

Cloud Search Service

Best Practices

Issue 01
Date 2025-01-06



Copyright © Huawei Cloud Computing Technologies Co., Ltd. 2025. All rights reserved.

No part of this document may be reproduced or transmitted in any form or by any means without prior written consent of Huawei Cloud Computing Technologies Co., Ltd.

Trademarks and Permissions



HUAWEI and other Huawei trademarks are the property of Huawei Technologies Co., Ltd.

All other trademarks and trade names mentioned in this document are the property of their respective holders.

Notice

The purchased products, services and features are stipulated by the contract made between Huawei Cloud and the customer. All or part of the products, services and features described in this document may not be within the purchase scope or the usage scope. Unless otherwise specified in the contract, all statements, information, and recommendations in this document are provided "AS IS" without warranties, guarantees or representations of any kind, either express or implied.

The information in this document is subject to change without notice. Every effort has been made in the preparation of this document to ensure accuracy of the contents, but all statements, information, and recommendations in this document do not constitute a warranty of any kind, express or implied.

Huawei Cloud Computing Technologies Co., Ltd.

Address: Huawei Cloud Data Center Jiaoxinggong Road
Qianzhong Avenue
Gui'an New District
Gui Zhou 550029
People's Republic of China

Website: <https://www.huaweicloud.com/intl/en-us/>

Contents

1 Elasticsearch Data Migration.....	1
1.1 About Elasticsearch Cluster Migration Solutions.....	1
1.2 Migrating Data Between Elasticsearch Clusters Using Huawei Cloud Logstash.....	10
1.3 Migrating Data Between Huawei Cloud Elasticsearch Clusters Using Backup and Restoration.....	31
1.4 Migrating Data from an On-premises Elasticsearch Cluster to Huawei Cloud Using the S3 Plugin.....	36
1.5 Migrating Data from a Third-Party Elasticsearch Cluster to Huawei Cloud Using Backup and Restoration.....	40
1.6 Migrating Data Between Huawei Cloud Elasticsearch Clusters Using the Read/Write Splitting Plugin.....	44
1.7 Migrating Data Between Elasticsearch Clusters Using the Reindex API.....	49
1.8 Migrating Data Between Elasticsearch Clusters Using ESM.....	55
1.9 Migrating Kibana Saved Objects Between Elasticsearch Clusters.....	60
2 Optimizing the Performance of Elasticsearch and OpenSearch Clusters.....	66
2.1 Optimizing the Write Performance of Elasticsearch and OpenSearch Clusters.....	66
2.2 Optimizing the Query Performance of Elasticsearch and OpenSearch Clusters.....	69
3 Testing the Performance of CSS's Elasticsearch Vector Search.....	72
4 Using Elasticsearch to Accelerate Query and Analysis for Relational Databases.....	81
5 Using Elasticsearch, In-House Built Logstash, and Kibana to Build a Log Management Platform.....	88
6 Ranking Search Results Using Elasticsearch Custom Rules.....	93
7 Synchronizing Data from RDS for MySQL to Elasticsearch Through Logstash.....	99

1 Elasticsearch Data Migration

1.1 About Elasticsearch Cluster Migration Solutions

Table 1-1 Elasticsearch cluster migration solutions

Migration Scenario	Migration Tool/ Method	Applicable Scenario	Constraints	Example
Migrating data between Huawei Cloud Elasticsearch clusters	Huawei Cloud Logstash	<ul style="list-style-type: none"> Migrate data from a Huawei Cloud Elasticsearch cluster of an earlier version to one of a later version. Migrate data from multiple Huawei Cloud Elasticsearch clusters to a single Huawei Cloud Elasticsearch cluster to form a unified data platform. 	During cluster migration, do not add, delete, or modify indexes in the source cluster, or the source and destination clusters will have inconsistent data after the migration.	Migrating Data Between Elasticsearch Clusters Using Huawei Cloud Logstash

Migration Scenario	Migration Tool/Method	Applicable Scenario	Constraints	Example
	Backup and restoration	<ul style="list-style-type: none"> • Migrate data between Huawei Cloud Elasticsearch clusters in the same region or across regions, and under the same or different accounts. • Migrate data from a Huawei Cloud Elasticsearch cluster of an earlier version to one of a later version. • Migrate data from multiple Huawei Cloud Elasticsearch clusters to a single Huawei Cloud Elasticsearch cluster to form a unified data platform. 	<ul style="list-style-type: none"> • The version of the destination cluster must not be earlier than that of the source cluster. For details, see Snapshot version compatibility. • The number of nodes in the destination cluster must be greater than half of that in the source cluster, and cannot be less than the number of shard replicas in the source cluster. • The CPU, memory, and disk capacities of the destination cluster must not be lower than those of the source cluster. 	Migrating Data Between Huawei Cloud Elasticsearch Clusters Using Backup and Restoration

Migration Scenario	Migration Tool/ Method	Applicable Scenario	Constraints	Example
	Read/write splitting plugin	<ul style="list-style-type: none"> • Migrate data between Huawei Cloud Elasticsearch clusters in the same region or across regions, and under the same or different accounts. • Migrate data from multiple Huawei Cloud Elasticsearch clusters to a single Huawei Cloud Elasticsearch cluster to form a unified data platform. 	The versions of the source and destination clusters must both be 7.6.2 or 7.10.2.	Migrating Data Between Huawei Cloud Elasticsearch Clusters Using the Read/Write Splitting Plugin
	Reindex API	Migrate data from multiple Huawei Cloud Elasticsearch clusters to a single Huawei Cloud Elasticsearch cluster to form a unified data platform.	During cluster migration, do not add, delete, or modify indexes in the source cluster, or the source and destination clusters will have inconsistent data after the migration.	Migrating Data Between Elasticsearch Clusters Using the Reindex API

