，将多个Elasticsearch集群的数据整合到一个Elasticsearch集群中，实现多个Elasticsearch数据的统一管理和分析。
- 服务迁移上云：将自建的Elasticsearch服务迁移到云平台，以利用云服务的可扩展性、维护简便性和成本效益。
- 变更服务提供商：如果企业当前使用的是第三方Elasticsearch服务，但出于成本、性能或其他战略考虑，希望更换服务提供商至华为云。

方案架构

图 1-1 迁移流程



通过华为云Logstash实现Elasticsearch集群间数据迁移的迁移流程如**图1-1**所示。

1. 输入（Input）：华为云Logstash接收来自华为云Elasticsearch、自建Elasticsearch或第三方Elasticsearch的数据。

   说明

   华为云Elasticsearch、自建Elasticsearch或第三方Elasticsearch数据迁移到华为云Elasticsearch的操作步骤相同，只是获取源集群的访问地址有差异，具体请参见**获取Elasticsearch集群信息**。

2. 过滤（Filter）：华为云Logstash对数据进行清洗和转换。
3. 输出（Output）：华为云Logstash将数据输出到目标设备，如华为云Elasticsearch。

根据业务需求，可以选择全量数据迁移或增量数据迁移。

- 全量数据迁移：使用Logstash进行全量数据迁移，适用于迁移初期或需要确保数据完整性的场景。
- 增量数据迁移：通过Logstash配置增量查询，可以只迁移有增量字段的索引数据。此方法适用于需要持续同步数据或对数据实时性有较高要求的场景。

方案优势

- 高版本兼容性：适用于不同版本的Elasticsearch集群迁移。
- 高效的数据处理能力：Logstash支持批量读写操作，可以大幅度提高数据迁移的效率。
- 并发同步技术：利用slice并发同步技术，可以提高数据迁移的速度和性能，尤其是在处理大规模数据时。
- 配置简单：华为云Logstash的配置相对简单直观，通过配置文件即可实现数据的输入、处理和输出。
- 强大的数据处理功能：Logstash内置了丰富的过滤器，可以在迁移过程中对数据进行清洗、转换和丰富。
- 灵活的迁移策略：根据业务需求，可以灵活选择全量迁移或增量迁移，优化存储使用和迁移时间。

性能影响

使用Logstash迁移集群依托于Scroll API，此API能够高效读取源集群的索引数据，并批量同步至目标集群。这一过程可能会对源集群性能产生影响，具体影响程度取决于目标集群对源集群的读取速度，而读取速度取决于Scroll API的size和slice参数配置。参数配置的详细指导可参考**Reindex API**文档。

- 对于资源消耗较高的集群，建议通过调整size参数来减缓迁移速率，或者选择在业务流量低谷时段进行迁移操作，以减轻对集群资源的影响。
- 对于资源消耗较低的集群，在迁移时可以采用默认参数配置，建议同时监控源集群的性能负载，并根据实际情况适时调整size和slice参数，以优化迁移效率和资源使用。

约束限制

集群迁移过程中，源集群的索引数据不能增删改，否则会导致迁移后的源集群数据和目标集群数据内容不一致。

前提条件

- 源Elasticsearch集群和目标Elasticsearch集群处于可用状态。
- 集群间需要保证网络连通。
  - 如果源集群、Logstash和目标集群在不同VPC，则需要先打通VPC网络建立对等连接。具体操作请参见**对等连接简介**。
  - 如果是自建Elasticsearch集群迁移至华为云，则可以通过给自建Elasticsearch集群配置公网访问打通网络。
  - 如果是第三方Elasticsearch集群迁移至华为云，则需要建立企业内部数据中心到华为云的VPN通道或专线。
- 确认集群的索引已开启“_source”。

  集群索引的“_source”默认是开启的。执行命令**GET {index}/_search**，当返回的索引信息里有“_source”信息时表示已开启。

操作步骤

1. **获取Elasticsearch集群信息**

2. **（可选）迁移索引结构**：通过脚本迁移Elasticsearch集群的索引模板和索引结构。
3. **创建Logstash集群**：创建一个Logstash集群用于迁移数据。
4. **验证集群间的网络连通性**：验证Logstash和源Elasticsearch集群的连通性。
5. 使用Logstash迁移集群
   - 在集群迁移初期或需要确保数据完整性的场景，推荐**使用Logstash全量迁移集群数据**。
   - 在需要持续同步数据或对数据实时性有较高要求的场景，推荐**使用Logstash增量迁移集群数据**。
6. **释放Logstash集群**：当集群迁移完成后，请及时释放Logstash集群。

获取 Elasticsearch 集群信息

在迁移集群前，需要先获取必备的集群信息，用于配置迁移任务。

表 1-2 需要获取的 Elasticsearch 集群信息

集群来源		要获取的信息	获取方式
源集群	华为云Elasticsearch集群	• 源集群的名称 • 源集群的访问地址 • 访问源集群的用户名和密码（仅安全集群涉及）	• 获取集群名称和访问地址请参见3。 • 用户名和密码请联系服务管理员获取。
	自建Elasticsearch集群	• 源集群的名称 • 源集群的公网访问地址 • 访问源集群的用户名和密码（仅安全集群涉及）	联系服务管理员获取。
	第三方Elasticsearch集群	• 源集群的名称 • 源集群的访问地址 • 访问源集群的用户名和密码（仅安全集群涉及）	联系服务管理员获取。
目标集群	华为云Elasticsearch集群	• 目标集群的访问地址 • 访问目标集群的用户名和密码（仅安全集群涉及）	• 获取访问地址请参见3。 • 用户名和密码请联系服务管理员获取。

源集群的来源不同，获取信息的方式不同，此处仅介绍如何获取华为云Elasticsearch集群的信息。

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”。

3. 在Elasticsearch集群列表，获取集群名称和访问地址。

图 1-2 获取集群信息

（可选）迁移索引结构

如果您直接在目标Elasticsearch集群手动创建索引结构，则跳过该步骤。此处提供了一种通过脚本迁移Elasticsearch集群的索引模板和索引结构的方法。

1. 创建弹性云服务器ECS，用于迁移源集群的元数据。
   a. 创建弹性云服务器ECS，ECS的操作系统选择CentOS，规格选择2U4G，且和CSS服务的集群在同一个虚拟私有云和安全组中。
   b. 测试ECS和源集群、目标集群的连通性。

      在ECS执行命令**curl http:// {ip}:{port}**测试连通性，当返回200时，则表示已经连通。

      IP是源集群和目标集群访问地址；port是端口号，默认是9200，请以集群实际端口号为准。

```
curl http://10.234.73.128:9200 # 输入实际的IP地址，此处仅以非安全集群举例。
{
  "name" : "es_cluster_migrate-ess-esn-1-1",
  "cluster_name" : "es_cluster_migrate",
  "cluster_uuid" : "1VbP7-39QNOx_R-llXKKtA",
  "version" : {
    "number" : "6.5.4",
    "build_flavor" : "default",
    "build_type" : "tar",
    "build_hash" : "d2ef93d",
    "build_date" : "2018-12-17T21:17:40.758843Z",
    "build_snapshot" : false,
    "lucene_version" : "7.5.0",
    "minimum_wire_compatibility_version" : "5.6.0",
    "minimum_index_compatibility_version" : "5.0.0"
  },
  "Tagline" : "You Know, for Search"
}
```

2. 准备工具和软件，判断ECS是否可以联网，选择不同的安装方式。
   - 是，选择在线安装工具和软件，直接使用yum和pip安装，具体请参见**3**。
   - 否，选择离线安装工具和软件，下载安装包到虚拟机上执行安装命令，具体请参见**4**。

表 1-3 准备工具和软件

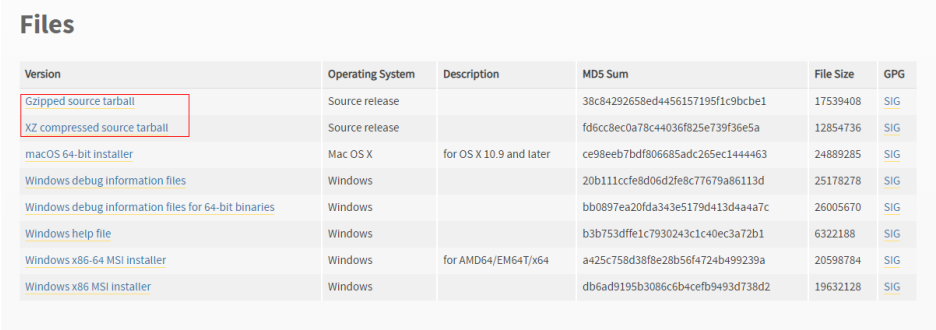
类型	目的	获取位置
Python2	主要用户执行迁移脚本。	**下载地址**，版本选择python 2.7.18。
winscp	Linux上传脚本，跨平台文件传输工具。	**下载Winscp**。

3. 在线安装工具和软件。

a. 执行yum install python2安装python2。
```
[root@ecs opt]# yum install python2
```
b. 执行yum install python-pip安装pip。
```
[root@ecs opt]# yum install python-pip
```
c. 执行pip install pyyaml安装yaml依赖。
d. 执行pip install requests安装requests依赖。

4. 离线安装工具和软件。
   a. 下载python2安装包，下载地址**https://www.python.org/downloads/release/python-2718/**。选择源码下载安装。

图 1-3 下载 python2 安装包



   b. 使用WinSCP工具上传Python安装包到opt目录下，安装python。
```
# 解压python压缩包
[root@ecs-52bc opt]# tar -xvf Python-2.7.18.tgz
Python-2.7.18/Modules/zlib/crc32.c
Python-2.7.18/Modules/zlib/gzlib.c
Python-2.7.18/Modules/zlib/inffast.c
Python-2.7.18/Modules/zlib/example.c
Python-2.7.18/Modules/python.c
Python-2.7.18/Modules/nismodule.c
Python-2.7.18/Modules/Setup.config.in
...
# 解压完成进入目录
[root@ecs-52bc opt]# cd Python-2.7.18
# 检查文件配置安装路径
[root@ecs-52bc Python-2.7.18]# ./configure --prefix=/usr/local/python2
...
checking for build directories... checking for --with-computed-gotos... no value specified
checking whether gcc -pthread supports computed gotos... yes
done
checking for ensurepip... no
configure: creating ./config.status
config.status: creating Makefile.pre
config.status: creating Modules/Setup.config
config.status: creating Misc/python.pc
config.status: creating Modules/ld_so_aix
config.status: creating pyconfig.h
creating Modules/Setup
creating Modules/Setup.local
creating Makefile
# 编译python
[root@ecs-52bc Python-2.7.18]# make
# 安装python
[root@ecs-52bc Python-2.7.18]# make install
```
   c. 安装完成检查，检查python安装结果。
```
# 检查python版本
[root@ecs-52bc Python-2.7.18]# python --version
Python 2.7.5
# 检查pip版本
```



```
[root@ecs-52bc Python-2.7.18]# pip --version
pip 7.1.2 from /usr/lib/python2.7/site-packages/pip-7.1.2-py2.7.egg (python 2.7)
[root@ecs-52bc Python-2.7.18]#
```

5. 准备Elasticsearch集群的索引迁移脚本。

   a. 执行“vi migrateConfig.yaml”配置文件，输入并基于实际信息修改以下内容，执行**wq**保存为**Logstash迁移脚本**。集群信息的获取方式请参见**获取Elasticsearch集群信息**。

```
es_cluster_new:
  # 源集群的名称
  clustername: es_cluster_new
  # 源Elasticsearch集群的访问地址，加上“http://”。
  src_ip: http://x.x.x.x:9200
  # 访问源Elasticsearch集群的用户名和密码，如果为非安全集群则设置为""。
  src_username: ""
  src_password: ""
  # 目标Elasticsearch集群的访问地址，加上“http://”。
  dest_ip: http://x.x.x.x:9200
  # 访问目标Elasticsearch集群的用户名和密码，如果为非安全集群则设置为""。
  dest_username: ""
  dest_password: ""
  # “only_mapping”可以不定义，默认值为false,需要搭配“migrateMapping.py”使用，表示是否只处理这个文件中mapping地址的索引。当设置成true时，则只迁移源集群中和下面mapping的key一致的索引数据；当设置成false时，则迁移源集群中除“.kibana”和“.*”之外的所有索引数据。
  # 迁移过程中会将索引名称与下面的mapping匹配，如果匹配一致，则使用mapping的value作为目标集群的索引名称；如果匹配不到，则使用源集群原始的索引名称。
  only_mapping: false
  # 设置要迁移的索引，key为源集群的索引名字，value为目标集群的索引名字。
  mapping:
    test_index_1: test_index_1
  # “only_compare_index”可以不定义，默认值为false,需要搭配“checkIndices.py”使用，当设置为false会比较所有的索引和文档数量，当设置为true只比较索引数量。
  only_compare_index: false
```

   b. 执行**vi migrateTemplate.py**命令，直接复制输入以下内容无需修改，执行**wq**保存为**索引模板迁移脚本**。

```
# -*- coding:UTF-8 -*-
import sys
import yaml
import requests
import json
import os


def printDividingLine():
    print("<============================================================>")


def loadConfig(argv):
    if argv is None or len(argv) != 2:
        config_yaml = "migrateConfig.yaml"
    else:
        config_yaml = argv[1]
    config_file = open(config_yaml)
    # config = yaml.load(config_file, Loader=yaml.FullLoader)
    return yaml.load(config_file)


def put_template_to_target(url, template, cluster, template_name, dest_auth=None):
    headers = {'Content-Type': 'application/json'}
    create_resp = requests.put(url, headers=headers, data=json.dumps(template),
auth=dest_auth, verify=False)
    if not os.path.exists("templateLogs"):
        os.makedirs("templateLogs")
    if create_resp.status_code != 200:
        print(
            "create template " + url + " failed with response: " + str(
                create_resp) + ", source template is " + template_name)
        print(create_resp.text)
```



```
            filename = "templateLogs/" + str(cluster) + "#" + template_name
            with open(filename + ".json", "w") as f:
                json.dump(template, f)
            return False
        else:
            return True


def main(argv):
    requests.packages.urllib3.disable_warnings()
    print("begin to migration template!")
    config = loadConfig(argv)
    src_clusters = config.keys()
    print("process cluster name:")
    for name in src_clusters:
        print(name)
    print("cluster total number:" + str(src_clusters.__len__()))

    for name, value in config.items():
        printDividingLine()
        source_user = value["src_username"]
        source_passwd = value["src_password"]
        source_auth = None
        if source_user != "":
            source_auth = (source_user, source_passwd)
        dest_user = value["dest_username"]
        dest_passwd = value["dest_password"]
        dest_auth = None
        if dest_user != "":
            dest_auth = (dest_user, dest_passwd)

        print("start to process cluster name：" + name)
        source_url = value["src_ip"] + "/_template"

        response = requests.get(source_url, auth=source_auth, verify=False)
        if response.status_code != 200:
            print("*** get all template failed. resp statusCode:" + str(
                response.status_code) + " response is " + response.text)
            continue
        all_template = response.json()
        migrate_itemplate = []

        for template in all_template.keys():
            if template.startswith(".") or template == "logstash":
                continue
            if "index_patterns" in all_template[template]:
                for t in all_template[template]["index_patterns"]:
                    # if "kibana" in template:
                    if t.startswith("."):
                        continue
                    migrate_itemplate.append(template)

        for template in migrate_itemplate:
            dest_index_url = value["dest_ip"] + "/_template/" + template
            result = put_template_to_target(dest_index_url, all_template[template], name,
template, dest_auth)
            if result is True:
                print('[success] delete success, cluster: %-10s, template %-10s ' % (str(name),
str(template)))
            else:
                print('[failure] delete failure, cluster: %-10s, template %-10s ' % (str(name),
str(template)))


if __name__ == '__main__':
    main(sys.argv)
```