Migration Scenario	Migration Tool/ Method	Applicable Scenario	Constraints	Example
	ESM	<ul style="list-style-type: none"> • Migrate data from a Huawei Cloud Elasticsearch cluster of an earlier version to one of a later version. • Migrate data from multiple Huawei Cloud Elasticsearch clusters to a single Huawei Cloud Elasticsearch cluster to form a unified data platform. 	<p>During cluster migration, do not add, delete, or modify indexes in the source cluster, or the source and destination clusters will have inconsistent data after the migration.</p>	<p>Migrating Data Between Elasticsearch Clusters Using ESM</p>
<p>Migrating data from an in-house built Elasticsearch cluster to Huawei Cloud</p>	<p>Huawei Cloud Logstash</p>	<ul style="list-style-type: none"> • Migrate data from an on-premises Elasticsearch cluster of an earlier version to a Huawei Cloud Elasticsearch cluster of a later version. • Migrate data from multiple on-premises Elasticsearch clusters to a single Huawei Cloud Elasticsearch cluster to form a unified data platform. • Migrate in-house built Elasticsearch workloads to Huawei Cloud. 	<ul style="list-style-type: none"> • During cluster migration, do not add, delete, or modify indexes in the source cluster, or the source and destination clusters will have inconsistent data after the migration. • Ensure that the network between the clusters is connected. Configure public network connectivity for in-house built Elasticsearch cluster. 	<p>Migrating Data Between Elasticsearch Clusters Using Huawei Cloud Logstash</p>

Migration Scenario	Migration Tool/Method	Applicable Scenario	Constraints	Example
	Backup and restoration	<ul style="list-style-type: none"> • Migrate data from an on-premises Elasticsearch cluster of an earlier version to a Huawei Cloud Elasticsearch cluster of a later version. • Migrate data from multiple on-premises Elasticsearch clusters to a single Huawei Cloud Elasticsearch cluster to form a unified data platform. • Migrate in-house built Elasticsearch workloads to Huawei Cloud. 	<ul style="list-style-type: none"> • The version of the destination cluster must not be earlier than that of the source cluster. For details, see Snapshot version compatibility. • This method does not support incremental data synchronization. You need to pause data update before starting to back up data. • An on-premises Elasticsearch cluster needs public network access to back up snapshots to OBS. 	Migrating Data from an On-premises Elasticsearch Cluster to Huawei Cloud Using the S3 Plugin
	Reindex API	<ul style="list-style-type: none"> • Migrate data from multiple on-premises Elasticsearch clusters to a single Huawei Cloud Elasticsearch cluster to form a unified data platform. • Migrate in-house built Elasticsearch workloads to Huawei Cloud. 	<p>During cluster migration, do not add, delete, or modify indexes in the source cluster, or the source and destination clusters will have inconsistent data after the migration.</p>	Migrating Data Between Elasticsearch Clusters Using the Reindex API

Migration Scenario	Migration Tool/ Method	Applicable Scenario	Constraints	Example
	ESM	<ul style="list-style-type: none"> • Migrate data from an on-premises Elasticsearch cluster of an earlier version to a Huawei Cloud Elasticsearch cluster of a later version. • Migrate data from multiple on-premises Elasticsearch clusters to a single Huawei Cloud Elasticsearch cluster to form a unified data platform. • Migrate in-house built Elasticsearch workloads to Huawei Cloud. 	During cluster migration, do not add, delete, or modify indexes in the source cluster, or the source and destination clusters will have inconsistent data after the migration.	Migrating Data Between Elasticsearch Clusters Using ESM
	CDM	This is a cloud migration tool provided by Huawei Cloud to migrate data between clusters that belong to different cloud services.	<ul style="list-style-type: none"> • A VPN or Direct Connect connection needs to be established between the customer's IDC and Huawei Cloud. • During cluster migration, do not delete any of the source cluster's indexes, or something might go wrong. 	Migrating an Entire Elasticsearch Database to CSS

Migration Scenario	Migration Tool/ Method	Applicable Scenario	Constraints	Example
Migrating data from a third-party Elasticsearch cluster to Huawei Cloud	Huawei Cloud Logstash	<ul style="list-style-type: none"> • Migrate Elasticsearch clusters across major versions, for example, from 6.X to 7.X. • Migrate data from multiple third-party Elasticsearch clusters to a single Huawei Cloud Elasticsearch cluster to form a unified data platform. • Migrate third-party Elasticsearch workloads to Huawei Cloud. 	<ul style="list-style-type: none"> • During cluster migration, do not add, delete, or modify indexes in the source cluster, or the source and destination clusters will have inconsistent data after the migration. • Ensure that the network between the clusters is connected. A VPN or Direct Connect connection needs to be established between the customer's IDC and Huawei Cloud. 	Migrating Data Between Elasticsearch Clusters Using Huawei Cloud Logstash

Migration Scenario	Migration Tool/Method	Applicable Scenario	Constraints	Example
	Backup and restoration	<ul style="list-style-type: none"> • Migrate data from a third-party Elasticsearch cluster of an earlier version to a Huawei Cloud Elasticsearch cluster of a later version. • Migrate data from multiple third-party Elasticsearch clusters to a single Huawei Cloud Elasticsearch cluster to form a unified data platform. • Migrate third-party Elasticsearch workloads to Huawei Cloud. 	<ul style="list-style-type: none"> • The version of the destination cluster must not be earlier than that of the source cluster. For details, see Snapshot version compatibility. • This method does not support incremental data synchronization. You need to pause data update before starting to back up data. • Snapshots can be migrated only when public access is enabled for third-party storage repositories. 	Migrating Data from a Third-Party Elasticsearch Cluster to Huawei Cloud Using Backup and Restoration
	Reindex API	<ul style="list-style-type: none"> • Migrate data from multiple third-party Elasticsearch clusters to a single Huawei Cloud Elasticsearch cluster to form a unified data platform. • Migrate third-party Elasticsearch workloads to Huawei Cloud. 	<p>During cluster migration, do not add, delete, or modify indexes in the source cluster, or the source and destination clusters will have inconsistent data after the migration.</p>	Migrating Data Between Elasticsearch Clusters Using the Reindex API

Migration Scenario	Migration Tool/ Method	Applicable Scenario	Constraints	Example
	ESM	<ul style="list-style-type: none"> • Migrate data from a third-party Elasticsearch cluster of an earlier version to a Huawei Cloud Elasticsearch cluster of a later version. • Migrate data from multiple third-party Elasticsearch clusters to a single Huawei Cloud Elasticsearch cluster to form a unified data platform. • Migrate third-party Elasticsearch workloads to Huawei Cloud. 	During cluster migration, do not add, delete, or modify indexes in the source cluster, or the source and destination clusters will have inconsistent data after the migration.	Migrating Data Between Elasticsearch Clusters Using ESM
	CDM	This is a cloud migration tool provided by Huawei Cloud to migrate data between clusters that belong to different cloud services.	<ul style="list-style-type: none"> • A VPN or Direct Connect connection needs to be established between the customer's IDC and Huawei Cloud. • During cluster migration, do not delete any of the source cluster's indexes, or something might go wrong. 	Migrating an Entire Elasticsearch Database to CSS

Migration Scenario	Migration Tool/Method	Applicable Scenario	Constraints	Example
Migrating data from a Huawei Cloud Elasticsearch Cluster to an OpenSearch cluster	Cross-engine upgrade	Upgrade an Elasticsearch 7.10.2 cluster to an OpenSearch 1.3.6 cluster in CSS.	<ul style="list-style-type: none"> You are advised to perform the upgrade during off-peak hours. Make sure the Elasticsearch cluster does not have ongoing tasks during the upgrade. 	Upgrading the Elasticsearch Cluster Version

1.2 Migrating Data Between Elasticsearch Clusters Using Huawei Cloud Logstash

You can use a Logstash cluster created using Huawei Cloud's Cloud Search Service (CSS) to migrate data between Elasticsearch clusters.

Scenarios

Huawei Cloud Logstash is a fully managed data ingestion and processing service. It is compatible with open-source Logstash and can be used for data migration between Elasticsearch clusters.

You can use Huawei Cloud Logstash to migrate data from Huawei Cloud Elasticsearch, self-built Elasticsearch, or third-party Elasticsearch to Huawei Cloud Elasticsearch. This solution applies to the following scenarios:

- Cross-version migration: Migrate data between different versions to maintain data availability and consistency in the new version, leveraging the compatibility and flexibility of Logstash. This mode applies to migration scenarios where the Elasticsearch cluster version span is large, for example, from 6.X to 7.X.
- Cluster merging: Utilize Logstash to transfer and consolidate data from multiple Elasticsearch clusters into a single Elasticsearch cluster, enabling unified management and analysis of data from different Elasticsearch clusters.
- Cloud migration: Migrate an on-premises Elasticsearch service to the cloud to enjoy the benefits of cloud services, such as scalability, ease-of-maintenance, and cost-effectiveness.
- Changing the service provider: An enterprise currently using a third-party Elasticsearch service wishes to switch to Huawei Cloud for reasons such as cost, performance, or other strategic considerations.

Solution Architecture

Figure 1-1 Migration process

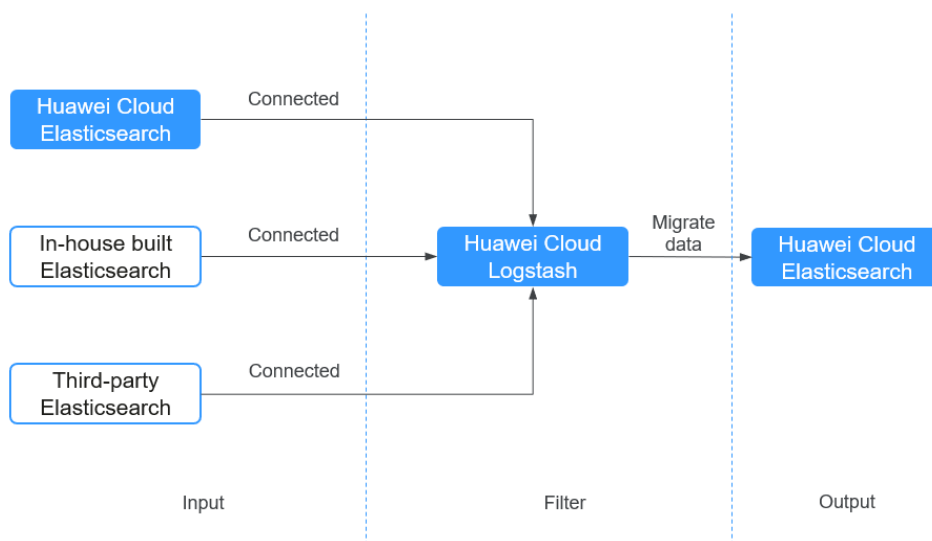


Figure 1-1 shows how to migrate data between Elasticsearch clusters using Huawei Cloud Logstash.