c. 执行**vi migrateMapping.py**命令，直接复制输入以下内容无需修改，执行**wq**保存为**索引结构迁移脚本**。


```
# -*- coding:UTF-8 -*-
import sys
import yaml
import requests
import re
import json
import os


def printDividingLine():
    print("<============================================================>")


def loadConfig(argv):
    if argv is None or len(argv) != 2:
        config_yaml = "migrateConfig.yaml"
    else:
        config_yaml = argv[1]
    config_file = open(config_yaml)
    # config = yaml.load(config_file, Loader=yaml.FullLoader)
    return yaml.load(config_file)


def get_cluster_version(url, auth=None):
    response = requests.get(url, auth=auth)
    if response.status_code != 200:
        print("*** get ElasticSearch message failed. resp statusCode:" + str(
            response.status_code) + " response is " + response.text)
        return False
    cluster = response.json()
    version = cluster["version"]["number"]

    return True


def process_mapping(index_mapping, dest_index):
    # remove unnecessary keys
    del index_mapping["settings"]["index"]["provided_name"]
    del index_mapping["settings"]["index"]["uuid"]
    del index_mapping["settings"]["index"]["creation_date"]
    del index_mapping["settings"]["index"]["version"]

    if "lifecycle" in index_mapping["settings"]["index"]:
        del index_mapping["settings"]["index"]["lifecycle"]

    # check alias
    aliases = index_mapping["aliases"]
    for alias in list(aliases.keys()):
        if alias == dest_index:
            print(
                "source index " + dest_index + " alias " + alias + " is the same as dest_index name,
will remove this alias.")
            del index_mapping["aliases"][alias]
    # if index_mapping["settings"]["index"].has_key("lifecycle"):
    if "lifecycle" in index_mapping["settings"]["index"]:
        lifecycle = index_mapping["settings"]["index"]["lifecycle"]
        opendistro = {"opendistro": {"index_state_management":
                                         {"policy_id": lifecycle["name"],
                                          "rollover_alias": lifecycle["rollover_alias"]}}}
        index_mapping["settings"].update(opendistro)
        # index_mapping["settings"]["opendistro"]["index_state_management"]["rollover_alias"] =
lifecycle["rollover_alias"]
        del index_mapping["settings"]["index"]["lifecycle"]

    # replace synonyms_path
    if "analysis" in index_mapping["settings"]["index"]:
        analysis = index_mapping["settings"]["index"]["analysis"]
        if "filter" in analysis:
            filter = analysis["filter"]
```

```
            if "my_synonym_filter" in filter:
                my_synonym_filter = filter["my_synonym_filter"]
                if "synonyms_path" in my_synonym_filter:
                    index_mapping["settings"]["index"]["analysis"]["filter"]["my_synonym_filter"][
                        "synonyms_path"] = "/rds/datastore/elasticsearch/v7.10.2/package/
elasticsearch-7.10.2/plugins/analysis-dynamic-synonym/config/synonyms.txt"
    return index_mapping


def getAlias(source, source_auth):
    # get all indices
    response = requests.get(source + "/_alias", auth=source_auth)
    if response.status_code != 200:
        print("*** get all index failed. resp statusCode:" + str(
            response.status_code) + " response is " + response.text)
        exit()

    all_index = response.json()
    system_index = []
    create_index = []
    for index in list(all_index.keys()):
        if (index.startswith(".")):
            system_index.append(index)
        else:
            create_index.append(index)

    return system_index, create_index


def put_mapping_to_target(url, mapping, cluster, source_index, dest_auth=None):
    headers = {'Content-Type': 'application/json'}
    create_resp = requests.put(url, headers=headers, data=json.dumps(mapping),
auth=dest_auth, verify=False)
    if not os.path.exists("mappingLogs"):
        os.makedirs("mappingLogs")
    if create_resp.status_code != 200:
        print(
            "create index " + url + " failed with response: " + str(create_resp) +
            ", source index is " + str(source_index))
        print(create_resp.text)
        filename = "mappingLogs/" + str(cluster) + "#" + str(source_index)
        with open(filename + ".json", "w") as f:
            json.dump(mapping, f)
        return False
    else:
        return True


def main(argv):
    requests.packages.urllib3.disable_warnings()
    print("begin to migrate index mapping!")
    config = loadConfig(argv)
    src_clusters = config.keys()

    print("begin to process cluster name :")
    for name in src_clusters:
        print(name)
    print("cluster count:" + str(src_clusters.__len__()))

    for name, value in config.items():
        printDividingLine()
        source = value["src_ip"]
        source_user = value["src_username"]
        source_passwd = value["src_password"]
        source_auth = None
        if source_user != "":
            source_auth = (source_user, source_passwd)
        dest = value["dest_ip"]
        dest_user = value["dest_username"]
```

```
        dest_passwd = value["dest_password"]
        dest_auth = None
        if dest_user != "":
            dest_auth = (dest_user, dest_passwd)

        print("start to process cluster:   " + name)
        # only deal with mapping list
        if 'only_mapping' in value and value["only_mapping"]:
            for source_index, dest_index in value["mapping"].iteritems():
                print("start to process source index" + source_index + ", target index: " + dest_index)
                source_url = source + "/" + source_index
                response = requests.get(source_url, auth=source_auth)
                if response.status_code != 200:
                    print("*** get ElasticSearch message failed. resp statusCode:" + str(
                        response.status_code) + " response is " + response.text)
                    continue
                mapping = response.json()
                index_mapping = process_mapping(mapping[source_index], dest_index)
                dest_url = dest + "/" + dest_index
                result = put_mapping_to_target(dest_url, index_mapping, name, source_index,
dest_auth)
                if result is False:
                    print("cluster name:" + name + ", " + source_index + ":failure")
                    continue
                print("cluster name:" + name + ", " + source_index + ":success")
        else:
            # get all indices
            system_index, create_index = getAlias(source, source_auth)
            success_index = 0
            for index in create_index:
                source_url = source + "/" + index
                index_response = requests.get(source_url, auth=source_auth)
                if index_response.status_code != 200:
                    print("*** get ElasticSearch message failed. resp statusCode:" + str(
                        index_response.status_code) + " response is " + index_response.text)
                    continue
                mapping = index_response.json()

                dest_index = index
                if 'mapping' in value:
                    if index in value["mapping"].keys():
                        dest_index = value["mapping"][index]
                index_mapping = process_mapping(mapping[index], dest_index)

                dest_url = dest + "/" + dest_index
                result = put_mapping_to_target(dest_url, index_mapping, name, index, dest_auth)
                if result is False:
                    print("[failure]: migrate mapping cluster name: " + name + ", " + index)
                    continue
                print("[success]: migrate mapping cluster name: " + name + ", " + index)
                success_index = success_index + 1
            print("create index mapping success total: " + str(success_index))


if __name__ == '__main__':
    main(sys.argv)
```

d. 执行**vi checkIndices.py**命令，直接复制输入以下内容无需修改，执行**wq**保存为**索引数据对比脚本**。

```
# -*- coding:UTF-8 -*-
import sys
import yaml
import requests
import re
import json
import os


def printDividingLine():
    print("<============================================================>")
```



```
def get_cluster_version(url, auth=None):
    response = requests.get(url, auth=auth)
    if response.status_code != 200:
        print("*** get ElasticSearch message failed. resp statusCode:" + str(
            response.status_code) + " response is " + response.text)
        return False
    cluster = response.json()
    version = cluster["version"]["number"]
    return True


# get all indices
def get_indices(url, source_auth):
    response = requests.get(url + "/_alias", auth=source_auth)
    if response.status_code != 200:
        print("*** get all index failed. resp statusCode:" + str(
            response.status_code) + " response is " + response.text)
        exit()
    all_index = response.json()
    system_index = []
    create_index = []
    for index in list(all_index.keys()):
        if (index.startswith(".")):
            system_index.append(index)
        else:
            create_index.append(index)
    return create_index


def get_mapping(url, _auth, index):
    source_url = url + "/" + index
    index_response = requests.get(source_url, auth=_auth)
    if index_response.status_code != 200:
        print("*** get ElasticSearch message failed. resp statusCode:" + str(
            index_response.status_code) + " response is " + index_response.text)
        return "[failure] --- index is not exist in destination es. ---"
    mapping = index_response.json()
    return mapping


def get_index_total(url, index, es_auth):
    stats_url = url + "/" + index + "/_stats"
    index_response = requests.get(stats_url, auth=es_auth, verify=False)
    if index_response.status_code != 200:
        print("*** get ElasticSearch stats message failed. resp statusCode:" + str(
            index_response.status_code) + " response is " + index_response.text)
        return 0
    return index_response.json()


def get_indices_stats(url, es_auth):
    endpoint = url + "/_cat/indices"
    indicesResult = requests.get(endpoint, es_auth)
    indicesList = indicesResult.split("\n")
    indexList = []
    for indices in indicesList:
        indexList.append(indices.split()[2])
    return indexList


def loadConfig(argv):
    if argv is None or len(argv) != 2:
        config_yaml = "migrateConfig.yaml"
    else:
        config_yaml = argv[1]
    config_file = open(config_yaml)
    # python3
```

```
    # return yaml.load(config_file, Loader=yaml.FullLoader)
    return yaml.load(config_file)


def main(argv):
    requests.packages.urllib3.disable_warnings()
    print("begin to migrate index mapping!")
    config = loadConfig(argv)
    src_clusters = config.keys()

    print("begin to process cluster name :")
    for name in src_clusters:
        print(name)
    print("cluster count:" + str(src_clusters.__len__()))

    for name, value in config.items():
        printDividingLine()
        source = value["src_ip"]
        source_user = value["src_username"]
        source_passwd = value["src_password"]
        source_auth = None
        if source_user != "":
            source_auth = (source_user, source_passwd)
        dest = value["dest_ip"]
        dest_user = value["dest_username"]
        dest_passwd = value["dest_password"]
        dest_auth = None
        if dest_user != "":
            dest_auth = (dest_user, dest_passwd)
        cluster_name = name
        if "clustername" in value:
            cluster_name = value["clustername"]

        print("start to process cluster ： " + cluster_name)
        # get all indices
        all_source_index = get_indices(source, source_auth)
        all_dest_index = get_indices(dest, dest_auth)

        if "only_compare_index" in value and value["only_compare_index"]:
            print("[success] only compare mapping, not compare index count.")
            continue

        for index in all_source_index:
            index_total = get_index_total(value["src_ip"], index, source_auth)
            src_total = index_total["_all"]["primaries"]["docs"]["count"]
            src_size = int(index_total["_all"]["primaries"]["store"]["size_in_bytes"]) / 1024 / 1024
            dest_index = get_index_total(value["dest_ip"], index, dest_auth)
            if dest_index is 0:
                print('[failure] not found, index: %-20s, source total: %-10s size %6sM'
                      % (str(index), str(src_total), src_size))
                continue
            dest_total = dest_index["_all"]["primaries"]["docs"]["count"]
            if src_total != dest_total:
                print('[failure] not consistent, '
                      'index: %-20s, source total: %-10s size %6sM destination total: %-10s '
                      % (str(index), str(src_total), src_size, str(dest_total)))
                continue
            print('[success] compare index total equal : index : %-20s,  total: %-20s '
                  % (str(index), str(dest_total)))


if __name__ == '__main__':
    main(sys.argv)
```

6. 执行如下命令，迁移Elasticsearch集群的索引模板和索引结构。

```
python migrateTemplate.py
python migrateMapping.py
```


创建 Logstash 集群

当ECS中的迁移环境准备完成后，在CSS服务创建一个Logstash集群用于迁移数据。

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 > Logstash”。
3. 单击右上角的“创建集群”，进入创建集群页面。
4. 在创建集群页面，根据指导完成集群配置。

   关键配置参数请参见**表1-4**，其他参数可以保持默认值。创建集群的详细说明请参见**创建Logstash集群**。

   表 1-4 Logstash 集群的关键配置

参数	说明
计费模式	选择“按需计费”，按实际使用时长计费，计费周期为一小时，不足一小时按一小时计费。
集群类型	选择“Logstash”。
集群版本	选择“7.10.0”。
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。 此处以“Logstash-ES”为例。
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。 此处选择和目标Elasticsearch集群同一个虚拟私有云（VPC）。
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。 此处选择和目标Elasticsearch集群同一个安全组。

5. 单击“下一步：高级配置”，此配置保持默认配置即可。
6. 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。
7. 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”。

验证集群间的网络连通性

在启动迁移任务前，需要先验证Logstash和源Elasticsearch集群的网络连通性。

1. 在Logstash集群列表，选择创建的Logstash集群“Logstash-ES”，单击操作列的“配置中心”，进入配置中心页面。
2. 在配置中心页面，单击“连通性测试”。
3. 在弹窗中输入源集群的IP地址和端口号，单击“测试”。

图 1-4 连通性测试

当显示“可用”时，表示集群间网络连通。如果网络不连通，可以配置Logstash集群路由，连通集群间的网络，具体操作请参见**配置Logstash集群路由**。

使用 Logstash 全量迁移集群数据

在集群迁移初期或需要确保数据完整性的场景，推荐使用Logstash全量迁移集群数据，该方法会一次迁移整个Elasticsearch集群的数据。

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 > Logstash”。
3. 在Logstash集群列表，选择创建的Logstash集群“Logstash-ES”，单击操作列的“配置中心”，进入配置中心页面。
4. 在配置中心页面，单击右上角“创建”，进入创建配置文件页面，编辑Elasticsearch集群的全量迁移配置文件。
   a. 选择集群模板：展开系统模板，选择“elasticsearch”，单击操作列的“应用”。
   b. 设置配置文件名称：在“名称”处自定义配置文件名称，例如“es-es-all”。
   c. 修改配置文件内容：在“配置文件内容”处填写Elasticsearch集群的迁移配置方案，配置文件示例如下。集群信息的获取方式请参见**获取Elasticsearch集群信息**。

```
input{
    elasticsearch{
      # 源Elasticsearch的访问地址，不需要添加协议，添加HTTPS协议会导致报错。
      hosts =>  ["xx.xx.xx.xx:9200", "xx.xx.xx.xx:9200"]
      # 访问源集群的用户名和密码，非安全集群无需配置。
      # user => "css_logstash"
      # password => "*****"
      # 配置待迁移的索引信息，多个索引以逗号隔开，可以使用通配符设置，例如“index*”。
      index => "*_202102"
      docinfo => true
      slices => 3
      size => 3000
      # 当目标集群是HTTPS访问方式时，则需额外配置以下信息。
      # 配置集群HTTPS访问证书，CSS集群保持以下不变。
      # ca_file => "/rds/datastore/logstash/v7.10.0/package/logstash-7.10.0/extend/certs"  # for 7.10.0
      # 是否开启HTTPS通信，HTTPS访问集群则设置为true。
      #ssl => true
    }
}
```

```
# 移除一些logstash增加的字段。
 filter {
   mutate {
     remove_field => ["@version"]
   }
 }

 output{
     elasticsearch{
       # 目标Elasticsearch集群的访问地址
       hosts => ["xx.xx.xx.xx:9200","xx.xx.xx.xx:9200"]
       # 访问目标集群的用户名和密码，非安全集群无需配置。
       # user => "css_logstash"
       # password => "*****"
       # 配置目标集群的索引，以下配置为索引名称和源端保持一致，保持默认。
       index => "%{[@metadata][_index]}"
       document_type => "%{[@metadata][_type]}"
       document_id => "%{[@metadata][_id]}"
       # 当目标集群是HTTPS访问方式时，则需额外配置以下信息。
       # 配置集群HTTPS访问证书，CSS集群保持以下不变。
       #cacert => "/rds/datastore/logstash/v7.10.0/package/logstash-7.10.0/extend/certs"  # for
7.10.0
       # 是否开启HTTPS通信，HTTPS访问集群则设置为true。
       #ssl => true
       # 是否验证服务端Elasticsearch证书，设置为false表示不验证证书。
       #ssl_certificate_verification => false
     }
 }
```

表 1-5 全量迁移配置项说明

配置项名称		说明
input	hosts	源集群的访问地址，如果集群有多个访问节点请分别填写，使用逗号隔开。
	user	访问集群的用户名，如果是非安全集群此项使用“#”注释掉。
	password	访问集群的密码，如果是非安全集群此项使用“#”注释掉。
	index	需要全量迁移的源端索引信息，使用逗号隔开，可以使用通配符设置，例如“index*”。
	docinfo	是否重新索引文档，必须为true。
	slices	配置该参数可以使用切片滚动同时对查询的不同切片，提高整体吞吐量。建议在2-8内。
	size	每次查询返回的最大命中数。
output	hosts	目标集群访问地址，如果集群有多个节点，请分别填写，使用逗号隔开。
	user	访问集群的用户名，如果是非安全集群此项使用“#”注释掉。
	password	访问集群的密码，如果是非安全集群此项使用“#”注释掉。

配置项名称		说明
	index	迁移到目标集群的索引名称，支持扩展修改，如“Logstash-%{+yyyy.MM.dd}”。
	document_type	使目标端文档类型与源端保持一致。
	document_id	索引中的文档ID，建议与源端保持一致，如需要自动生成，使用“#”注释掉即可。

d. 编辑完成后，单击“下一页”配置Logstash配置文件运行参数。

此示例保持默认值即可，如需设置请参见。

e. 配置完成后，单击“创建”。

在配置中心页面可以看到创建的配置文件，状态为“可用”，表示创建成功。

5. 启动全量迁移任务。

a. 在配置文件列表，选择配置文件“es-es-all”，单击左上角的“启动”。

b. 在“启动Logstash服务”对话框中，根据业务需要选择“是否保持常驻”。此示例不开启保持常驻。

开启“保持常驻”以后，将会在每个节点上面配置一个守护进程，当logstash服务出现故障的时候，会主动拉起并修复。“保持常驻”适用于需要长期运行的业，不适用于短期运行的业务，短期业务如果源端无数据，开启保持常驻会导致任务失败。

c. 单击“确定”，开始启动配置文件启动Logstash全量迁移任务。

可以在管道列表看到启动的配置文件。

6. 数据迁移完毕检查数据一致性。

- 方式一：使用Putty登录迁移虚拟机，执行命令**python checkIndices.py**对比数据结果。
- 方式二：分别在源集群和目标集群的Kibana执行命令**GET _cat/indices**，对比两者的索引信息是否一致。

使用 Logstash 增量迁移集群数据

在需要持续同步数据或对数据实时性有较高要求的场景，推荐使用Logstash增量迁移集群数据，该方法通过Logstash配置增量查询，仅支持迁移有增量字段的索引数据。

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 > Logstash”。
3. 在Logstash集群列表，选择创建的Logstash集群“Logstash-ES”，单击操作列的“配置中心”，进入配置中心页面。
4. 在配置中心页面，单击右上角“创建”，进入创建配置文件页面，编辑Elasticsearch集群的增量迁移配置文件。

a. 选择集群模板：展开系统模板，选择“elasticsearch”，单击操作列的“应用”。

b. 设置配置文件名称：在“名称”处自定义配置文件名称，例如“es-es-inc”。

c. 修改配置文件内容：在“配置文件内容”处填写Elasticsearch集群的迁移配置方案，配置文件示例如下。

不同的索引的增量迁移配置不同，必须基于索引分析给出增量配置文件迁移命令。集群信息的获取方式请参见**获取Elasticsearch集群信息**。

```
input{
    elasticsearch{
        # 源Elasticsearch的访问地址，不需要添加协议，添加HTTPS协议会导致报错。
        hosts =>  ["xx.xx.xx.xx:9200"]
        # 访问源集群的用户名和密码，非安全集群无需配置。
        user => "css_logstash"
        password => "******"
        # 配置增量迁移索引。
        index => "*_202102"
        # 配置增量迁移查询语句。
        query => '{"query":{"bool":{"should":[{"range":{"postsDate":{"from":"2021-05-25
00:00:00"}}}]}}}'
        docinfo => true
        size => 1000
        # 当目标集群是HTTPS访问方式时，则需额外配置以下信息。
        # 配置集群HTTPS访问证书，CSS集群保持以下不变。
        # ca_file => "/rds/datastore/logstash/v7.10.0/package/logstash-7.10.0/extend/certs"  # for
7.10.0
        # 是否开启HTTPS通信，HTTPS访问集群则设置为true。
        #ssl => true
    }
}

filter {
  mutate {
    remove_field => ["@timestamp", "@version"]
  }
}

output{
    elasticsearch{
        # 目标集群的访问地址。
        hosts => ["xx.xx.xx.xx:9200","xx.xx.xx.xx:9200"]
        # 访问目标集群的用户名和密码，非安全集群无需配置。
        #user => "admin"
        #password => "******"
        # 配置目标集群的索引，以下配置为索引名称和源端保持一致，保持默认。
        index => "%{[@metadata][_index]}"
        document_type => "%{[@metadata][_type]}"
        document_id => "%{[@metadata][_id]}"
        # 当目标集群是HTTPS访问方式时，则需额外配置以下信息。
        # 配置集群HTTPS访问证书，CSS集群保持默认。
        #cacert => "/rds/datastore/logstash/v7.10.0/package/logstash-7.10.0/extend/certs"  # for
7.10.0
        # 是否开启HTTPS通信，HTTPS访问集群则设置为true。
        #ssl => true
        # 是否验证服务端Elasticsearch证书，设置为false表示不验证证书。
        #ssl_certificate_verification => false
    }

    #stdout { codec => rubydebug { metadata => true }}
}
```