1. Input: Huawei Cloud Logstash receives data from Huawei Cloud Elasticsearch, self-built Elasticsearch, or third-party Elasticsearch.

NOTE

The procedure for migrating data from Huawei Cloud Elasticsearch, self-built Elasticsearch, or third-party Elasticsearch to Huawei Cloud Elasticsearch remains the same. The only difference is the access address of the source cluster. For details, see [Obtaining Elasticsearch Cluster Information](#).

2. Filter: Huawei Cloud Logstash cleanses and converts data.
3. Output: Huawei Cloud Logstash outputs data to the destination system, for example, Huawei Cloud Elasticsearch.

You can choose between full data migration or incremental data migration based on service requirements.

- **Full data migration:** Utilize Logstash to perform a complete data migration. This method is suitable for the initial phase of migration or scenarios where data integrity must be ensured.
- **Incremental data migration:** Configure Logstash to perform incremental queries and migrate only index data with incremental fields. This method is suitable for scenarios requiring continuous data synchronization or real-time data updates.

Advantages

- **Compatibility with later versions:** This solution supports the migration of Elasticsearch clusters across different versions.
- **Efficient data processing:** Logstash supports batch read and write operations, significantly enhancing data migration efficiency.

- Concurrent synchronization technology: The slice concurrent synchronization technology can be utilized to boost data migration speed and performance, especially when handling large volumes of data.
- Simple configuration: Huawei Cloud Logstash offers a straightforward and intuitive configuration process, allowing you to input, process, and output data through configuration files.
- Powerful data processing: Logstash includes various built-in filters to clean, convert, and enrich data during migration.
- Flexible migration policies: You can choose between full migration or incremental migration based on service requirements to optimize storage usage and migration time.

Impact on Performance

Using Logstash for data migration between clusters relies on the Scroll API. This API can efficiently retrieve index data from the source cluster and synchronize the data to the destination cluster in batches. This process may impact the performance of the source cluster. The specific impact depends on how fast data is retrieved from the source cluster, and the data retrieval speed depends on the **size** and **slice** settings of the Scroll API. For details, see the [Reindex API](#) document.

- If the source cluster has a high resource usage, it is advisable to tune the **size** parameter to slow down the data retrieval speed or perform the migration during off-peak hours, reducing impact on the performance of the source cluster.
- If the source cluster has a low resource usage, keep the default settings of the Scroll API. In the meantime, monitor the load of the source cluster. Tune the **size** and **slice** parameters based on the load conditions of the source cluster, optimizing migration efficiency and resource utilization.

Constraints

During cluster migration, do not add, delete, or modify indexes in the source cluster, or the source and destination clusters will have inconsistent data after the migration.

Prerequisites

- The source and destination Elasticsearch clusters are available.
- Ensure that the network between the clusters is connected.
 - If the source cluster, Logstash, and destination cluster are in different VPCs, establish a VPC peering connection between them. For details, see [VPC Peering Connection Overview](#).
 - To migrate an on-premises Elasticsearch cluster to Huawei Cloud, you can configure public network access for the on-premises Elasticsearch cluster.
 - To migrate a third-party Elasticsearch cluster to Huawei Cloud, you need to establish a VPN or Direct Connect connection between the third party's internal data center and Huawei Cloud.
- Ensure that **_source** has been enabled for indexes in the cluster.

By default, `_source` is enabled. You can run the `GET {index}/_search` command to check whether it is enabled. If the returned index information contains `_source`, it is enabled.

Procedure

1. **Obtaining Elasticsearch Cluster Information**
2. **(Optional) Migrating the Index Structure:** Migrate an Elasticsearch cluster's index template and index structure using scripts.
3. **Creating a Logstash Cluster:** Create a Logstash cluster for data migration.
4. **Verifying Connectivity Between Clusters:** Verify the connectivity between the Logstash cluster and the source Elasticsearch cluster.
5. Migrate the source Elasticsearch cluster using Logstash.
 - **Using Logstash to Perform Full Data Migration** is recommended at the initial stage of cluster migration or in scenarios where data integrity needs to be ensured.
 - **Using Logstash to Incrementally Migrate Cluster Data** is recommended for scenarios that require continuous data synchronization or real-time data updates.
6. **Deleting a Logstash Cluster:** After the cluster migration is complete, release the Logstash cluster in a timely manner.

Obtaining Elasticsearch Cluster Information

Before migrating a cluster, you need to obtain necessary cluster information for configuring a migration task.

Table 1-2 Required Elasticsearch cluster information

Cluster Source		Required Information	How to Obtain
Source cluster	Huawei Cloud Elasticsearch cluster	<ul style="list-style-type: none"> • Name of the source cluster • Access address of the source cluster • Username and password for accessing the source cluster (only for security-mode clusters) 	<ul style="list-style-type: none"> • For details about how to obtain the cluster name and access address, see 3. • Contact the service administrator to obtain the username and password.
	Self-built Elasticsearch cluster	<ul style="list-style-type: none"> • Name of the source cluster. • Public network address of the source cluster • Username and password for accessing the source cluster (only for security-mode clusters) 	Contact the service administrator to obtain the information.

Cluster Source		Required Information	How to Obtain
	Third-party Elasticsearch cluster	<ul style="list-style-type: none"> Name of the source cluster. Access address of the source cluster Username and password for accessing the source cluster (only for security-mode clusters) 	Contact the service administrator to obtain the information.
Destination cluster	Huawei Cloud Elasticsearch cluster	<ul style="list-style-type: none"> Access address of the destination cluster Username and password for accessing the destination cluster (only for security-mode clusters) 	<ul style="list-style-type: none"> For details about how to obtain the access address, see 3. Contact the service administrator to obtain the username and password.