表 1-6 增量迁移配置项说明

配置	说明
hosts	分别填写源集群和目标集群的访问地址，如果集群有多个访问节点请分别填写。

配置	说明
user	访问集群的用户名，如果是非安全集群此项使用“#”注释掉。
password	访问集群的密码，如果是非安全集群此项使用“#”注释掉。
index	需要增加迁移的索引信息，一个配置文件只支持一个索引的增量迁移。
query	增量数据的识别标识，一般是Elasticsearch的DLS语句，需要提前分析。其中postsDate为业务中的时间字段。 {"query":{"bool":{"should":[{"range":{"postsDate":{"from":"2021-05-25 00:00:00"}}}]}}} 此处命令是迁移2021-05-25之后新增加的数据，在多次增量迁移过程中需要修改此处的日志值，如果日期是时间戳方式请转换为时间戳。此处命令需要提前验证有效性。
scroll	当源端数据量过大，为了防止Logstash内存溢出，可以使用scroll分批次获取数据。默认为"1m"。间隔时间不要太长，否则可能会丢失数据。

5. 启动增量迁移任务。
   a. 在配置文件列表，选择配置文件“es-es-inc”，单击左上角的“启动”。
   b. 在“启动Logstash服务”对话框中，根据业务需要选择“是否保持常驻”。此示例不开启保持常驻。
      开启“保持常驻”以后，将会在每个节点上面配置一个守护进程，当logstash服务出现故障的时候，会主动拉起并修复。“保持常驻”适用于需要长期运行的业，不适用于短期运行的业务，短期业务如果源端无数据，开启保持常驻会导致任务失败。
   c. 单击“确定”，开始启动配置文件启动Logstash增量迁移任务。
      可以在管道列表看到启动的配置文件。
6. 数据迁移完毕检查数据一致性。
   - 方式一：使用Putty登录迁移虚拟机，执行命令**python checkIndices.py**对比数据结果。
   - 方式二：分别在源集群和目标集群的Kibana执行命令**GET _cat/indices**，对比两者的索引信息是否一致。

释放 Logstash 集群

当集群迁移完成后，请及时释放Logstash集群，可以节约资源，避免产生不必要的费用。

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 > Logstash”。
3. 在Logstash集群列表，选择创建的Logstash集群“Logstash-ES”，单击操作列的“更多 > 删除”，在弹出的确认提示框中，输入“DELETE”，单击“确定”完成集群删除。

1.3 通过备份与恢复实现华为云 Elasticsearch 集群间数据迁移

CSS服务的Elasticsearch集群之间的数据迁移，可以通过备份与恢复集群快照功能实现。

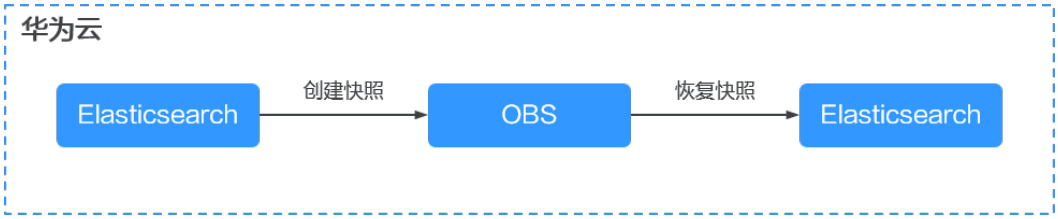
应用场景

通过备份与恢复实现华为云Elasticsearch集群间数据迁移仅适用于源集群和目标集群都是CSS服务的集群，且依赖对象存储服务OBS。常用于以下场景：

- 跨地域或跨账号迁移：将其他Region或账号下的Elasticsearch集群迁移到当前集群中。
- 跨版本迁移：将低版本的Elasticsearch集群数据迁移到高版本的集群中。
- 集群合并：将两个Elasticsearch集群的索引数据合并到一个集群中。

方案架构

图 1-5 迁移流程



通过备份与恢复实现华为云Elasticsearch集群间数据迁移的迁移流程如图1-5所示。

1. 源Elasticsearch集群创建快照，将快照存储在OBS桶中。
2. 目标Elasticsearch集群恢复快照，从OBS桶中恢复快照。

方案优势

- 易于操作和管理：在CSS服务控制台使用集群快照功能实现备份恢复，操作简单，且易于管理和自动化。
- 适用于大规模数据迁移：快照备份适用于数据量大的场景，特别是当数据量达到GB、TB甚至PB级别时。
- 支持跨地域和跨账号迁移：通过结合OBS的跨区域复制功能，可以实现跨地域和跨账号的数据迁移。
- 恢复过程可控：在恢复数据时，可以选择恢复特定索引或全部索引，并且可以指定恢复到特定的集群状态。
- 迁移时长可控：基于迁移时长评估公式可以配置数据迁移速率，理想状态下等于文件复制速率。

性能影响

使用备份与恢复迁移集群的核心在于直接复制数据存储层的文件以实现数据备份，该方案避免了对Elasticsearch外部读写接口的依赖，从而显著降低了对源集群性能的影响。对于对业务延迟要求不严苛的集群而言，此方法对性能的干扰几乎可以忽略。

约束限制

- 目标集群的版本不能低于源集群的版本，版本兼容性分析请参见**Snapshot version compatibility**。
- 目标集群的节点数要大于源集群的节点数的一半，且不能小于源集群的shard副本数。
- 目标集群的CPU、MEM和Disk配置不能低于源集群的配置。

迁移时长

迁移过程的耗时长短依赖于源集群和目标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，恢复阶段耗时由目标集群决定。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÷40MB÷源集群节点个数+800G÷40MB÷目的集群节点个数）x索引个数
- 当索引shard个数小于节点个数时

  总时长(S)=（800G÷40MB÷源集群索引shard个数+800G÷40MB÷目的集群索引shard个数）x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。

前提条件

- 目标集群（Es-2）和源集群（Es-1）处于可用状态。建议在业务空闲期进行集群迁移。
- 确认目标集群（Es-2）和源集群（Es-1）在同一个Region和账号下。

  如果集群跨地域或跨账号，请参考**配置跨区域复制**将源集群存储快照的OBS桶复制到目标集群存储快照的OBS桶中，迁移时在目标集群执行恢复操作。

操作步骤

1. 登录云搜索服务管理控制台。
2. 在“集群管理 > Elasticsearch”页面，单击源端集群名称“Es-1”进入集群基本信息页面。
3. 在左侧导航栏选择“集群快照”，打开集群快照开关，设置快照的基础配置。

表 1-7 集群快照基础配置

参数	说明
OBS桶	选择存储集群快照的OBS桶。
备份路径	集群快照在OBS桶中的存放路径。可以保持默认值。
IAM委托	快照数据存储到OBS桶中，需要OBS的操作权限。选择IAM委托，授权当前账号访问和使用OBS的权限。 ● 当首次配置委托时，可以单击“自动创建委托”新建委托“css-obs-agency”直接使用。 ● 当已有自动创建的委托时，可以单击“委托一键授权”，自动删除委托中OBS Administrator系统策略的权限，并自动新增如下自定义策略授权委托到最小化权限。 "obs:bucket:GetBucketLocation", "obs:object:GetObjectVersion", "obs:object:GetObject", "obs:object:DeleteObject", "obs:bucket:HeadBucket", "obs:bucket:GetBucketStoragePolicy", "obs:object:DeleteObjectVersion", "obs:bucket:ListBucketVersions", "obs:bucket:ListBucket", "obs:object:PutObject" ● 当OBS桶使用SSE-KMS加密模式时，委托中需要包含KMS的权限，“自动创建委托”和“委托一键授权”会自动增加如下自定义策略授权委托到最小化权限。 "kms:cmk:create", "kms:dek:create", "kms:cmk:get", "kms:dek:decrypt", "kms:cmk:list" ● 执行“自动创建委托”和“委托一键授权”的用户需要如下最小权限。 "iam:agencies:listAgencies", "iam:roles:listRoles", "iam:agencies:getAgency", "iam:agencies:createAgency", "iam:permissions:listRolesForAgency", "iam:permissions:grantRoleToAgency", "iam:permissions:listRolesForAgencyOnProject", "iam:permissions:revokeRoleFromAgency", "iam:roles:createRole" ● 使用委托的用户需要如下最小权限。 "iam:agencies:listAgencies", "iam:agencies:getAgency", "iam:permissions:listRolesForAgencyOnProject", "iam:permissions:listRolesForAgency"

4. 完成基础配置后，单击“创建快照”，在弹窗中完成参数配置，单击“确定”启动手动创建快照。

表 1-8 创建快照的配置

参数	说明
快照名称	自定义快照名称，可以保持默认值。

参数	说明
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。 **说明** 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	描述快照信息。

在快照管理列表中，当“快照状态”为“可用”时表示快照创建成功。

5. 快照创建成功后，在快照管理列表，单击快照操作列的“恢复”，配置恢复参数将数据恢复至目标集群“Es-2”。

表 1-9 恢复快照的配置

参数	说明
索引	填写需要进行恢复的索引名称。如果不填写，则表示恢复所有的索引数据。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	索引名称匹配规则。“索引名称匹配模式”和“索引名称替换模式”必须同时设置才会生效。通过配置这两参数，可对快照中匹配到的索引进行重命名。
索引名称替换模式	索引名称重命名规则。设置“索引名称替换模式”参数时，“索引名称匹配模式”参数和该参数必须同时设置才能生效。 默认值“restored_index_$1”表示在所有恢复的索引名称前面加上“restored_”。
集群	选择要恢复快照的目标集群，本案例选择“Es-2”。 选择“是否覆盖目标集群同名索引”，默认不覆盖，即不勾选。通过快照恢复数据是以覆盖快照文件的形式进行数据恢复，当目标集群存在同名索引时，需要勾选覆盖才能恢复同shard结构的索引，不同shard结构的索引不支持恢复。请谨慎勾选操作。

在快照管理列表中，当“任务状态”变更为“恢复成功”时表示集群数据迁移完成。

6. 数据迁移完毕，检查目标Elasticsearch集群“Es-2”和源集群“Es-1”数据的一致性。例如，分别在源集群和目标集群执行**_cat/indices**命令，对比两者的索引信息是否一致。

1.4 通过 S3 插件备份与恢复迁移自建 Elasticsearch 集群至华为云

自建Elasticsearch集群数据迁移到华为云Elasticsearch集群可以使用S3快照备份恢复方式。

应用场景

S3插件（repository-s3）是一个专为Elasticsearch设计的插件，该插件允许用户将Elasticsearch的快照备份存储到与S3 API兼容的存储服务中，例如华为云的对象存储服务(OBS)。S3插件提供了一种高效、灵活且安全的方式来备份Elasticsearch集群的数据。

通过S3插件实现自建Elasticsearch集群和华为云Elasticsearch集群之间的数据迁移，常用于以下场景：

- 服务迁移上云：将自建的Elasticsearch服务迁移到云平台，以利用云服务的可扩展性、维护简便性和成本效益。
- 跨版本迁移：将低版本的自建Elasticsearch集群数据迁移到高版本的华为云Elasticsearch集群中。
- 集群整合：将多个自建Elasticsearch集群迁移到一个华为云Elasticsearch集群中整合为统一的数据平台，以简化管理和提高数据一致性。
- 技术栈统一：当企业已经在华为云上运行其他服务时，为了技术栈的统一和简化管理，可以选择将Elasticsearch集群也迁移至华为云。

方案架构

图 1-6 迁移流程

通过S3插件备份与恢复迁移自建Elasticsearch集群（源集群）至华为云Elasticsearch（目标集群）的迁移流程如图1-6所示。

1. 在自建Elasticsearch集群安装repository-s3插件。
2. 备份自建Elasticsearch数据到华为云对象存储服务OBS上。
3. 从华为OBS恢复数据到CSS服务的Elasticsearch。

方案优势

- 跨版本兼容性高：支持在不同版本的Elasticsearch集群之间进行数据迁移，包括从低版本到高版本的升级。
- 高可用性和持久性：S3提供的高持久性和可用性保证了备份数据的安全性，减少了数据丢失的风险。

- 灵活的备份策略：可以根据业务需求，灵活选择全量备份或增量备份，优化存储使用和迁移时间。

性能影响

使用备份与恢复迁移集群的核心在于直接复制数据存储层的文件以实现数据备份，该方案避免了对Elasticsearch外部读写接口的依赖，从而显著降低了对源集群性能的影响。对于对业务延迟要求不严苛的集群而言，此方法对性能的干扰几乎可以忽略。

约束限制

- 目标集群的版本不能低于源集群的版本，版本兼容性分析请参见**Snapshot version compatibility**。
- 备份与恢复不支持动态增量数据同步，建议停止数据更新后再进行备份。
- 自建Elasticsearch集群需要配置公网访问权限才能备份快照到OBS。

前提条件

- 源Elasticsearch集群和目标Elasticsearch集群处于可用状态。
- 已备好存储快照数据的OBS桶“backup-obs”，OBS桶必须和CSS服务的Elasticsearch集群在同一Region。
- 已获取账号的AK/SK，获取方式请参见**如何获取访问密钥AK/SK**。

操作步骤

步骤1 在自建Elasticsearch集群上安装repository-s3插件。

1. 查看自建Elasticsearch集群的版本，确认要安装的repository-s3插件版本。

   执行如下命令获取集群信息。

   ```
   curl http://x.x.x.x:9200 # 需输入Elasticsearch集群的IP地址。
   ```

   在返回结果中查看“version.number”，如下所示表示集群版本号是7.10.2。

   ```
   {
     "name" : "es_cluster_migrate-ess-esn-1-1",
     "cluster_name" : "es_cluster_migrate",
     "cluster_uuid" : "1VbP7-39QNOx_R-llXKKtA",
     "version" : {
       "number" : "7.10.2",
       "build_flavor" : "default",
       "build_type" : "tar",
       "build_hash" : "d2ef93d",
       "build_date" : "2018-12-17T21:17:40.758843Z",
       "build_snapshot" : false,
       "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"
   }
   ```

2. 下载对应版本的repository-s3插件安装包。此处以7.10.2为例，基于集群实际版本号修改下载地址中的版本号。

   下载地址：“https://artifacts.elastic.co/downloads/elasticsearch-plugins/repository-s3/repository-s3-7.10.2.zip”

3. 登录自建Elasticsearch集群的实例节点上，并cd到plugins目录下。
4. 执行如下命令将repository-s3插件安装包解压到plugins目录下。


```
unzip  repository-s3-7.10.2.zip -d plugins/repository-s3
```

5. 给repository-s3插件添加华为云的访问密钥AK/SK。

   a. 进入自建集群Elasticsearch的安装路径，执行以下命令创建keystore文件。

   ```
   bin/elasticsearch-keystore create s3.keystore
   ```

   说明

   创建keystore文件必须用普通用户，不能用root账号，否则配置不生效。

   b. 将华为云的访问密钥AK/SK增加到keystore文件中。

   ```
   bin/elasticsearch-keystore add s3.client.default.access_key
   bin/elasticsearch-keystore add s3.client.default.secret_key
   ```

   access_key是获取到的访问密钥ID（AK）；secret_key是获取到的秘密访问密钥（SK）。

6. 当自建Elasticsearch集群有多个实例节点时，重复执行**步骤1.3**到**步骤1.5**给其他节点安装S3插件和配置AK/SK。
7. AKSK添加完成，重启Elastiseach集群使配置生效。
8. 重启完成后，执行如下命令查看S3插件是否安装成功。

   ```
   curl http://x.x.x.x:9200/_cat/plugins # 需输入Elasticsearch集群的IP地址。
   ```

   当正常返回repository-s3插件的版本信息时表示安装成功，如下所示。

   ```
   node-1 repository-s3 7.10.2
   ```

步骤2 备份自建Elasticsearch数据到华为云OBS。

1. 登录自建Elasticsearch集群的Kibana，选择“Dev Tools”，进入命令行界面。
2. 执行如下命令，创建快照仓库“my_backup”，对接华为云OBS存放快照数据的OBS桶“backup-obs”。

   ```
   PUT /_snapshot/my_backup
   {
     "type": "s3",
     "settings": {
       # OBS桶名称
       "bucket": "backup-obs",
       # OBS的外网访问地址
       "endpoint": "obs.xxx.example.com",
       "base_path": "snapshot",
       "max_snapshot_bytes_per_sec": "1000mb"
     }
   }
   ```

3. 执行如下命令，当正常返回快照仓库信息时，表示创建成功。

   ```
   GET /_snapshot/my_backup
   ```

4. 执行如下命令，创建快照。

   排除“.”开始的系统索引，为其他索引创建全量快照，其中“wait_for_completion”设置为“true”表示异步执行。

   ```
   PUT  _snapshot/my_backup/snapshot_all?wait_for_completion=true
   {
      "indices": "*,-.*"
   }
   ```

5. 快照创建完成可以执行如下命令查看快照信息。

   ```
   # 查看所有存在的快照
   GET _cat/repositories
   # 查看所有的快照信息
   GET _snapshot/my_backup/_all
   # 根据快照名称查看快照信息，其中snapshot_all是快照名称
   GET _snapshot/my_backup/snapshot_all
   # 根据快照名称查看快照状态，其中snapshot_all是快照名称
   GET _snapshot/my_backup/snapshot_all/_status
   ```


步骤3 从OBS恢复数据到CSS服务的Elasticsearch。

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. 选择目标集群，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. 执行如下命令，创建快照仓库“my_backup_all”。

```
PUT _snapshot/my_backup_all/
{
   "type" : "obs",
   "settings" : {
      # OBS的内网访问域名。
      "endpoint" : "obs.xxx.example.com",
      "region" : "cn-south-1",
      # 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" : "backup-obs",
      "compress" : "false",
      "chunk_size" : "1g",
      "base_path" : "snapshot",
      "max_restore_bytes_per_sec": "1000mb",
      "max_snapshot_bytes_per_sec": "1000mb"
   }
}
```