The method of obtaining the cluster information varies depending on the source cluster. This section describes how to obtain information about the Huawei Cloud Elasticsearch cluster.

1. Log in to the CSS management console.
2. In the navigation pane on the left, choose **Clusters > Elasticsearch**.
3. In the Elasticsearch cluster list, obtain the cluster name and access address.

Figure 1-2 Obtaining cluster information

Name/ID	Cluster Status	Task Status	Version	Created	Enterprise Project	Private Network Address
css- d6e2a884-d68a-4...	Available		7.10.0	Jul 19, 2024 10:	default	192.168.0.130:9600,192.16.

(Optional) Migrating the Index Structure

If you plan to manually create an index structure in the destination Elasticsearch cluster, skip this section. This section describes how to migrate an Elasticsearch cluster's index template and index structure using scripts.

1. Create an ECS to migrate the metadata of the source cluster.
 - a. Create an ECS. Select CentOS as the OS of the ECS and 2U4G as its flavor. The ECS must be in the same VPC and security group as the CSS cluster.
 - b. Test the connectivity between the ECS and the source and destination clusters.

Run the **curl http://{ip}:{port}** command on the ECS to test the connectivity. If 200 is returned, the connection is successful.

IP indicates the access address of the source or destination cluster. **port** indicates the port number. The default port number is **9200**. Use the actual port number of the cluster.

```
curl http://10.234.73.128:9200 # Enter the actual IP address. A non-security-mode cluster is used here as an example.
{
  "name" : "es_cluster_migrate-ess-esn-1-1",
  "cluster_name" : "es_cluster_migrate",
  "cluster_uuid" : "1VbP7-39QNOx_R-lXKKtA",
  "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. Prepare tools and software. Depending on whether the ECS is connected to the Internet, choose an appropriate method to install them.
 - The VM is connected: use yum and pip to install them. For details, see 3.
 - The VM is disconnected: download the installation package to the VM, and run commands on the VM to install them. For details, see 4.

Table 1-3 Tools and software

Type	Purpose	How to Obtain
Python2	Used to execute data migration scripts.	Python2 . Select Python 2.7.18.
winscp	Cross-platform file transfer tool. Used by Linux to upload scripts.	WinSCP

3. The online installation procedure is as follows:
 - a. Run **yum install python2** to install python2.
[root@ecs opt]# yum install python2
 - b. Run **yum install python-pip** to install pip.
[root@ecs opt]# yum install python-pip
 - c. Run **pip install pyyaml** to install the YAML dependency.
 - d. Run **pip install requests** to install the requests dependency.
4. The offline installation procedure is as follows:
 - a. Download the python2 installation package from <https://www.python.org/downloads/release/python-2718/>. Download and install the source code.

Figure 1-3 Downloading the python2 package

Version	Operating System	Description	MD5 Sum	File Size	GPG
Gzipped source tarball	Source release		38c84292658ed4456157195f1c9bcbe1	17539408	SIG
XZ compressed source tarball	Source release		fd6cc8ec0a78c44036f825e739f36e5a	12854736	SIG
macOS 64-bit installer	Mac OS X	for OS X 10.9 and later	ce98eeb7bdf806685adc265ec1444463	24889285	SIG
Windows debug information files	Windows		20b111ccfe8d06d2fe8c77679a86113d	25178278	SIG
Windows debug information files for 64-bit binaries	Windows		bb0897ea20fda343e5179d413d4a4a7c	26005670	SIG
Windows help file	Windows		b3b753dff1c7930243c1c40ec3a72b1	6322188	SIG
Windows x86-64 MSI installer	Windows	for AMD64/EM64T/x64	a425c758d38f8e28b56f4724b499239a	20598784	SIG
Windows x86 MSI installer	Windows		db6ad9195b3086c6b4cefb9493d738d2	19632128	SIG

- b. Use WinSCP to upload the Python installation package to the **opt** directory and install Python.

```
# Extract the Python package.
[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
...
# Enter the installation directory.
[root@ecs-52bc opt]# cd Python-2.7.18
# Check the file configuration installation path.
[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
# Compile Python.
[root@ecs-52bc Python-2.7.18]# make
# Install Python.
[root@ecs-52bc Python-2.7.18]# make install
```

- c. Check the Python installation result.

```
# Check the Python version.
[root@ecs-52bc Python-2.7.18]# python --version
Python 2.7.5
# Check the pip version.
[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. Prepare the index migration script of the source Elasticsearch cluster.

- a. Run the **vi migrateConfig.yaml** file, modify the following content based on site requirements, and run the **wq** command to save the file as **Logstash migration script**: For details about how to obtain the cluster information, see [Obtaining Elasticsearch Cluster Information](#).

```
es_cluster_new:
# Name of the source cluster
  clustername: es_cluster_new
# Access address of the source cluster, plus http://.
```

```
src_ip: http://x.x.x.x:9200
# Username and password for accessing the source cluster. For a non-security-mode cluster, set
the value to "".
src_username: ""
src_password: ""
# Access address of the destination Elasticsearch cluster, plus http://.
dest_ip: http://x.x.x.x:9200
# Username and password for accessing the destination Elasticsearch cluster. For a non-security-
mode cluster, set the value to "".
dest_username: ""
dest_password: ""
# only_mapping is an optional parameter with a default value of false. It must be used in
conjunction with migrateMapping.py to specify whether to process only the index of the
mapping address in the file. When set to true, only the index data that matches the mapping
key in the source cluster is migrated. When set to false, all index data, except for .kibana and .*,
is migrated from the source cluster.
# During migration, the index name is compared with the provided mapping. If a match is
found, the mapping value is used as the index name in the destination cluster. If no match is
found, the original index name from the source cluster is retained.
only_mapping: false
# Set the index to be migrated. key indicates the index name of the source cluster, and value
indicates the index name of the destination cluster.
mapping:
  test_index_1: test_index_1
# only_compare_index is an optional parameter with a default value of false. It must be used
in conjunction with checkIndices.py. When set to false, both the number of indexes and
documents are compared. When set to true, only the number of indexes is compared.
only_compare_index: false
```

- b. Run the **vi migrateTemplate.py** command. Copy the following script, and run the **wq** command to save it as an **index template migration script**.

```
# -*- 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. Run the **vi migrateMapping.py** command. Copy the following script and run the **wq** command to save it as an **index structure migration script**.