6. 执行如下命令，恢复快照数据到目标集群。
   - 查看存在的快照，其中my_backup_all是仓库名称。

     ```
     GET _snapshot/my_backup_all/_all
     ```

   - 基于查看到的快照名称恢复数据到目标elasticsearch集群，其中my_backup_all是快照仓库名称，snapshot_all是快照名称。

     ```
     POST _snapshot/my_backup_all/snapshot_all/_restore?wait_for_completion=true
     ```

   - 恢复指定索引到目标Elasticsearch集群，其中my_index1和my_index2是需要恢复的索引名称。

     ```
     POST _snapshot/my_backup_all/snapshot_all/_restore?wait_for_completion=true
     {
       "indices": "my_index1,my_index2,-.*",
       "ignore_unavailable": "true"
     }
     ```

----结束

1.5 通过备份与恢复迁移第三方 Elasticsearch 集群至华为云

通过备份与恢复可以将第三方Elasticsearch集群数据迁移至华为云Elasticsearch集群。

应用场景

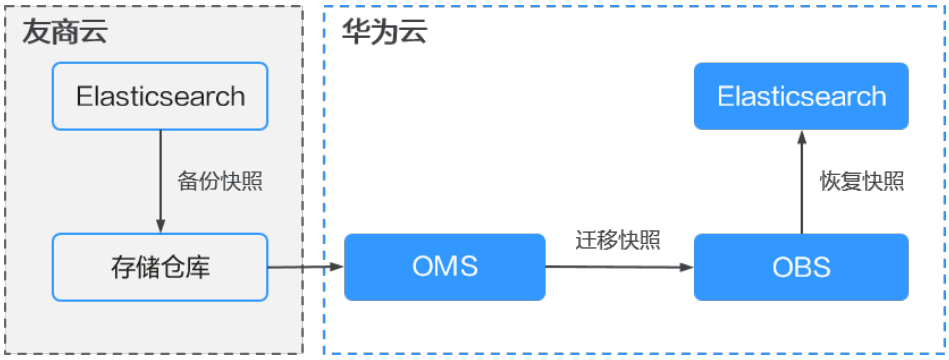
通过备份与恢复实现第三方Elasticsearch集群和华为云Elasticsearch集群之间的数据迁移，依赖存储仓库。常用于以下场景：

- 变更服务提供商：如果企业当前使用的是第三方Elasticsearch服务，但出于成本、性能或其他战略考虑，希望更换服务提供商至华为云。
- 集群整合：将分散在不同第三方Elasticsearch集群中的数据统一数据迁移到华为云Elasticsearch集群进行集中管理，以实现更高效的数据分析和查询。
- 跨版本迁移：将低版本的第三方Elasticsearch集群数据迁移到高版本的华为云Elasticsearch集群中。

- 技术栈统一：当企业已经在华为云上运行其他服务时，为了技术栈的统一和简化管理，可以选择将Elasticsearch集群也迁移至华为云。

方案架构

图 1-7 迁移流程



通过备份与恢复迁移第三方Elasticsearch集群（源集群）至华为云Elasticsearch集群（目标集群）的迁移流程如图1-7所示。

1. 备份第三方Elasticsearch数据到第三方共享存储仓库上。
2. 通过华为云OMS迁移共享存储仓库上的数据到华为云OBS上。
3. 通过华为OBS恢复数据到华为云Elasticsearch。

方案优势

- 易于操作和管理：使用Elasticsearch自带的快照和恢复API，操作简单，易于管理和自动化。
- 适用于大规模数据迁移：快照备份适用于数据量大的场景，特别是当数据量达到GB、TB甚至PB级别时。
- 恢复过程可控：在恢复数据时，可以选择恢复特定索引或全部索引，并且可以指定恢复到特定的集群状态。

性能影响

使用备份与恢复迁移集群的核心在于直接复制数据存储层的文件以实现数据备份，该方案避免了对Elasticsearch外部读写接口的依赖，从而显著降低了对源集群性能的影响。对于对业务延迟要求不严苛的集群而言，此方法对性能的干扰几乎可以忽略。

约束限制

- 目标集群的版本不能低于源集群的版本，版本兼容性分析请参见**Snapshot version compatibility**。
- 备份与恢复不支持动态增量数据同步，建议停止数据更新后再进行备份。
- 第三方存储仓库要配置公网访问才能迁移快照数据。

前提条件

- 源Elasticsearch集群和目标Elasticsearch集群处于可用状态。

- 已备好存储快照数据的OBS桶“esbak”，OBS桶必须和CSS服务的Elasticsearch集群在同一Region。

操作步骤

步骤1 登录Elasticsearch所在的第三方友商云，创建一个支持s3协议的共享存储仓库，例如登录阿里云的进入OSS服务创建目录“patent-esbak”， 或者登录腾讯云进入COS服务创建目录“patent-esbak”。

步骤2 在第三方Elasticsearch集群中创建快照备份仓库，用于存放Elasticsearch快照数据。

例如，在Elasticsearch中创建一个备份仓库“my_backup”，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.example.com",
     # 存储仓库的用户ID和密码。认证用的ak和sk硬编码到代码中或者明文存储都有很大的安全风险，建议在配置文件或者环境变量中密文存放，使用时解密, 确保安全；本示例以ak和sk保存在环境变量中为例，运行本示例前请先在本地环境中设置环境变量ak和sk。
     "access_key_id": "ak",
     "secret_access_key": "sk",
     # 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 在第三方Elasticsearch集群中创建快照。

- 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 在第三方Elasticsearch集群中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 通过华为云对象存储迁移服务OMS将快照数据从存储仓库迁移到对象存储服务OBS的“esbak”桶中。

OMS支持多种云服务商数据迁移到对象存储服务OBS中，具体请参见**各云服务商迁移教程**。

说明

在OMS创建迁移任务时，“元数据迁移方式”一定要选择“保留元数据”，否则数据迁移会有异常。

步骤6 在CSS服务的Elasticsearch集群中创建一个存储仓库关联到OBS，用于恢复第三方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OBS桶“esbak”。

```
PUT _snapshot/my_backup_all/
{
   "type" : "obs",
   "settings" : {
      # OBS的内网访问域名。
      "endpoint" : "obs.xxx.example.com",
      "region" : "xxx",
      # 访问OBS的用户名和密码。认证用的ak和sk硬编码到代码中或者明文存储都有很大的安全风险, 建议在配置文件或者环境变量中密文存放, 使用时解密, 确保安全；本示例以ak和sk保存在环境变量中为例, 运行本示例前请先在本地环境中设置环境变量ak和sk。
      "access_key": "ak",
      "secret_key": "sk",
      # OBS的桶名称，和上一步迁移目标集群的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集群中通过快照恢复数据。

1. 查看所有快照信息。
   ```
   GET _snapshot
   ```
2. 通过快照恢复数据。
   - 恢复某一快照中的所有索引。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
     ```
   - 恢复某一快照中的部分索引。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
  ```

----结束

1.6 通过读写分离插件实现华为云 Elasticsearch 集群间数据迁移

通过CSS服务的读写分离插件可以实现华为云Elasticsearch集群间的数据迁移。

应用场景

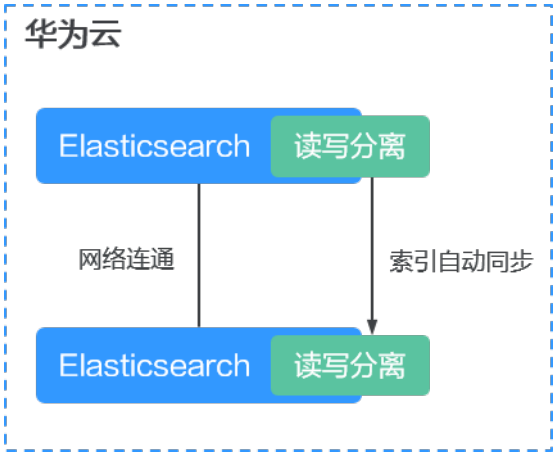
7.6.2和7.10.2版本的华为云Elasticsearch集群默认安装了CSS服务的读写分离插件，通过配置读写分离可以实现Elasticsearch集群间的索引数据近实时同步。

通过读写分离插件实现华为云Elasticsearch集群间数据迁移仅适用于源集群和目标集群都是CSS服务的集群。常用于以下场景：

- 跨地域或跨账号迁移：将其他Region或账号下的Elasticsearch集群迁移到当前集群中。
- 集群合并：将两个Elasticsearch集群的索引数据合并到一个集群中。

方案架构

图 1-8 迁移流程



通过CSS服务的读写分离插件迁移华为云Elasticsearch集群（源集群）至华为云Elasticsearch（目标集群）的数据迁移流程如**图1-8**所示。

1. 通过读写分离插件建立源集群和目标集群的连接。
2. 在目标集群配置索引自动同步，实现源集群的数据自动同步到目标集群。同步周期默认是30秒，支持修改。
3. 查询自动同步状态，确认集群数据是否迁移完成。

更多关于CSS服务的读写分离功能的介绍请参见**配置Elasticsearch集群读写分离**。

方案优势

- 数据一致性高：利用读写分离的主从复制机制，确保数据在不同分片之间的同步，提高数据的一致性 。
- 迁移速度快：读写分离的自动同步速度依赖带宽，不受源集群和目标集群的影响，同步快。
- 实时性可控：读写分离的默认同步周期为30秒，支持修改，可以控制数据迁移的实时性，减少数据同步过程中的延迟 。

性能影响

使用读写分离迁移集群的核心在于通过底层文件复制来实现数据同步，此方法绕开了Elasticsearch的外部读写API，因此对源集群的性能影响被降至最低。

约束限制

源集群和目标集群的版本必须一致，都是7.6.2或7.10.2版本。

前提条件

- 源Elasticsearch集群和目标Elasticsearch集群处于可用状态，且都安装了读写分离插件（ Elasticsearch 7.6.2和7.10.2版本默认安装了读写分离插件 ）。
- 集群间需要保证网络连通。

  如果源集群和目标集群在不同VPC，则需要先打通VPC网络建立对等连接。具体操作请参见**对等连接简介**。

操作步骤

步骤1 获取Elasticsearch集群信息，用于配置迁移任务。

表 1-10 需要获取的 Elasticsearch 集群信息

集群来源		要获取的信息	获取方式
源集群	华为云Elasticsearch集群	• 源集群的访问地址 • 访问源集群的用户名和密码（仅安全集群涉及）	• 获取集群的访问地址请参见**步骤1.3**。 • 用户名和密码请联系服务管理员获取。
	自建Elasticsearch集群	• 源集群的公网访问地址 • 访问源集群的用户名和密码（仅安全集群涉及）	联系服务管理员获取。
	第三方Elasticsearch集群	• 源集群的访问地址 • 访问源集群的用户名和密码（仅安全集群涉及）	联系服务管理员获取。

集群来源		要获取的信息	获取方式
目标集群	华为云Elasticsearch集群	• 目标集群的访问地址 • 访问目标集群的用户名和密码（仅安全集群涉及）	• 获取访问地址请参见步骤1.3。 • 用户名和密码请联系服务管理员获取。

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”。
3. 在Elasticsearch集群列表，获取集群的访问地址。

图 1-9 获取集群信息

步骤2 登录目标Elasticsearch集群的Kibana操作页面。

1. 在CSS服务的Elasticsearch集群列表，选择目标集群，单击操作列的“Kibana”，登录Kibana。
2. 单击左侧导航栏的“Dev Tools”进入操作页面。

步骤3 通过读写分离插件，在目标集群建立与源集群的连接。

执行如下命令，在目标集群配置源集群的信息。

```
PUT /_cluster/settings
{
  "persistent" : {
    "cluster" : {
      "remote.rest" : {
        "leader1" : {
          "seeds" : [
            "http://10.0.0.1:9200",
            "http://10.0.0.2:9200",
            "http://10.0.0.3:9200"
          ] ,
            "username": "elastic",
            "password": "*****"
        }
      }
    }
  }
}
```

表 1-11 请求体参数说明

参数名	说明
leader1	配置任务的名称，由用户自定义，在后续读写分离配置时会用到该名称。
seeds	源集群的访问地址。当集群开启HTTPS访问时，URI schema需要对应使用https。

参数名	说明
username	源集群的用户名，仅主集群是安全集群时才需要配置。
password	源集群的密码，仅主集群是安全集群时才需要配置。

当返回结果中，“acknowledged”为“true”时，表示配置成功。

步骤4 在目标集群配置索引自动同步，实现源集群的数据自动同步到目标集群。

执行如下命令，创建匹配模式同步索引，将源集群中匹配的索引同步到目标集群中。

```
PUT auto_sync/pattern/pattern1
{
 "remote_cluster": "leader1",
 "remote_index_patterns": "log*",
 "local_index_pattern": "{{remote_index}}",
 "apply_exist_index": true
}
```

表 1-12 请求体参数说明

参数名	说明
pattern1	自定义同步集群配置的pattern名字，用于区分多个不同的匹配模式。
remote_cluster	配置任务的名称，例如上一步的“leader1”。
remote_index_patterns	源集群待同步索引模式，支持通配符“*”。
local_index_pattern	同步到目标集群的索引模式，支持模板替换。例如取值为“{{remote_index}}”时，待同步索引为“log1”，同步过来的索引为“log1”。
apply_exist_index	是否同步主集群上已存在的索引，默认为“true”。

步骤5 在目标集群查询集群的自动同步状态，同步周期默认是30秒。

执行如下命令，获取匹配索引的同步状态。

```
GET auto_sync/stats
```

当返回结果中，“failed_count”为“0”时，表示同步完成。

步骤6 集群同步完成后，检查目标Elasticsearch集群和源集群数据的一致性。

例如，分别在源集群和目标集群登录Kibana，进入Dev Tool，执行**GET cat/indices**命令，对比两者的索引信息是否一致。

----结束

1.7 通过 Reindex API 实现 Elasticsearch 集群间数据迁移

使用Reindex API可以实现Elasticsearch集群间的数据迁移。

应用场景

Elasticsearch作为一款开源搜索引擎，提供了Reindex API以支持集群间的索引数据迁移。CSS服务同样支持利用Reindex API实现Elasticsearch集群间的数据迁移。以下是该方案适用的专业迁移场景概述：

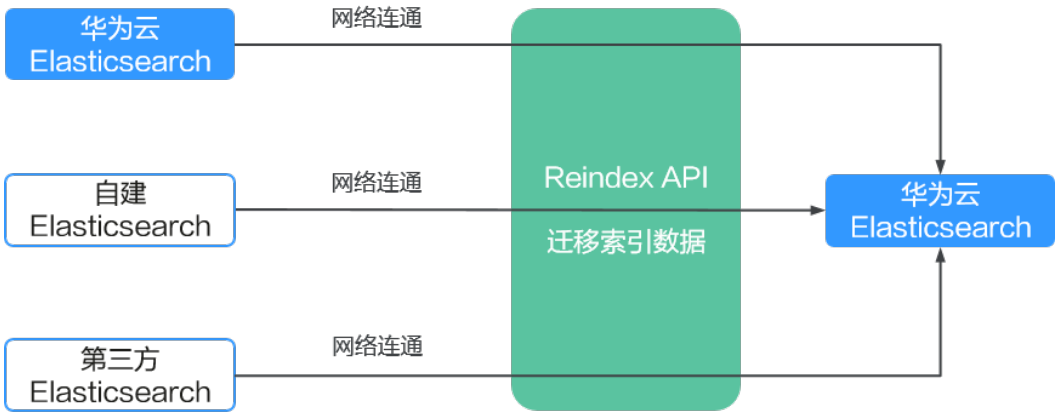
- 集群合并：Reindex API能够将分散在多个Elasticsearch集群中的索引数据合并至单一集群，实现数据的集中管理和分析。
- 服务迁移上云：企业可以将自建的Elasticsearch服务迁移至云平台，以利用云服务的弹性扩展、简化维护和成本效益。
- 变更服务提供商：对于当前使用第三方Elasticsearch服务的企业，若出于成本、性能或其他战略考量，希望更换至华为云等其他服务提供商，Reindex API提供了一种迁移途径。

Reindex API支持以下迁移策略：

- 全量迁移：实现跨集群的完整索引数据迁移。在此过程中，必须暂停对源集群的写入操作，以确保源集群与目标集群数据的一致性。
- 增量迁移：当集群索引包含时间字段时，Reindex API支持基于时间戳的增量数据迁移。在业务切换阶段，全量迁移完成后，需暂停源集群的写入操作，并利用Reindex API基于最近更新时间快速同步增量数据，随后将业务全面切换至目标集群。
- 索引重构：当需要调整索引结构，如Mapping、分词器或分片设置时，Reindex API可用于在迁移过程中重新配置索引结构。

方案架构

图 1-10 迁移流程



通过Reindex API实现Elasticsearch集群间数据迁移的迁移流程如图1-10所示。

1. 在目标集群配置Reindex白名单，连接源集群和目标集群。
2. 通过Reindex API，将源集群的索引数据迁移至目标集群。

方案优势

- 操作简单：Reindex API作为Elasticsearch的内置功能，提供了一个简单直接的方法来迁移数据，无需复杂的配置或额外的工具。

- 数据处理灵活：在迁移过程中，可以对数据进行处理，例如调整Mapping、分词、Shard等。
- 性能控制：在迁移过程中，可以通过配置Scroll API的参数来控制迁移速率，优化迁移性能。

性能影响

Reindex API的工作原理基于Scroll API，此API能够高效地从源集群查找索引数据，并批量同步至目标集群。这一过程可能会对源集群性能产生影响，具体影响程度取决于目标集群对源集群的读取速度，而读取速度取决于Scroll API的size和slice参数配置。参数配置的详细指导可参考**Reindex API**文档。

由于Reindex任务在Elasticsearch集群中作为异步任务执行，在低并发情况下对源集群性能的影响是可管理的。然而，对于资源占用较高的集群，建议通过调整Scroll API的size参数来减缓迁移速度，或选择在业务低峰期进行迁移，以降低对业务的影响。

约束限制

- 集群迁移过程中，源集群的索引数据不能增删改，否则会导致迁移后的源集群数据和目标集群数据内容不一致。
- 源集群和目标集群的版本必须一致。

前提条件

- 源Elasticsearch集群和目标Elasticsearch集群处于可用状态。
- 集群间需要保证网络连通。
  - 如果源集群和目标集群在不同VPC，则需要先打通VPC网络建立对等连接。具体操作请参见**对等连接简介**。
  - 如果是自建Elasticsearch集群迁移至华为云，则可以通过给自建Elasticsearch集群配置公网访问打通网络。
  - 如果是第三方Elasticsearch集群迁移至华为云，则需要建立企业内部数据中心到华为云的VPN通道或专线。
- 确认集群的索引已开启“_source”。

  集群索引的“_source”默认是开启的。执行命令**GET {index}/_search**，当返回的索引信息里有“_source”信息时表示已开启。

获取 Elasticsearch 源集群信息

在迁移集群前，需要先获取必备的源集群信息，用于配置迁移任务。

表 1-13 需要获取的 Elasticsearch 源集群信息

源集群来源	要获取的信息	获取方式
华为云Elasticsearch集群	• 源集群的访问地址 • 访问源集群的用户名和密码（仅安全集群涉及） • 索引结构	• 获取集群名称和访问地址请参见3。 • 用户名和密码请联系服务管理员获取。 • 查询索引结构请参见6。

源集群来源	要获取的信息	获取方式
自建Elasticsearch集群	• 源集群的公网访问地址 • 访问源集群的用户名和密码（仅安全集群涉及） • 索引结构	联系服务管理员获取。
第三方Elasticsearch集群	• 源集群的访问地址 • 访问源集群的用户名和密码（仅安全集群涉及） • 索引结构	联系服务管理员获取。

源集群的来源不同，获取信息的方式不同，此处仅介绍如何获取华为云Elasticsearch集群的信息。

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”。
3. 在Elasticsearch集群列表，获取集群名称和访问地址。

图 1-11 获取集群信息

4. 单击集群操作列的“Kibana”，登录Kibana。
5. 单击左侧导航栏的“Dev Tools”进入操作页面。
6. 执行如下命令，查询源集群的索引结构。

```
GET {index_name}
```

index_name表示待迁移的索引名称。

配置 Reindex 白名单

在Elasticsearch目标集群，配置Reindex白名单。

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”。
3. 在Elasticsearch集群列表，单击目标集群，进入集群基本信息页面。
4. 在左侧导航栏选择“参数配置”，单击“编辑”，添加自定义参数配置Reindex白名单。
   - 参数名称：reindex.remote.whitelist
   - 参数值：填写源集群的访问地址，获取方式请参见**获取Elasticsearch源集群信息**。

   当源Elasticsearch集群是HTTPS访问方式，则还需要添加自定义参数忽略SSL认证。
   - 参数名称：reindex.ssl.verification_mode
   - 参数值：none

5. 单击“提交”，在弹窗中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”。
6. 返回Elasticsearch集群列表，单击目标集群操作列的“更多 > 重启”重启集群，使修改的配置生效。

通过 Reindex API 迁移集群数据

1. 在目标集群，创建索引结构。
   a. 登录云搜索服务管理控制台。
   b. 在左侧菜单栏选择“集群管理 > Elasticsearch”。
   c. 在Elasticsearch集群列表，单击集群操作列的“Kibana”，登录Kibana。
   d. 单击左侧导航栏的“Dev Tools”进入操作页面。
   e. 执行命令，创建和源集群一样的索引结构。

```
PUT {index_name}
{
 源集群的索引结构
}
```

   index_name表示迁移后的索引名称。源集群的索引结构可参见**获取Elasticsearch源集群信息**获取。
2. 执行如下命令，通过Reindex API迁移集群数据。
   - **全量迁移**：源集群的索引数据全量迁移到目标集群。

     在目标集群的Kibana，执行如下命令：

```
POST _reindex?wait_for_completion=false
{
 "source": {
    "remote": {
     "host": "http://xx.xx.xx.xx:9200",    //源集群的访问地址，当源集群是HTTPS访问方式，则使用“https://xx.xx.xx.xx:9200”。
     "username": "xxx",    //访问源集群的用户名，仅安全集群涉及。
     "password": "******"    //访问源集群的用户密码，仅安全集群涉及。
   },
   "index": "index_name",    //源集群的索引名称。
   "size": 3000
 },
 "dest": {
   "index": "index_name"    //目标集群的索引名称。
 }
}
```

   - **增量迁移**：通过时间字段，将源集群的索引数据增量迁移到目标集群。该方案也可以用于进行大数据量索引的分段迁移。

     在目标集群的Kibana，执行如下命令：

```
POST _reindex?wait_for_completion=false
{
 "source": {
   "remote": {
     "host": "http://xx.xx.xx.xx:9200",    //源集群的访问地址，当源集群是HTTPS访问方式，则使用“https://xx.xx.xx.xx:9200”。
     "username": "xxx",    //访问源集群的用户名，仅安全集群涉及。
     "password": "******"    //访问源集群的用户密码，仅安全集群涉及。
   },
   "query": {
     "range" : {
       "timestamps" : {    //增量时间字段。
         "gte" : "xxx",    //增量迁移的开始时间。
         "lte" : "xxx"    //增量迁移的结束时间。
       }
     }
   },
```



```
  "index": "index_name",    //源集群的索引名称。
  "size": 3000
 },
 "dest": {
  "index": "index_name"    //目标集群的索引名称。
 }
}
```

- **同集群索引重构：**Reindex API在迁移过程中可以重新配置索引结构。

  在目标集群的Kibana，执行如下命令：

```
POST _reindex?wait_for_completion=false
{
 "source": {
  "index": "index_name",    //源集群的索引名称
  "size": 3000
 },
 "dest": {
  "index": "index_name"    //目标集群的索引名称
 }
}
```

常见问题：当索引数据大、数据同步慢时，如何处理?

当索引数据大、数据同步慢时，有如下几种方案可以提高效率。

- Reindex API是使用Scroll方式读取源端数据写入目标端，因此可以通过配置size和slice参数增加迁移并发和速度。具体使用指导请参见**Reindex API**。
- 当单个索引数据量比较大时，可以在迁移前将目标索引的副本数设置为0，以加快数据同步速度。待数据迁移完成后，再修改为实际值。
- 当源端数据量很大时，建议采用快照方式迁移数据。例如**通过备份与恢复实现华为云Elasticsearch集群间数据迁移**、**通过S3插件备份与恢复迁移自建Elasticsearch集群至华为云**和**通过备份与恢复迁移第三方Elasticsearch集群至华为云**。
- 当索引数据存在时间字段时，可以使用跨集群增量迁移方案，分段迁移索引数据。

1.8 通过 ESM 实现 Elasticsearch 集群间数据迁移

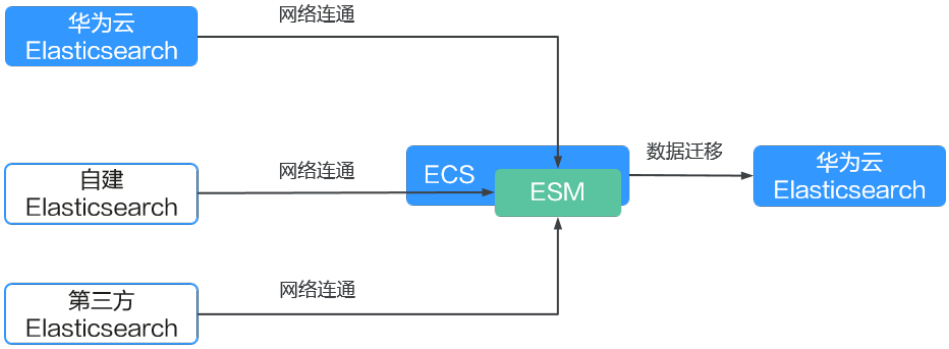
应用场景

ESM（Elasticsearch Migration Tool）是一个开源的Elasticsearch集群迁移工具。它支持不同版本的Elasticsearch之间的数据迁移，并且可以通过配置Scroll API的参数来控制迁移速率，以适应不同的网络环境和业务需求。以下是ESM实现Elasticsearch集群间数据迁移的一些应用场景：

- 跨版本迁移：在Elasticsearch集群需要升级到新版本时，ESM可以帮助平滑迁移数据，确保升级过程中数据的完整性和可用性。
- 集群合并：当企业需要将多个Elasticsearch集群的数据整合到一个集群中以简化管理时，ESM可以高效地完成这一任务。
- 服务迁移上云：企业可以将自建的Elasticsearch服务迁移至云平台，以利用云服务的弹性扩展、简化维护和成本效益。
- 变更服务提供商：如果企业当前使用的是第三方Elasticsearch服务，但出于成本、性能或其他战略考虑，希望更换服务提供商至华为云。

方案架构

图 1-12 迁移流程



通过Reindex API实现Elasticsearch集群间数据迁移的迁移流程如图1-12所示。

1. 在Linux虚拟机上安装ESM迁移工具。
2. 通过ESM迁移命令，将源集群的索引数据迁移至目标集群。

方案优势

- 跨版本兼容性：ESM支持不同版本的Elasticsearch集群间的数据迁移，包括从老版本迁移到新版本。
- 简化操作：ESM使用简单方便，使用Go语言开发，只需下载编译包安装即可使用。
- 性能控制：在迁移过程中，可以通过配置Scroll API的参数来控制迁移速率，优化迁移性能。
- 灵活的迁移方案：ESM提供了灵活的迁移方案，包括全量迁移和增量迁移，以适应不同的业务需求。
- 开源免费：作为一个开源工具，ESM代码托管在GitHub上，用户可以免费使用。

性能影响

ESM迁移集群的工作原理基于Scroll API，此API能够高效地从源集群查找索引数据，并批量同步至目标集群。这一过程可能会对源集群性能产生影响，具体影响程度取决于目标集群对源集群的读取速度，而读取速度取决于Scroll API的size和slice参数配置。参数配置的详细指导可参考**Reindex API**文档。

由于ESM能够迅速从源集群读取数据，可能会对源集群的性能产生影响。因此，建议在业务低峰时段进行数据迁移，以监控源集群CPU和内存的性能指标变化。通过调整迁移速度和选择适宜的迁移时间窗口，可以有效控制性能影响。对于涉及大量数据迁移或资源占用较高的集群，特别推荐在业务低峰期执行数据迁移，以确保源端业务操作不受影响。

约束限制

集群迁移过程中，源集群的索引数据不能增删改，否则会导致迁移失败，或是迁移后的源集群数据和目标集群数据内容不一致。

前提条件

- 源Elasticsearch集群和目标Elasticsearch集群处于可用状态。
- 集群间需要保证网络连通。
  - 如果源集群和目标集群在不同VPC，则需要先打通VPC网络建立对等连接。具体操作请参见**对等连接简介**。
  - 如果是自建Elasticsearch集群迁移至华为云，则可以通过给自建Elasticsearch集群配置公网访问打通网络。
  - 如果是第三方Elasticsearch集群迁移至华为云，则需要建立企业内部数据中心到华为云的VPN通道或专线。
- 确认集群的索引已开启“_source”。

  集群索引的“_source”默认是开启的。执行命令**GET {index}/_search**，当返回的索引信息里有“_source”信息时表示已开启。

获取 Elasticsearch 集群信息

在迁移集群前，需要先获取必备的集群信息，用于配置迁移任务。

表 1-14 需要获取的 Elasticsearch 集群信息

集群来源		要获取的信息	获取方式
源集群	华为云Elasticsearch集群	• 源集群的访问地址 • 访问源集群的用户名和密码（仅安全集群涉及）	• 获取集群名称和访问地址请参见3。 • 用户名和密码请联系服务管理员获取。
	自建Elasticsearch集群	• 源集群的公网访问地址 • 访问源集群的用户名和密码（仅安全集群涉及）	联系服务管理员获取。
	第三方Elasticsearch集群	• 源集群的访问地址 • 访问源集群的用户名和密码（仅安全集群涉及）	联系服务管理员获取。
目标集群	华为云Elasticsearch集群	• 目标集群的访问地址 • 访问目标集群的用户名和密码（仅安全集群涉及）	• 获取访问地址请参见3。 • 用户名和密码请联系服务管理员获取。

源集群的来源不同，获取信息的方式不同，此处仅介绍如何获取华为云Elasticsearch集群的信息。

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”。
3. 在Elasticsearch集群列表，获取集群名称和访问地址。

图 1-13 获取集群信息

准备迁移虚拟机

创建ECS用于迁移Elasticsearch集群。

1. 购买Linux ECS，“镜像”选择CentOS 7系列，“虚拟私有云”和目标集群保持一致。购买操作指导请参见**快速购买和使用Linux ECS**。
2. 测试ECS和源集群、目标集群的连通性。

   在ECS执行如下命令测试连通性，当正常返回集群信息时表示已经连通。

   ```
   # 非安全集群
   curl -ik http://ip:9200
   #安全集群+HTTPS访问
   curl -ik https://ip:9200 -u[Username]:[password]
   ```

通过 ESM 迁移集群

1. 访问**ESM下载地址**，下载“migrator-linux-amd64”软件包。
2. 通过SCP工具将下载的“migrator-linux-amd64”软件包上传到Linux ECS的执行路径下。
3. 在Linux ECS的执行路径下执行命令，将源集群的索引结构和数据迁移到目标集群。

   ```
   # 索引全量迁移
   ./migrator-linux-amd64 -s http://source:9200 -d http://dest:9200 -x index_name -m admin:password -n admin:password --copy_settings --copy_mappings -w 5 -b 10

   #索引增量迁移
   ./migrator-linux-amd6 -s http://source:9200 -d http://dest:9200 -x index-test -m admin:password -n admin:password -w 5 -b 10 -q "timestamp:[\"2022-01-17 03:41:20\" TO \"2022-01-22 03:41:20\"]"
   ```

   迁移命令的常用参数说明请参见**表1-15**，更多详细参数说明请参见**ESM文档**。

表 1-15 常用参数说明

选型	示例	说明
-s, --source=	http://source:9200	源Elasticsearch集群访问地址。
-d, --dest=	http://dest:9200	目标Elasticsearch集群访问地址。
-x, --src_indexes=	index_name index1,index2	源集群迁移索引名，支持正则匹配和逗号分隔。
-y, --dest_index=	index_name_rename	目标集群索引名，支持单索引名称，如果不配置则和源索引名称相同。
-m, --source_auth=	admin:password	访问源Elasticsearch集群的用户名和密码，仅安全集群涉及。
-n, --dest_auth=	admin:password	访问目标Elasticsearch集群的用户名和密码，仅安全集群涉及。

选型	示例	说明
-w, --workers=	5	读取源端数据速率控制参数，Bulk读取数据并发线程数。 默认值：1
-b, --bulk_size=	10	读取源端数据速率控制参数，Bulk一次读取数据的数据大小。 默认值：5MB
--sliced_scroll_size	4	读取源端数据速率控制参考，Bulk Scorll中sliced的大小。 默认值：1
--copy_settings	-	迁移源端索引的settings。
--copy_mappings	-	迁移源端索引的mappings。
--buffer_count=	-	迁移虚拟机缓存在内存中的文档数量。 默认值：10w

4. 数据迁移完成后，通过对比文档个数检查数据一致性。

```
# 非安全集群
curl -ik http://ip:9200/{索引名称}/_count
#安全集群+HTTPS访问
curl -ik https://ip:9200 -u[Username]:[password]/{索引名称}/_count
```

常见问题

- **迁移过程出现报错“out of memory”怎么办?**

  迁移过程中出现报错“out of memory”，表示迁移虚拟机内存溢出，有如下解决方案：

  - 可能是ECS虚拟机的配置不够，建议升级ECS规格，操作指导请参见**变更单台ECS规格**。
  - 调整ESM的迁移速度，通过在迁移命令中减小“buffer_count”参数，限制内存中的文档数量。

- **迁移完成后，为什么源集群和目标集群的索引数据存储量不一致?**

  ESM迁移集群后，两个集群的数据存储量不一致是正常的，这是由Elasticsearch的内部存储机制决定的。Elasticsearch存储是有多个shard，每个shard又有多个segment，由于ESM迁移是把源集群数据写入到目标集群，会导致在目标集群重新生成segment和shard，由于segment和shard都会对数据有膨胀，不同集群的segment的shard不同，数据量也不同，如果需要比较数据一致性，则需要对比文档个数的多少，而不是对比数据大小。

1.9 迁移 Elasticsearch 的 Kibana 可视化图表

应用场景

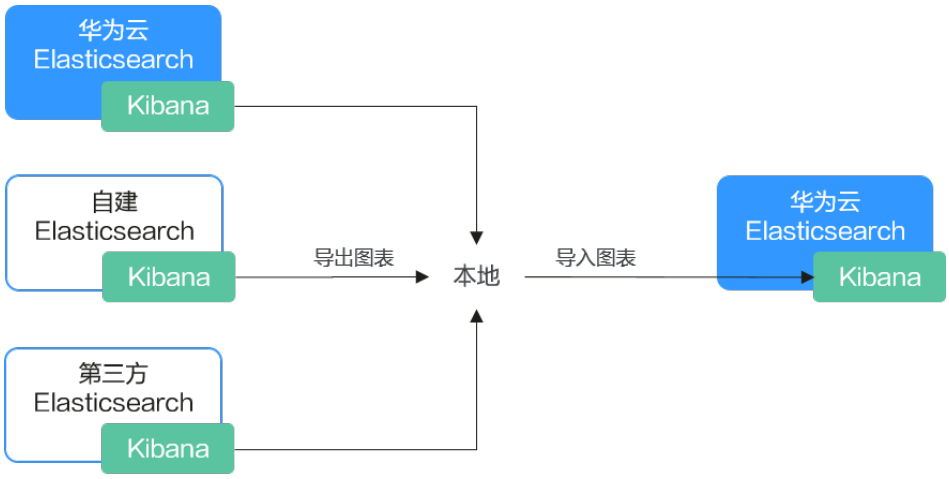
迁移Elasticsearch集群间的Kibana可视化图表的应用场景主要包括以下几个方面：