```
# -*- 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. Run the **vi checkIndices.py** command. Copy the following script and run the **wq** command to save it as an **index data comparison script**.

```
# -*- 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. Run the following commands to migrate the index template and index structure of the Elasticsearch cluster:

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

Creating a Logstash Cluster

After the migration environment in the ECS is prepared, create a Logstash cluster in CSS for data migration.

1. Log in to the CSS [management console](#).

2. In the navigation pane on the left, choose **Clusters > Logstash**.
3. Click **Create Cluster** in the upper right corner. The **Create Cluster** page is displayed.
4. On the cluster creation page, configure the cluster as prompted.

Table 1-4 describes the key parameters. Use the default values for other parameters. For details about how to create a cluster, see [Creating a Logstash Cluster](#).

Table 1-4 Key Logstash cluster configurations

Parameter	Description
Billing Mode	Select Pay-per-use . In this billing mode, you are billed by actual duration of use, with a billing cycle of one hour. For example, 58 minutes of usage will be rounded up to an hour and billed.
Cluster Type	Select Logstash .
Version	Choose 7.10.0 .
Name	Cluster name, which contains 4 to 32 characters. Only letters, numbers, hyphens (-), and underscores (_) are allowed and the value must start with a letter. Logstash-ES is an example.
VPC	A VPC is a secure, isolated logical network environment. Select the same VPC as the destination Elasticsearch cluster.
Subnet	A subnet provides dedicated network resources that are isolated from other networks, improving network security. Select the destination subnet. You can access the VPC management console to view the names and IDs of the existing subnets in the VPC.
Security Group	A security group is a collection of access control rules for ECSs that have the same security protection requirements and are mutually trusted in a VPC. Select the same security group as the destination Elasticsearch cluster.

5. Click **Next: Advanced Settings** and retain the default settings.
6. Click **Next: Confirm Configuration**. Confirm the settings, and click **Create Now** to create the cluster.
7. Click **Back to Cluster List** to switch to the **Clusters** page. The cluster you created is now in the cluster list and its status is **Creating**. If the cluster is successfully created, its status changes to **Available**.

Verifying Connectivity Between Clusters

Before starting a migration task, verify the network connectivity between Logstash and the source Elasticsearch cluster.

1. In the Logstash cluster list, select the created Logstash cluster **Logstash-ES** and click **Configuration Center** in the **Operation** column.
2. On the **Configuration Center** page, click **Test Connectivity**.
3. In the dialog box that is displayed, enter the IP address and port number of the source cluster and click **Test**.

Figure 1-4 Testing connectivity



If **Available** is displayed, the network is connected. If the network is disconnected, configure routes for the Logstash cluster to connect the clusters. For details, see [Configuring Routes for a Logstash Cluster](#).

Using Logstash to Perform Full Data Migration

At the initial stage of cluster migration, or in scenarios where guaranteeing data integrity takes top priority, it is recommended to use Logstash for full data migration. This approach migrates the entire Elasticsearch cluster's data in one go.

1. Log in to the CSS management console.
2. In the navigation pane on the left, choose **Clusters > Logstash**.
3. In the Logstash cluster list, select the created Logstash cluster **Logstash-ES** and click **Configuration Center** in the **Operation** column.
4. On the **Configuration Center** page, click **Create** in the upper right corner. On the **Create Configuration File** page, edit the configuration file for the full Elasticsearch cluster migration.
 - a. **Selecting a cluster template:** Expand the system template list, select **elasticsearch**, and click **Apply** in the **Operation** column.
 - b. **Setting the name of the configuration file:** Set **Name**, for example, **es-es-all**.
 - c. **Editing the configuration file:** Enter the migration configuration plan of the Elasticsearch cluster in **Configuration File Content**. The following is an example of the configuration file: For details about how to obtain the cluster information, see [Obtaining Elasticsearch Cluster Information](#).

```
input{
  elasticsearch{
    # Access address of the source Elasticsearch cluster. You do not need to add a protocol. If you
    # add the HTTPS protocol, an error will be reported.
    hosts => ["xx.xx.xx.xx:9200", "xx.xx.xx.xx:9200"]
    # Username and password for accessing the source cluster. You do not need to configure them
    # for a non-security-mode cluster.
    # user => "css_logstash"
```

```

# password => "*****"
# Configure the indexes to be migrated. Use commas (,) to separate multiple indexes. You can
# use wildcard characters, for example, index*.
index => "*_202102"
docinfo => true
slices => 3
size => 3000
# If the destination cluster is accessed through HTTPS, you need to configure the following
# information:
# HTTPS access certificate of the cluster. Retain the following for the CSS cluster:
# ca_file => "/rds/datastore/logstash/v7.10.0/package/logstash-7.10.0/extend/certs" # for
# 7.10.0
# Whether to enable HTTPS communication. Set this parameter to true for a cluster accessed
# through HTTPS.
#ssl => true
}
}