- 当需要将数据从一个Elasticsearch集群迁移到另一个新集群时，Kibana可视化图表的迁移是确保业务连续性的关键步骤。通过导出源集群的Kibana可视化配置（如仪表板、图表等），并在目标集群中导入，可以确保用户界面和监控仪表板的一致性。
- 在开发、测试和生产环境之间复制Elasticsearch环境时，Kibana可视化图表的迁移可以确保不同环境之间的一致性，提升开发和测试效率。
- 在业务发生故障或数据丢失的情况下，通过迁移Kibana图表到备份集群，可以快速恢复数据监控和分析能力。
- 在多集群环境中，可能需要将不同Elasticsearch集群的数据和可视化图表整合到一个统一的集群，以便于实现跨集群的数据分析和管理。

这些场景展示了Kibana可视化图表迁移在实际应用中的重要性，它不仅涉及到数据的移动，还包括了确保业务连续性、提高效率和满足合规性要求等多个方面。

方案架构

图 1-14 迁移 Elasticsearch 的 Kibana 可视化图表



迁移Elasticsearch的Kibana可视化图表的方案流程如图1-14所示。

- 通过Kibana的图表导出功能，在源Elasticsearch集群的Kibana实例中导出所需的可视化对象。
- 通过Kibana的图表导入功能，在目标Elasticsearch集群的Kibana实例中导入这些对象。

方案优势

- 业务连续性：通过迁移Kibana可视化图表，可以确保在集群升级或迁移过程中，集群的监控和分析工作不会中断，从而保障业务的连续性。
- 环境一致性：在多环境（开发、测试、生产）之间复制Kibana配置，可以确保不同环境的一致性，这有助于减少环境差异导致的问题，提高开发和测试的效率。
- 快速恢复：在故障恢复场景中，Kibana图表的迁移能力可以快速恢复关键的监控和分析功能，减少系统故障对业务的影响。
- 数据整合能力：在多集群环境中，迁移Kibana图表有助于实现数据的集中管理和分析，提高数据整合的效率和效果。

约束限制

源Elasticsearch集群和目标Elasticsearch集群的版本要相近，否则可能迁移异常。如果迁移过程中出现图表不兼容的报错，可以参考**常见问题：迁移过程中，发现图表不兼容怎么办？**处理。

前提条件

源Elasticsearch集群和目标Elasticsearch集群处于可用状态。

操作步骤

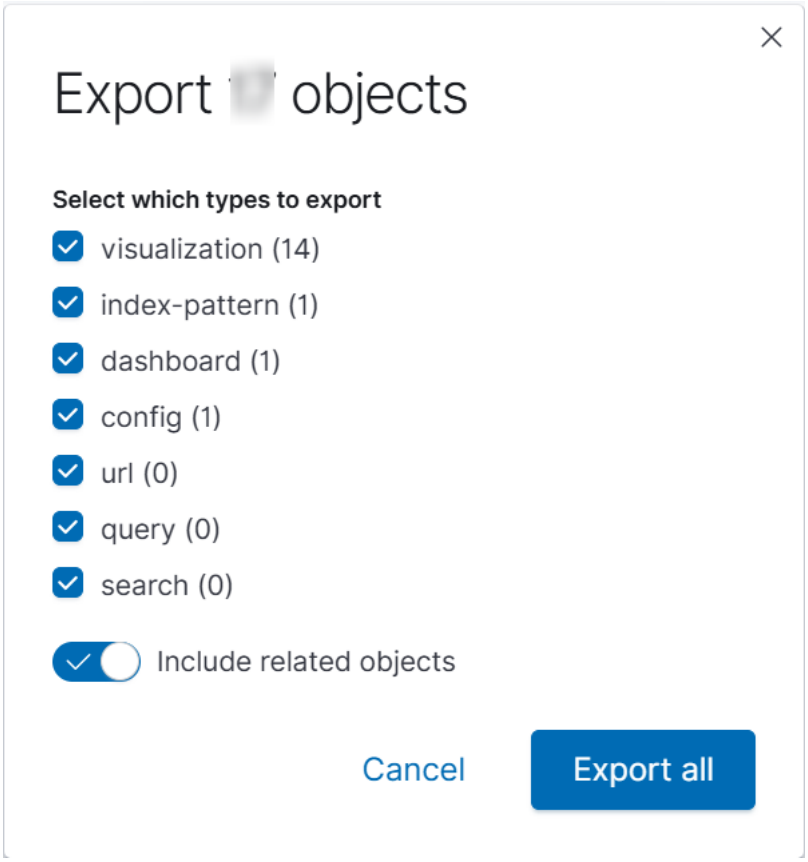
说明

不同版本的Kibana操作界面会有差异，请以实际界面为准。本文仅以7.10.2版本为例。

步骤1 通过Kibana的图表导出功能将源Elasticsearch集群的可视化对象导出到本地。以源集群是华为云Elasticsearch为例，介绍操作步骤。

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”。
3. 在Elasticsearch集群列表，单击集群操作列的“Kibana”，登录Kibana。
4. 在左侧导航栏选择“Stack Management > Saved Objects”。
5. 在Saved Objects页面，单击“Export xx objects”，在弹窗中选择对应的可视化图表，单击“Export all”，将可视化对象文件“export.ndjson”下载到本地。

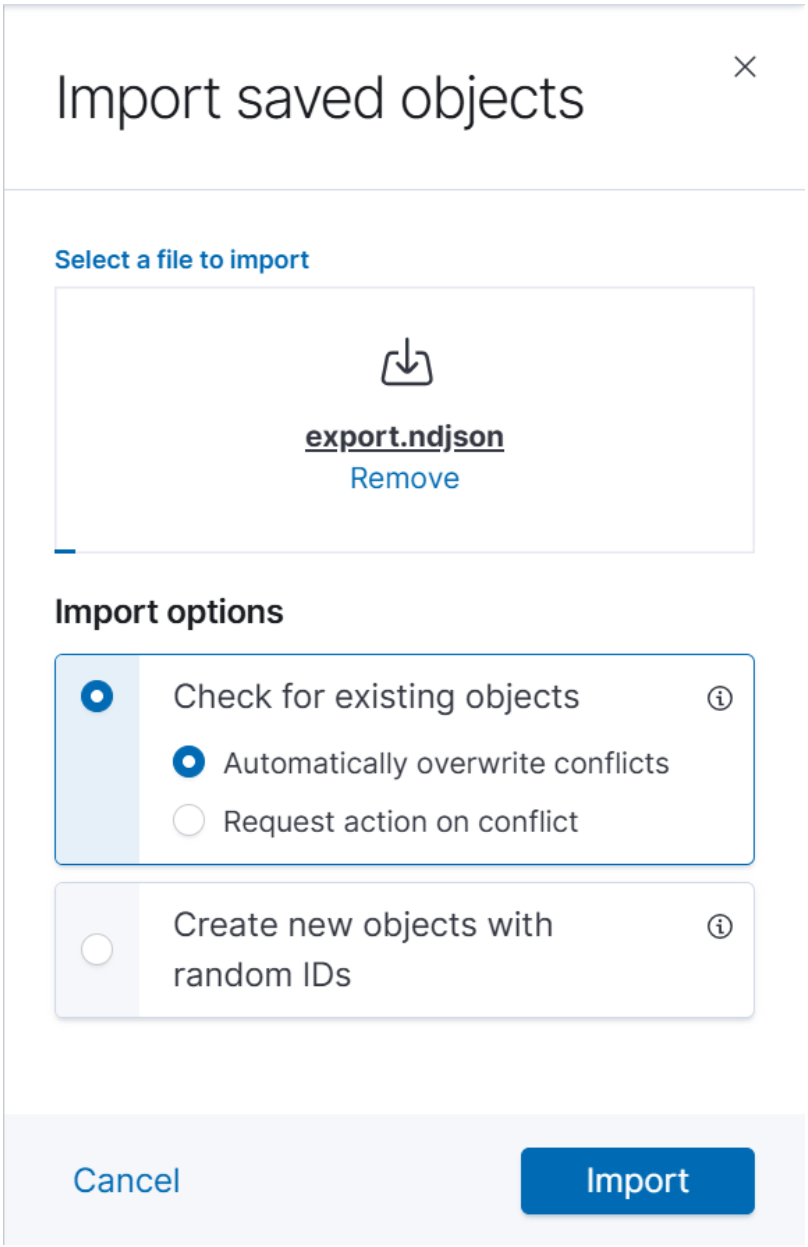
图 1-15 导出可视化对象



步骤2 通过Kibana的图表导入功能将源集群的可视化对象导入目标Elasticsearch集群。

1. 在云搜索服务管理控制台，选择“集群管理 > Elasticsearch”。
2. 在Elasticsearch集群列表，单击集群操作列的“Kibana”，登录Kibana。
3. 在左侧导航栏选择“Stack Management > Saved Objects”。
4. 在Saved Objects页面，单击“Import”，在弹窗中选择保存在本地的源集群的可视化对象文件“export.ndjson”，“Import options”选择“Automatically overwrite conflicts”，单击“Import”。

图 1-16 导入可视化对象



5. 确认导入完成后，单击Done，关闭弹窗。

----结束

常见问题：迁移过程中，发现图表不兼容怎么办?

在导入可视化对象时，出现如下报错信息，则表示源集群和目标集群的图表版本不兼容。

```
The file could not be processed due to error: "Unprocessable Entity: Document "7.1.1" has property "config" which belongs to a more recent version of Kibana [7.13.0]. The last known version is [7.9.0]"
```

此时，可以在本地修改可视化对象文件“export.ndjson”，将版本号信息修改一致，本例中需要将代码中的[7.13.0]修改为[7.9.0]。保存后，重新导入文件即可。如果修改兼容字段后依然导入失败，则需要手动在目标集群重建图表。

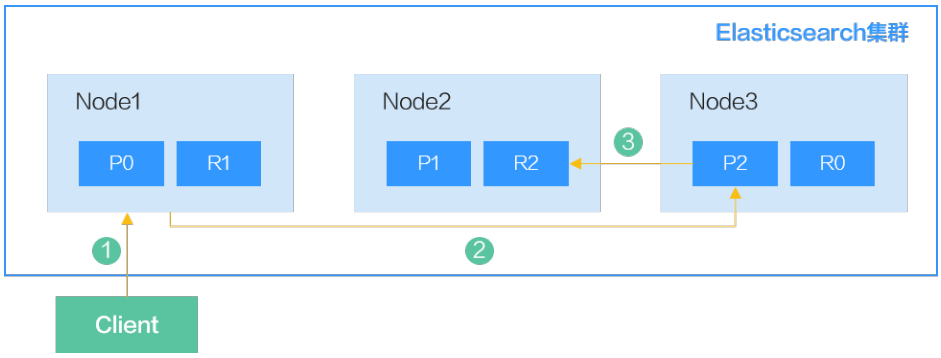
2 优化 Elasticsearch 和 OpenSearch 集群性能

2.1 优化 Elasticsearch 和 OpenSearch 集群写入性能

CSS服务中的Elasticsearch和OpenSearch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 2-1 数据写入流程

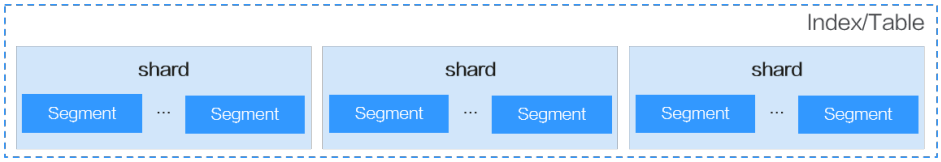


如图2-1所示，以Elasticsearch集群为例，介绍客户端往Elasticsearch或OpenSearch集群中写入数据的流程。图中的P表示主分片Primary，R表示副本分片Replica，主副分片在数据节点Node里是随机分配的，但是不能在同一个节点里。

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。

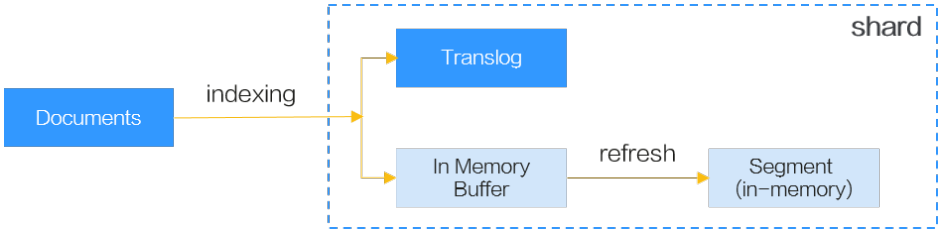
Elasticsearch中的单个索引由一个或多个分片(shard)组成，每个分片包含多个段（Segment），每一个Segment都是一个倒排索引。

图 2-2 Elasticsearch 的索引组成



如图2-3所示，将文档插入Elasticsearch时，文档首先会被写入缓冲区Buffer中，同时写入日志Translog中，然后在刷新时定期从该缓冲区刷新文档到Segment中。刷新频率由refresh_interval参数控制，默认每1秒刷新一次。更多写入性能相关的介绍请参见Elasticsearch的官方介绍**Near Real-Time Search**。

图 2-3 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 2-1 写入性能优化

优化方案	方案说明
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。
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。
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。
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。

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
	}

优化方案	方案说明

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*"
	}

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。

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
	}
	}
	}
	}

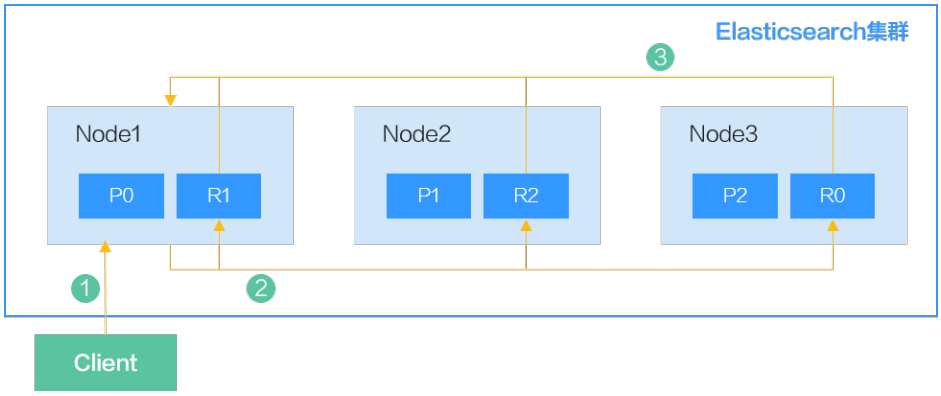
优化shard均衡策略	Elasticsearch默认采用基于磁盘容量大小的Load balance策略，在多节点场景下，尤其是在新扩容的节点上，可能出现shard在各节点上分配不均的问题。为避免这类问题，可以通过设置索引级别的参数“routing.allocation.total_shards_per_node”控制索引分片在各节点的分布情况。此参数可以在索引模板中配置，也可以修改已有索引的setting生效。
	修改已有索引的setting的命令如下：
	PUT *{index}*/_settings
	{
	"index": {
	"routing.allocation.total_shards_per_node": 2
	}
	}

2.2 优化 Elasticsearch 和 OpenSearch 集群查询性能

CSS服务中的Elasticsearch和OpenSearch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 2-4 数据查询流程

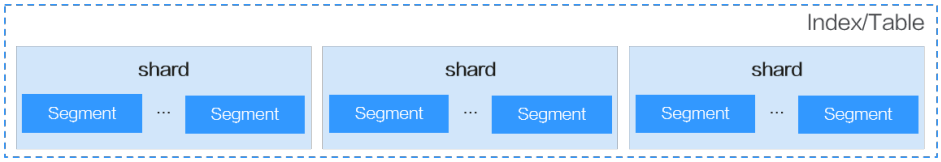


如图2-4所示，以Elasticsearch集群为例，介绍客户端往Elasticsearch或OpenSearch集群发送查询请求的流程。图中的P表示主分片Primary，R表示副本分片Replica，主副分片在数据节点Node里是随机分配的，但是不能在同一个节点里。

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 2-5 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 2-2 查询性能优化

优化方案	方案说明

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`

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"`
	`      }`
	`    }`
	`  }`
	`}`

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**参数配置**，修改参数配置后一定要重启集群使参数生效。

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`

3 Elasticsearch 向量检索的性能测试和比较

应用场景

云搜索服务的向量检索引擎提供了全托管、高性能的分布式向量数据库服务。为了方便用户在业务场景进行向量搜索的性能压力测试，为产品选择和资源配置提供准确的参考依据，本文提供了基于开源数据集和开源压力测试工具的Elasticsearch向量检索的性能测试方案。

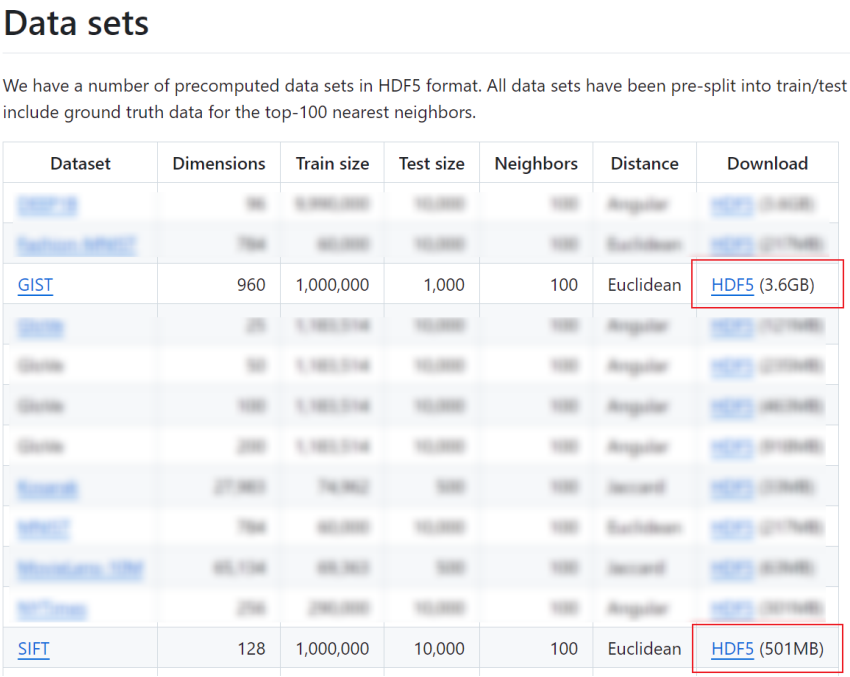
测试前准备

- 创建Elasticsearch向量数据库，参考**创建Elasticsearch集群**。

  “节点数量”选择“3”，“节点规格”选择“通用计算型”的“4vCPUs | 16GB”（由于测试的数据量不大，且为了和第三方的基线测试保持相同的CPU规格），“节点存储”选择“超高I/O”，不启用安全模式。
- 获取测试数据集。
  - sift-128-euclidean：维度128，base数据100万条，使用欧式距离度量。
  - cohere-768-cosine：维度768，base数据100万条，使用余弦距离度量。
  - gist-960-euclidean：维度960，base数据100万条，使用欧式距离度量。

  “sift-128-euclidean”和“gist-960-euclidean”数据的下载地址是**https://github.com/erikbern/ann-benchmarks**。如需使用“cohere-768-cosine”数据，请提交工单获取。

图 3-1 下载“sift-128-euclidean”和“gist-960-euclidean”数据



Data sets

We have a number of precomputed data sets in HDF5 format. All data sets have been pre-split into train/test and include ground truth data for the top-100 nearest neighbors.