# Remove specified fields added by Logstash.
filter {
  mutate {
    remove_field => ["@version"]
  }
}

output{
  elasticsearch{
# Access address of the destination Elasticsearch cluster
hosts => ["xx.xx.xx.xx:9200","xx.xx.xx.xx:9200"]
# Username and password for accessing the destination cluster. You do not need to configure
# them for a non-security-mode cluster.
# user => "css_logstash"
# password => "*****"
# Configure the index of the target cluster. The following configuration indicates that the index
# name is the same as that of the source end.
index => "%{[@metadata][_index]}"
document_type => "%{[@metadata][_type]}"
document_id => "%{[@metadata][_id]}"
# If the destination cluster is accessed through HTTPS, you need to configure the following
# information:
# HTTPS access certificate of the cluster. Retain the following for the CSS cluster:
# cacert => "/rds/datastore/logstash/v7.10.0/package/logstash-7.10.0/extend/certs" # for
# 7.10.0
# Whether to enable HTTPS communication. Set this parameter to true for a cluster accessed
# through HTTPS.
#ssl => true
# Whether to verify the elasticsearch certificate on the server. Set this parameter to false,
# indicating that the certificate is not verified.
#ssl_certificate_verification => false
}
}
}

```

Table 1-5 Configuration items for full migration

Item		Description
input	hosts	IP address of the source cluster. If the cluster has multiple access nodes, separate them with commas (,).
	user	Username for accessing the cluster. For a non-security-mode cluster, use # to comment out this parameter.

Item		Description
	password	Password for accessing the cluster. For a non-security-mode cluster, use # to comment out this item.
	index	The source indexes to be fully migrated. Use commas (,) to separate multiple indexes. Wildcard is supported, for example, <i>index*</i> .
	docinfo	Indicates whether to re-index the document. The value must be true .
	slices	In some cases, it is possible to improve overall throughput by consuming multiple distinct slices of a query simultaneously using sliced scrolls. It is recommended that the value range from 2 to 8.
	size	Maximum number of hits returned for each query
output	hosts	Access address of the destination cluster. If the cluster has multiple nodes, separate them with commas (,).
	user	Username for accessing the cluster. For a non-security-mode cluster, use # to comment out this parameter.
	password	Password for accessing the cluster. For a non-security-mode cluster, use # to comment out this item.
	index	Name of the index migrated to the destination cluster. It can be modified and expanded, for example, <i>Logstash-%{+yyyy.MM.dd}</i> .
	document_type	Ensure that the document type on the destination end is the same as that on the source end.
	document_id	Document ID in the index. It is advisable to keep consistent document IDs on the source and destination clusters. If you want to have document IDs automatically generated, use the number sign (#) to comment it out.

- d. Click **Next** to configure Logstash pipeline parameters.
In this example, retain the default values. For details about how to set the parameters, see .
- e. Click **OK**.

On the **Configuration Center** page, you can check the created configuration file. If its status changes to **Available**, it has been successfully created.

5. Execute the full migration task.
 - a. In the configuration file list, select configuration file **es-es-all** and click **Start** in the upper left corner.
 - b. In the **Start Logstash** dialog box, select **Keepalive** if necessary. In this example, **Keepalive** is not enabled.

When **Keepalive** is enabled, a daemon process will be configured on each node. If the Logstash service becomes faulty, the daemon process will try to rectify the fault and restart the service. You are advised to enable **Keepalive** for services running long-term. Do not enable it for services running only short-term, or your migration tasks may fail due to a lack of source data.
 - c. Click **OK** to start the configuration file and hence the Logstash full migration task.

You can view the started configuration file in the pipeline list.
6. After data migration is complete, check data consistency.
 - Method 1: Use PuTTY to log in to the VM used for the migration and run the **python checkIndices.py** command to compare the data.
 - Method 2: Run the **GET _cat/indices** command on the Kibana console of the source and destination clusters, separately, to check whether their indexes are consistent.

Using Logstash to Incrementally Migrate Cluster Data

In scenarios where continuous data synchronization or real-time data is required, it is recommended to use Logstash incremental cluster data migration. This method involves configuring incremental queries in Logstash, allowing only index data with incremental fields to be migrated.

1. Log in to the CSS [management console](#).
2. In the navigation pane on the left, choose **Clusters > Logstash**.
3. In the Logstash cluster list, select the created Logstash cluster **Logstash-ES** and click **Configuration Center** in the **Operation** column.
4. On the **Configuration Center** page, click **Create** in the upper right corner. On the **Create Configuration File** page, edit the configuration file for the incremental migration.
 - a. **Selecting a cluster template:** Expand the system template list, select **elasticsearch**, and click **Apply** in the **Operation** column.
 - b. **Setting the name of the configuration file:** Set **Name**, for example, **es-es-inc**.
 - c. **Editing the configuration file:** Enter the migration configuration plan of the Elasticsearch cluster in **Configuration File Content**. The following is an example of the configuration file:

The incremental migration configuration varies according to the index and must be provided based on the index analysis. For details about how to obtain the cluster information, see [Obtaining Elasticsearch Cluster Information](#).