Dataset	Dimensions	Train size	Test size	Neighbors	Distance	Download
GIST	960	1,000,000	1,000	100	Euclidean	HDF5 (3.6GB)
SIFT	128	1,000,000	10,000	100	Euclidean	HDF5 (501MB)

- 准备测试工具。
  - 准备数据写入和召回率测试脚本，参考**脚本base_test_example.py**。
  - 下载性能测试使用的开源压测工具Wrk，获取地址**https://github.com/wg/wrk/tree/master**。

性能测试的操作步骤

1. 创建一个弹性云服务器ECS，用于安装压测工具和执行测试脚本。操作指导请参见**快速购买和使用Linux ECS**。
   - ECS必须和Elasticsearch集群在同一个虚拟私有云和安全组中。
   - 也可以使用其他客户端服务器，但是必须保证服务器和Elasticsearch集群在同一VPC。
2. 将测试数据集上传到ECS上。
3. 将数据写入和召回率测试脚本上传到ECS上，并执行如下命令。

   ```
   pip install h5py
   pip install elasticsearch==7.10

   python3 base_test_example.py
   ```

   执行完成后，会创建测试的向量索引，写入测试数据，并返回平均查询召回率Recall。
4. 在ECS上安装开源压测工具Wrk。
5. 在ECS上准备压测的查询请求文件，用于模拟真实业务场景。参考**脚本prepare_query.py**。

   ```
   pip install h5py

   python3 prepare_query.py
   ```
6. 在ECS上准备Wrk的压测配置脚本。参考**脚本perf.lua**，脚本中的查询请求文件名称、集群访问地址和索引名称需要根据实际环境修改。
7. 在ECS执行如下命令进行向量检索的性能压测。

   ```
   wrk -c60 -t60 -d10m -s perf.lua http://x.x.x.x:9200
   ```

- “t”表示压测线程数。
- “c”表示与服务端的连接数。
- “d”表示压测时间，“10m”表示10分钟 。
- “s”表示Wrk的压测配置脚本。
- “x.x.x.x”表示Elasticsearch集群的访问地址。

在回显中获得测试数据，其中“Requests/sec”即查询吞吐量QPS。

图 3-2 测试结果示例

```
root@ecs-          -vdb:~/workspaces/luvector# wrk -c60 -t60 -d5m -s perf.lua
Running 5m test @ http://              :9200
  60 threads and 60 connections
  Thread Stats   Avg      Stdev     Max   +/- Stdev
    Latency     4.16ms    2.70ms 125.82ms   89.86%
    Req/Sec   259.93     38.66     0.92k    68.65%
  4670330 requests in 5.00m, 4.92GB read
Requests/sec:  15562.60
Transfer/sec:     16.80MB
```

性能测试比较

- **GRAPH类索引**

  百万规模的场景推荐使用GRAPH索引类型。

  - **测试方案一**：使用不同维度的数据集，在Top10召回率均达到99%的情况下，测试向量数据库能支撑的最大QPS。每个数据集均基于默认参数和调优参数分别进行测试，通过调整构建参数可以使得图索引结构更优，在同等召回精度下能取得更高的查询性能。

    测试结果：

    表 3-1 GRAPH 类索引测试结果 1

数据集	构建参数		查询参数		性能指标	
	efc	shrink	ef	max_scan_num	QPS	Recall
sift-128-euclidean	200	1.0	84	10000	15562	0.99
	500	0.8	50	10000	17332	0.99
cohere-768-cosine	200	1.0	154	10000	3232	0.99
	500	0.95	106	10000	3821	0.99
gist-960-euclidean	200	1.0	800	19000	860	0.99
	500	0.9	400	15000	1236	0.99

结论：对于不同的数据集，使用默认参数均能达到99%以上的召回率。在进一步调整构建参数和查询参数后，增加了一定的索引构建开销，同时也达到更高的查询性能。

- **测试方案二：**使用同一数据集，通过调整索引参数，测试不同召回率下的查询性能。本方案用COHERE数据集，分别测试了Top10召回率为99%、98%及95%时的集群最大QPS。

  测试结果：

  表 3-2 GRAPH 类索引测试结果 1

数据集	构建参数	查询参数	性能指标	
	efc	ef	QPS	Recall
cohere-768-cosine	500	128	3687	0.99
	500	80	5320	0.98
	500	36	9028	0.95

  结论：同一集群在统一索引构建参数的情况下，通过调整ef参数可以获得不同的查询精度，在略微损失召回率的场景下可以获得成倍的性能提升。

- **GRAPH_PQ类索引**

  基于图算法的索引为了保证查询性能通常需要常驻内存，因此当向量维度较高或数据量较大时，内存资源成为影响成本及性能的关键因素。具体来说，高维度的向量和大规模的数据集对内存的需求显著增加，这不仅关系到存储成本，还直接影响到索引算法的运行效率和响应速度。该场景推荐使用GRAPH_PQ索引类型。

  测试方案：使用维度较高的COHERE与GIST数据集，测试在Top10召回率达到95%时的集群最大QPS，并与GRAPH索引对比常驻内存开销。

  测试结果：

  表 3-3 GRAPH_PQ 类索引测试结果

数据集	构建参数		查询参数		性能指标		内存开销	
	efc	fragment_num	ef	topk	QPS	Recall	GRAPH_PQ	GRAPH
cohere-768-cosine	200	64	85	130	8723	0.95	332MB	3.3GB
gist-960-euclidean	200	120	200	360	4267	0.95	387MB	4.0GB

结论：结果显示使用GRAPH_PQ类索引能够在节约10倍+内存开销的情况下，取得与GRAPH索引差不多的精度和性能。因此，CSS向量索引的GRAPH_PQ算法融合了图索引与量化算法，能够大幅降低内存的开销，提升单机的数据容量。

测试数据中涉及的索引参数说明请参见**表3-4**，关于构建参数的详细说明请参见**在Elasticsearch集群创建向量索引**，关于查询参数的详细说明请参见**在Elasticsearch集群使用向量索引搜索数据**。

表 3-4 索引参数说明

类型	参数名称	说明
构建参数	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。
	shrink	构建hnsw时的裁边系数，默认值为1.0f。
	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。
查询参数	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。
	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。
	topk	查询时返回top k条数据。

脚本“base_test_example.py”

```
# -*- coding: UTF-8 -*-
import json
import time

import h5py
from elasticsearch import Elasticsearch
from elasticsearch import helpers

def get_client(hosts: list, user: str = None, password: str = None):
    if user and password:
        return Elasticsearch(hosts, http_auth=(user, password), verify_certs=False, ssl_show_warn=False)
    else:
        return Elasticsearch(hosts)

# 索引参数说明请参见在Elasticsearch集群创建向量索引。
def create(es_client, index_name, shards, replicas, dim, algorithm="GRAPH",
           metric="euclidean", neighbors=64, efc=200, shrink=1.0):
    index_mapping = {
        "settings": {
            "index": {
                "vector": True
            },
            "number_of_shards": shards,
            "number_of_replicas": replicas,
        },
        "mappings": {
            "properties": {
                "id": {
```



```
                "type": "integer"
            },
            "vec": {
                "type": "vector",
                "indexing": True,
                "dimension": dim,
                "algorithm": algorithm,
                "metric": metric,
                "neighbors": neighbors,
                "efc": efc,
                "shrink": shrink,
            }
        }
    }
    es_client.indices.create(index=index_name, body=index_mapping)
    print(f"Create index success! Index name: {index_name}")

def write(es_client, index_name, vectors, bulk_size=1000):
    print("Start write! Index name: " + index_name)
    start = time.time()
    for i in range(0, len(vectors), bulk_size):
        actions = [{
            "_index": index_name,
            "id": i + j,
            "vec": v.tolist()
        } for j, v in enumerate(vectors[i: i + bulk_size])]
        helpers.bulk(es_client, actions, request_timeout=180)
    print(f"Write success! Docs count: {len(vectors)}, total cost: {time.time() - start:.2f} seconds")
    merge(es_client, index_name)

def merge(es_client, index_name, seg_cnt=1):
    print(f"Start merge! Index name: {index_name}")
    start = time.time()
    es_client.indices.forcemerge(index=index_name, max_num_segments=seg_cnt, request_timeout=7200)
    print(f"Merge success! Total cost: {time.time() - start:.2f} seconds")

# 查询参数说明请参考见在Elasticsearch集群使用向量索引搜索数据。
def query(es_client, index_name, queries, gts, size=10, k=10, ef=200, msn=10000):
    print("Start query! Index name: " + index_name)
    i = 0
    precision = []
    for vec in queries:
        hits = set()
        dsl = {
            "size": size,
            "stored_fields": ["_none_"],
            "docvalue_fields": ["id"],
            "query": {
                "vector": {
                    "vec": {
                        "vector": vec.tolist(),
                        "topk": k,
                        "ef": ef,
                        "max_scan_num": msn
                    }
                }
            }
        }
        res = es_client.search(index=index_name, body=json.dumps(dsl))
        for hit in res['hits']['hits']:
            hits.add(int(hit['fields']['id'][0]))
        precision.append(len(hits.intersection(set(gts[i, :size]))) / size)
        i += 1
    print(f"Query complete! Average precision: {sum(precision) / len(precision)}")

def load_test_data(src):
    hdf5_file = h5py.File(src, "r")
    base_vectors = hdf5_file["train"]
```

```
    query_vectors = hdf5_file["test"]
    ground_truths = hdf5_file["neighbors"]
    return base_vectors, query_vectors, ground_truths

def test_sift(es_client):
    index_name = "index_sift_graph"
    vectors, queries, gts = load_test_data(r"sift-128-euclidean.hdf5")
    # 根据实际测试需求调整分片和副本数、索引算法、索引参数等。本文性能测试均配置的是1个分片、2个副本。
    create(es_client, index_name, shards=1, replicas=2, dim=128)
    write(es_client, index_name, vectors)
    query(es_client, index_name, queries, gts)

if __name__ == "__main__":
    # 此处修改为CSS集群的实际访问地址。
    client = get_client(['http://x.x.x.x:9200'])
    test_sift(client)
```

脚本“prepare_query.py”

```
import base64
import json
import struct

import h5py

def prepare_query(src, dst, size=10, k=10, ef=200, msn=10000, metric="euclidean", rescore=False, use_base64=True):
    """
    使用该函数从hdf5格式的源数据文件读取出query向量，并生成完整的query请求体，用于性能压测。
    :param src: hdf5格式的源数据文件路径。
    :param dst: 目标文件路径。
    :param size: 指定查询返回结果数。
    :param k: 指定Segment级别索引查询topk条相似结果。
    :param ef: 索引查询参数，用于指定查询过程使用的队列大小。
    :param msn: 索引查询参数，用于指定max_scan_num。
    :param metric: 使用何种度量进行rescore精排，euclidean、cosine、inner_product等。
    :param rescore: 是否使用rescore精排，对于GRAPH_PQ索引可以开启精排。
    :param use_base64: 是否使用base64编码的向量数据。
    """
    hdf5_file = h5py.File(src, "r")
    query_vectors = hdf5_file["test"]
    with open(dst, "w", encoding="utf8") as fw:
        for vec in query_vectors:
            query_template = {
                "size": size,
                "stored_fields": ["_none_"],
                "docvalue_fields": ["id"],
                "query": {
                    "vector": {
                        "vec": {
                            "vector": vec.tolist() if not use_base64 else floats2base64(vec),
                            "topk": k,
                            "ef": ef,
                            "max_scan_num": msn,
                        }
                    }
                }
            }
            if rescore:
                query_template["query"]["rescore"] = {
                    "window_size": k,
                    "vector_rescore": {
                        "field": "vec",
                        "vector": vec.tolist() if not use_base64 else floats2base64(vec),
                        "metric": metric
                    }
                }
            fw.write(json.dumps(query_template))
```

```
            fw.write("\n")

def floats2base64(vector):
    data = struct.pack('<{}f'.format(len(vector)), *vector)
    return base64.b64encode(data).decode()

if __name__ == "__main__":
    # 修改为数据文件实际地址。
    prepare_query(r"/path/to/sift-128-euclidean.hdf5", r"requests.txt")
```

脚本“perf.lua”

```
local random = math.random

local reqs = {}
local cnt = 0

-- 压测的查询请求文件名称根据需要调整。
for line in io.lines("requests.txt") do
    table.insert(reqs, line)
    cnt = cnt + 1
end


local addrs = {}
local counter = 0
function setup(thread)
   local append = function(host, port)
      for i, addr in ipairs(wrk.lookup(host, port)) do
         if wrk.connect(addr) then
            addrs[#addrs+1] = addr
         end
      end
   end

   if #addrs == 0 then
      -- 根据集群的实际地址进行修改。
      append("x.x.x.x", 9200)
      append("x.x.x.x", 9200)
      append("x.x.x.x", 9200)
   end

   local index = counter % #addrs + 1
   counter = counter + 1
   thread.addr = addrs[index]
end

-- 索引名称根据需要调整。
wrk.path = "/index_sift_graph/_search?request_cache=false&preference=_local"
wrk.method = "GET"
wrk.headers["Content-Type"] = "application/json"


function request()
    return wrk.format(wrk.method, wrk.path, wrk.headers, reqs[random(cnt)])
end
```

4 使用 Elasticsearch 加速关系型数据库的查询分析

介绍如何将MySQL数据库中的数据同步到云搜索服务的Elasticsearch集群，通过Elasticsearch实现数据库的全文检索、Ad Hoc查询和统计分析能力。

应用场景

使用Elasticsearch加速关系型数据库，可以解决关系型数据库在某些方面的局限性，实现更加高效和智能的数据处理和分析。常用于以下应用场景：

- 电子商务平台：快速搜索商品，提供个性化推荐，实时监控用户行为和交易数据。
- 内容管理系统：对大量文档和内容进行高效检索，支持复杂的查询和数据分析。
- 金融服务：实时监控交易数据，进行风险分析和欺诈检测。
- 社交媒体分析：对用户生成的内容进行情感分析，趋势和影响力评估。
- 客户关系管理：快速检索客户信息，分析客户行为，提供定制化服务。
- 日志和事件监控：收集和分析大量日志数据，实时监控系统状态和安全事件。
- 医疗健康记录：快速检索和分析患者记录，支持临床决策和研究。

方案架构

图 4-1 Elasticsearch 加速关系型数据库的方案架构

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服务的Elasticsearch集群。
3. 在Elasticsearch集群中进行全文检索、Ad Hoc查询和统计分析。

方案优势

以下是使用Elasticsearch加速关系型数据库的方案优势：

- 全文检索能力提升：Elasticsearch是一个提供了强大的全文检索功能的搜索引擎。关系型数据库通常不擅长进行全文检索，而Elasticsearch可以有效地解决这一问题。
- 高并发Ad Hoc查询：Elasticsearch设计用于处理大量的并发查询请求，特别是在Ad Hoc查询场景下，它能够提供快速的响应时间，从而满足高并发环境下的查询需求。
- 实时数据同步：通过华为云数据复制服务DRS，可以实现MySQL数据库中的数据实时同步到Elasticsearch，确保数据的一致性和实时性。
- 简化的数据迁移和索引创建：在Elasticsearch中，可以创建与MySQL数据库表结构相对应的索引，简化了数据迁移和索引管理的复杂性。
- 灵活的查询语言：Elasticsearch提供了灵活的查询语言，支持复杂查询的构建，如范围查询、模糊查询、聚合查询等，这在关系型数据库中可能需要更复杂的SQL语句来实现。
- 统计分析能力：Elasticsearch的聚合功能可以快速进行数据统计和分析，如年龄分布统计等，这在关系型数据库中可能需要更多的计算资源和时间。
- 安全性和稳定性：通过配置安全模式的Elasticsearch集群和MySQL数据库，以及使用SSL安全连接，可以保证数据传输的安全性和系统的稳定性。
- 易于监控和维护：Elasticsearch提供了丰富的监控工具和API，使得系统维护和性能监控变得更加容易。
- 扩展性：Elasticsearch集群可以根据业务需求进行水平扩展，增加更多的节点来处理更大的数据量和查询负载。

这些优势使得Elasticsearch集群成为关系型数据库在处理全文检索和高并发Ad Hoc查询时的有效补充。

前提条件

- 已具备安全模式的Elasticsearch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。

  本文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 actor, rapper, and producer.'),
  ('3','James Francis Cameron','68','the director of avatar');
  ```

- Elasticsearch集群中已完成索引创建，且与MySQL中表相对应。

  本文执行如下命令创建Elasticsearch集群的索引。

  ```
  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**将MySQL同步到CSS/ES**。

在本章案例中，**表4-1**中的同步任务配置参数需要按建议填写。

表 4-1 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Elasticsearch集群所在子网保持一致。
源库及目标库 > 目标库信息	VPC	选择和Elasticsearch集群一致的VPC。
	子网	选择和Elasticsearch集群一致的子网。
	IP地址或域名	填写Elasticsearch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	填写Elasticsearch集群的管理员账户名（admin）。
	数据库密码	填写Elasticsearch集群的管理员密码。
	加密证书	选择Elasticsearch集群的安全证书，如果未启用“SSL安全连接”，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Elasticsearch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Elasticsearch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”，如果不一致请修改。
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Elasticsearch集群的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Elasticsearch。

   例如，源端插入“id”为“4”的新数据。

   ```
   INSERT INTO student (id,name,age,dsc)
   VALUES
   ('4','Bill Gates','50','Gates III is a business magnate, software developer, investor, author, and philanthropist.')
   ```

   在Elasticsearch集群的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Elasticsearch。

   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Elasticsearch集群的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Elasticsearch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Elasticsearch集群的Kibana中执行如下命令，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Elasticsearch集群查询“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Elasticsearch集群查询年龄大于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Elasticsearch集群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”。

图 4-2 下载证书

5 使用 Elasticsearch、自建 Logstash 和 Kibana 构建日志管理平台

使用CSS服务的Elasticsearch集群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。

应用场景

本文以Elasticsearch、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Elasticsearch中，最后通过Kibana进行日志的可视化查询与分析。该方案可以用于以下场景：

- 日志管理：集中管理应用程序和系统日志，快速定位问题。
- 安全监控：检测和响应安全威胁，进行入侵检测和异常行为分析。
- 业务分析：分析用户行为，优化产品和服务。
- 性能监控：监控系统和应用程序性能，实时发现瓶颈。

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。

- Elasticsearch是一个开源分布式的搜索和分析引擎，用于存储、搜索和分析大量数据。
- Logstash是一个服务器端的数据管道，负责收集、解析和丰富数据后，将其发送到Elasticsearch。
- Kibana为Elasticsearch提供一个开源的数据分析和可视化平台，用于对Elasticsearch中的数据进行搜索、查看和交互。
- Beats：轻量级的数据收集器（如Filebeat、Metricbeat等），安装在服务器上，负责收集和转发数据到Logstash。

使用Elasticsearch和Logstash构建日志管理平台的方案架构如图5-1所示。

图 5-1 ELKB 架构

1. 数据收集
   - Beats作为数据收集器，负责从各种源收集数据并发送到Logstash。
   - Logstash可以独立收集数据，或从Beats接收数据，对数据进行过滤、转换和增强。
2. 数据处理

   Logstash在将数据发送到Elasticsearch之前，对数据进行必要的处理，如解析结构化日志、过滤无用信息等。
3. 数据存储

   Elasticsearch作为核心存储组件，Elasticsearch索引和存储来自Logstash的数据，提供快速搜索和数据检索功能。
4. 数据分析与可视化

   使用Kibana对Elasticsearch中的数据进行分析和可视化，创建仪表板和报告，以直观展示数据。

ELKB系统中各组件的版本兼容性请参见**https://www.elastic.co/support/matrix#matrix_compatibility**。

方案优势

- 实时性：提供实时数据收集和分析能力。
- 灵活性：支持各种数据源和灵活的数据处理流程。
- 易用性：用户界面友好，简化了数据操作和可视化过程。
- 扩展性：水平扩展能力强，可以处理PB级别的数据。

前提条件

- 已创建非安全模式的Elasticsearch集群，操作步骤请参见**创建Elasticsearch集群**。
- 已申请弹性云服务器ECS，并安装了Java环境，购买ECS请参见**快速购买和使用Linux ECS**。

操作步骤

步骤1 登录ECS，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 部署并配置自建Logstash。

说明

为了获得更优的性能，自建Logstash中的JVM参数建议配置为“ECS/docker”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，在虚拟机上执行命令**curl http:// {ip}:{port}**测试虚拟机和ELasticsearch集群的连通性，返回200，表示已经连通。
3. 配置Logstash的配置文件“logstash-sample.conf”。

   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（**https://grokdebugger.com/**）。

步骤3 配置Elasticsearch集群的索引模板。

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”。
3. 在集群列表，选择待操作的集群，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. 创建一个索引模板令。

   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||strict_date_optional_time||epoch_millis||EEE MMM dd HH:mm:ss zzz yyyy"
        }
      }
   }
}
```

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 &
```

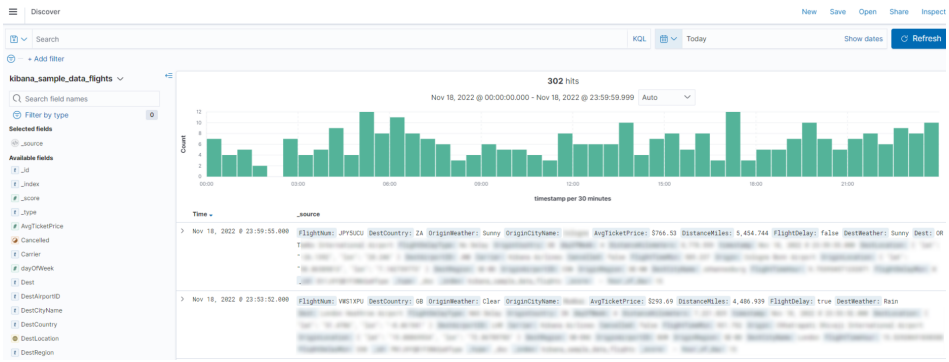
步骤6 执行如下命令，启动Filebeat。

```
./filebeat
```

步骤7 通过Kibana进行查询并制作报表。

1. 进入Elasticsearch集群的Kibana操作界面。
2. 单击左侧导航栏的“Discover”进行查询与分析，类似的效果如图5-2所示。

图 5-2 Discover 界面示例



----结束

6 使用 Elasticsearch 自定义规则排序搜索结果

通过Elasticsearch集群可以对搜索结果进行自定义规则排序。

应用场景

Elasticsearch是一个高度可扩展的开源搜索和分析引擎，支持用户通过自定义规则对搜索结果进行排序。自定义排序允许开发者根据业务需求，定义特定的排序规则，以优化搜索结果的相关性和用户体验。该方案可以用于以下场景：

- 电子商务：根据销量、用户评价、价格等因素对商品进行排序。
- 内容管理：根据阅读量、发布时间对文章或博客帖子进行排序。
- 金融服务：根据交易金额、频率或风险评分对交易记录进行排序。
- 客户支持：根据工单的紧急程度或打开时间对客户请求进行排序。

方案架构

通过自定义规则对搜索结果进行排序是通过Elasticsearch的排序API实现的。通过调用排序API查询数据，实现数据按自定义规则排序。

自定义规则查询有两种方式。

- 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。

方案优势

- 灵活性：自定义排序规则可以满足各种复杂的业务需求。
- 扩展性：Elasticsearch的分布式特性支持水平扩展，适应不断增长的数据量。
- 性能：Elasticsearch的优化机制确保了排序操作的高效性，即使在大规模数据集上也能保持良好的性能。
- 实时性：Elasticsearch的近实时搜索能力确保排序结果的时效性。

前提条件

已经准备好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. 自定义规则查询数据。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
        }
      }
    ]
  }
}
```

7 使用 Logstash 将 RDS MySQL 数据同步至 Elasticsearch

应用场景

CSS服务的Logstash集群默认安装了logstash-input-jdbc插件，该插件为Logstash提供了从关系型数据库RDS MySQL中导入和处理数据的能力，通过配置Logstash配置文件，定义JDBC输入和Elasticsearch输出，实现定期同步数据库中的数据到Elasticsearch。该方案可以用于以下场景：

- 数据实时更新与同步：将RDS MySQL中的数据实时同步到Elasticsearch，以便利用Elasticsearch强大的搜索和分析能力。
- 日志分析与检索：将MySQL中的日志数据同步到Elasticsearch，进行快速检索和分析。
- 应用性能监控：将应用性能数据存储在MySQL中，通过Logstash同步到Elasticsearch，进行实时监控和性能分析。
- 数据备份与恢复：通过Logstash将MySQL数据备份到Elasticsearch，以便在数据丢失或损坏时快速恢复。

方案架构

图 7-1 RDS MySQL 数据同步至 Elasticsearch

使用Logstash将RDS MySQL数据同步至Elasticsearch的方案流程如图7-1所示。

在CSS服务的Logstash中通过默认插件logstash-input-jdbc，在Logstash配置文件中配置数据库JDBC输入和Elasticsearch输出，将全量或增量RDS MySQL数据实时同步至Elasticsearch。

方案优势

- 灵活性：Logstash提供了数据采集、转换、优化和输出的能力，可以灵活地处理各种数据同步需求。
- 实时性：Logstash可以实现数据的准实时同步，满足大多数业务场景的需求。
- 易用性：通过Logstash配置文件即可实现数据同步，操作简单，无需复杂的代码开发。

约束限制

- Elasticsearch中的_id字段必须与MySQL中的id字段相同。

  这是为了确保当MySQL中的记录写入Elasticsearch时，同步任务可在MySQL记录与Elasticsearch索引之间建立一个直接映射的关系。例如，当MySQL中更新了某条记录时，同步任务会覆盖Elasticsearch中与更新记录具有相同ID的索引。
- 当MySQL中插入或者更新数据时，对应记录必须有一个包含更新或插入时间的字段。

  Logstash在每次轮询MySQL时，会记录最后读取记录的时间戳，并在下一次读取时只获取该时间戳之后更新或插入的符合条件的记录。
- 确保MySQL数据库、Logstash集群和Elasticsearch集群在同一时区，否则当同步与时间相关的数据时，同步前后的数据可能存在时区差。

前提条件

- 已准备好存有数据的MySQL数据库，本案例以云数据库的RDS for MySQL实例为例，具体操作请参见**购买RDS for MySQL实例**。
- 已准备好用于同步数据的Logstash集群，具体操作请参见**创建Logstash集群**。本文以7.10.0版本的Logstash集群为例。
- 已准备好Elasticsearch集群，具体操作请参见**创建Elasticsearch集群**。本文以7.10.2版本的Elasticsearch集群为例。

以上三者在同一VPC下面。

当使用的是自建或第三方MySQL数据库时，则需要确认数据库驱动是否是MariaDB驱动。

- 是，则可以直接开始配置数据同步。
- 否，则需要参考**常见问题：MySQL驱动不兼容怎么办？**上传与RDS版本兼容的SQL JDBC驱动到Logstash集群中。

操作步骤

步骤1 验证Logstash集群和数据源之间的网络连通性。

1. 登录云搜索服务管理控制台。
2. 左侧导航栏选择“集群管理 > Logstash”，进入集群列表页面。
3. 在集群列表，单击集群操作列的“配置中心”，进入配置中心页面。

   或者，在集群列表，单击集群名称，进入集群基本信息页面，在左侧导航栏选择“配置中心”，进入配置中心页面。
4. 在配置中心页面，单击“连通性测试”。
5. 在连通性测试弹窗中，输入数据源、目的端的IP地址和端口号，单击“测试”。

连通性测试最多可一次性测试10个IP地址。您可以单击“继续添加”，添加多个IP地址，然后单击“批量测试”，进行一次性测试多个IP地址的连通性。

图 7-2 连通性测试

当显示“可用”时，表示集群间网络连通。如果网络不连通，可以配置Logstash集群路由，连通集群间的网络，具体操作请参见**配置Logstash集群路由**。

步骤2 创建用于数据同步的Logstash配置文件。

1. 在Logstash集群的配置中心页面，单击右上角“创建”，进入创建配置文件页面，编辑配置文件。

表 7-1 创建配置文件

参数	说明
名称	自定义配置文件名称。 只能包含字母、数字、中划线或下划线，且必须以字母开头。必须大于等于4个字符。
配置文件内容	参考下面的代码示例开发配置文件内容。 **说明** 配置文件内容大小不能超过100k。
隐藏内容列表	配置隐藏字符串列表后，在返回的配置内容中，会将所有在列表中的字串隐藏为“***”。 本案例不用配置。

```
input {
  jdbc{
    # JDBC驱动配置。
    jdbc_driver_library => "/rds/datastore/logstash/v7.10.0/package/logstash-7.10.0/extend/jars/
mariadb-java-client-2.7.0.jar"
    jdbc_driver_class => "org.mariadb.jdbc.Driver"
    jdbc_connection_string => "jdbc:mariadb://xxx.xxx.xxx.xxx:port/cms?
useUnicode=true&characterEncoding=utf8mb4&autoReconnect=true&allowMultiQueries=true"
    jdbc_user => "root"
    jdbc_password => "xx"
    # 以下保持默认即可。
    jdbc_paging_enabled => "true"
    jdbc_page_size => "50000"
    # 迁移数据的SQL查询语句。
    statement => "select a.user_code AS doctor_id,a.record_status from cluster "
    # 定时任务，每5分钟同步一次，可以自定义。
    schedule => "*/5 * * * *"
  }
}
filter {
```

```
}
output {
  elasticsearch {
    hosts => ["xxx.xxx.xxx.xxx:port","xxx.xxx.xxx.xxx:port","xxx.xxx.xxx.xxx:port"]
    # 设置索引名称。
    index => "rds_doctor_index"
    user => "admin"
    password => "xx"
    # 索引中的文档id，建议和MySQL中表的主键名称保持一致。
    document_id => "%{primary_id}"
    # 目标Elasticsearch集群启用HTTPS访问时，才需要配置证书。
    ssl => true
    ssl_certificate_verification => false
    cacert => "/rds/datastore/logstash/v7.10.0/package/logstash-7.10.0/extend/cert/
CloudSearchService.cer"
    # 以下保持默认即可。
    manage_template => false
    ilm_enabled => false
  }
}
```

表 7-2 配置项说明

配置项名称		是否必填	说明
input	jdbc_driver_library	是	JDBC驱动程序库路径。 – 当数据库驱动是MariaDB驱动时，该值填写“/rds/datastore/logstash/v7.10.0/package/logstash-7.10.0/extend/jars/mariadb-java-client-2.7.0.jar”。 – 当数据库驱动是RDS版本兼容的SQL JDBC驱动时，该值需要联系技术支持修改。 JDBC驱动相关的详细参数配置请参见**Jdbc input plugin**。
	jdbc_driver_class	是	驱动程序库的class路径。 – 当数据库驱动是MariaDB驱动时，该值填写“org.mariadb.jdbc.Driver”。 – 当数据库驱动是RDS版本兼容的SQL JDBC驱动时，该值填写“com.mysql.jdbc.Driver”。

配置项名称		是否必填	说明
	jdbc_connection_string	是	MySQL JDBC的访问地址。 – 当数据库驱动是MariaDB驱动时，该值填写“jdbc:mariadb://xxx.xxx.xxx.xxx:port/cms?useUnicode=true&characterEncoding=utf8mb4&autoReconnect=true&allowMultiQueries=true”。 – 当数据库驱动是RDS版本兼容的SQL JDBC驱动时，该值填写“jdbc:mysql://xxx.xxx.xxx.xxx:port/cms”。 其中“xxx.xxx.xxx.xxx:port”填写数据库实际访问地址和端口号。
	jdbc_user	是	访问MySQL JDBC的用户名。
	jdbc_password	是	访问MySQL JDBC的密码。
	statement	是	迁移数据的SQL查询语句。
	schedule	是	定时任务，支持自定义同步周期。
output	hosts	是	Elasticsearch集群的访问地址。
	index	是	设置索引名称，即数据导入到哪个索引。
	user	否	访问Elasticsearch集群的用户名，仅安全集群涉及。
	password	否	访问Elasticsearch集群的密码，仅安全集群涉及。
	document_id	是	索引中的文档ID，建议和MySQL的记录ID（例如表的主键名称primary_id）保持一致。
	ssl	否	是否开启HTTPS通信。 当Elasticsearch集群启用HTTPS访问时，该值设置为“true”，否则不用配置。
	ssl_certificate_verification	否	是否验证服务端Elasticsearch证书。仅当“ssl”配置为“true”时，才需要配置该参数。 – true：验证证书。 – false：忽略证书。
	cacert	否	HTTPS访问证书，CSS集群保持默认值。

2. 编辑完成后，单击“下一页”配置Logstash管道参数。本案例保持默认值即可。
3. 配置完成后，单击“创建”。

   在配置中心页面可以看到创建的配置文件，状态为“可用”，表示创建成功。

步骤3 启动Logstash配置文件。

1. 在配置文件列表，选择需要启动的配置文件，单击左上角的“启动”。
2. 在“启动Logstash服务”对话框中，勾选“是否保持常驻”开启Logstash服务保持常驻。
3. 单击“确定”，开始启动配置文件启动Logstash迁移任务。

   可以在管道列表看到启动的配置文件。

步骤4 验证数据库和Elasticsearch集群的数据是否已同步。

1. 在云搜索服务管理控制台，选择“集群管理 > Elasticsearch”。
2. 在Elasticsearch集群列表，单击集群操作列的“Kibana”，登录Kibana。
3. 在左侧导航栏选择“Dev Tools”，进入Console页面。
4. 执行如下命令查询索引数据。

```
GET rds_doctor_index/_count
{
  "query": {"match_all": {}}
}
```

   当返回结果中，“count”的值不为0，则表示数据同步已成功。

----结束

常见问题：MySQL 驱动不兼容怎么办?

在Logstash集群启动Logstash配置文件后，在Logstash管道运行状态异常，单击“运行日志”，可以日志中看到类似以下报错信息，则表示MySQL驱动不兼容。

```
[2024-05-21T11:31:00,196][ERROR][logstash.inputs.jdbc     ] Java::JavaSql::SQLSyntaxErrorException: (conn=-1409730930) You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near '"T1" LIMIT 1' at line 1: SELECT count(*) AS "COUNT" FROM (select * from logstash_broker where updatetime+30000 > 0 order by updatetime) AS "T1" LIMIT 1
```

解决方案：

1. 停止Logstash配置文件。
2. 下载与RDS版本兼容的SQL JDBC驱动，例如“mysql-connector-java-8.0.11.tar.gz”，解压后获得“mysql-connector-java-8.0.11.jar”。

   下载地址：**https://downloads.mysql.com/archives/c-j/**
3. 联系技术支持，将SQL JDBC驱动jar包上传至用于同步数据的Logstash集群中。
4. 修改Logstash配置文件内容。

   修改参数值“jdbc_driver_class”和“jdbc_connection_string”。其中“xxx.xxx.xxx.xxx:port”填写数据库实际访问地址和端口号。

```
jdbc_driver_class => "com.mysql.jdbc.Driver"
# 填写为MySQL JDBC的访问地址。
jdbc_connection_string => "jdbc:mysql://xxx.xxx.xxx.xxx:port/cms"
```

5. 重新启动配置文件。