```
input{
  elasticsearch{
    # Access address of the source Elasticsearch cluster. You do not need to add a protocol. If you
    # add the HTTPS protocol, an error will be reported.
    hosts => ["xx.xx.xx.xx:9200"]
    # Username and password for accessing the source cluster. You do not need to configure them
    # for a non-security-mode cluster.
    user => "css_logstash"
    password => "*****"
    # Configure incremental migration indexes.
    index => "*_202102"
    # Configure incremental migration query statements.
    query => '{"query":{"bool":{"should":[{"range":{"postsDate":{"from":"2021-05-25
00:00:00"}}}]}}}'
    docinfo => true
    size => 1000
    # If the destination cluster is accessed through HTTPS, you need to configure the following
    # information:
    # HTTPS access certificate of the cluster. Retain the following for the CSS cluster:
    # ca_file => "/rds/datastore/logstash/v7.10.0/package/logstash-7.10.0/extend/certs" # for
    # 7.10.0
    # Whether to enable HTTPS communication. Set this parameter to true for HTTPS-based cluster
    # access.
    #ssl => true
  }
}

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

output{
  elasticsearch{
    # Access address of the destination cluster.
    hosts => ["xx.xx.xx.xx:9200","xx.xx.xx.xx:9200"]
    # Username and password for accessing the destination cluster. You do not need to configure
    # them for a non-security-mode cluster.
    #user => "admin"
    #password => "*****"
    # Configure the index of the target cluster. The following configuration indicates that the index
    # name is the same as that of the source end.
    index => "%{[@metadata][_index]}"
    document_type => "%{[@metadata][_type]}"
    document_id => "%{[@metadata][_id]}"
    # If the destination cluster is accessed through HTTPS, you need to configure the following
    # information:
    # HTTPS access certificate of the cluster. Retain the default value for the CSS cluster.
    #cacert => "/rds/datastore/logstash/v7.10.0/package/logstash-7.10.0/extend/certs" # for
    # 7.10.0
    # Whether to enable HTTPS communication. Set this parameter to true for HTTPS-based cluster
    # access.
    #ssl => true
    # Whether to verify the elasticsearch certificate on the server. Set this parameter to false,
    # indicating that the certificate is not verified.
    #ssl_certificate_verification => false
  }

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


Table 1-6 Incremental migration configuration items

Configuration	Description
hosts	Access addresses of the source and target clusters. If a cluster has multiple nodes, enter all their access addresses.
user	Username for accessing the cluster. For a non-security-mode cluster, use # to comment out this parameter.
password	Password for accessing the cluster. For a non-security-mode cluster, use # to comment out this item.
index	Indexes to be incrementally migrated. One configuration file supports the incremental migration of only one index.
query	<p>Identifier of incremental data. Typically, it is a DLS statement of Elasticsearch and needs to be analyzed in advance. postsDate indicates the time field in the service.</p> <pre> {"query":{"bool":{"should":[{"range":{"postsDate":{"from":"2021-05-25 00:00:00"}}}]}}} </pre> <p>This command means to migrate data added after 2021-05-25. During multiple incremental migrations, you need to change the log value. If the indexes in the source end Elasticsearch use the timestamp format, convert the data to a timestamp here. The validity of this command must be verified in advance.</p>
scroll	If there is massive data on the source end, you can use the scroll function to obtain data in batches to prevent Logstash memory overflow. The default value is 1m . The interval cannot be too long. Otherwise, data may be lost.

5. Execute the incremental migration task.
 - a. In the configuration file list, select configuration file **es-es-inc** and click **Start** in the upper left corner.
 - b. In the **Start Logstash** dialog box, select **Keepalive** if necessary. In this example, **Keepalive** is not enabled.

When **Keepalive** is enabled, a daemon process will be configured on each node. If the Logstash service becomes faulty, the daemon process will try to rectify the fault and restart the service. You are advised to enable **Keepalive** for services running long-term. Do not enable it for services running only short-term, or your migration tasks may fail due to a lack of source data.

- c. Click **OK** to start the configuration file and hence the Logstash incremental migration task.

You can view the started configuration file in the pipeline list.

6. After data migration is complete, check data consistency.
 - Method 1: Use PuTTY to log in to the VM used for the migration and run the **python checkIndices.py** command to compare the data.
 - Method 2: Run the **GET _cat/indices** command on the Kibana console of the source and destination clusters, separately, to check whether their indexes are consistent.

Deleting a Logstash Cluster

After the migration is complete, release the Logstash cluster in a timely manner to save resources and avoid unnecessary fees.

1. Log in to the CSS [management console](#).
2. In the navigation pane on the left, choose **Clusters > Logstash**.
3. In the Logstash cluster list, select the created Logstash cluster **Logstash-ES** and click **More > Delete** in the **Operation** column. In the confirmation dialog box, manually type in **DELETE**, and click **OK**.

1.3 Migrating Data Between Huawei Cloud Elasticsearch Clusters Using Backup and Restoration

Data can be migrated between CSS Elasticsearch clusters by backing up and restoring cluster snapshots.

Scenario

Data migration between Huawei Cloud Elasticsearch clusters via backup and restoration is applicable solely to scenarios where both the source and destination clusters are CSS clusters and rely on OBS. Typical application scenarios include:

- Cross-region or cross-account migration: Migrate the data of an Elasticsearch cluster in another region or under another account to the current cluster.
- Cross-version migration: Migrate data from a self-built Elasticsearch cluster of an earlier version to a cluster of a later version.
- Cluster merge: Merge the index data of two Elasticsearch clusters.

Overview

Figure 1-5 Process



Figure 1-5 shows the process of migrating data between Huawei Cloud Elasticsearch clusters using backup and restoration.

1. Create a snapshot for the source Elasticsearch cluster and store the snapshot in an OBS bucket.
2. Restore the cluster to a destination cluster using the snapshot store in the OBS bucket.

Advantages

- Easy operation and management: The cluster snapshot function on the CSS console allows for simple, easy-to-manage, and automatic data backup and restoration.
- Applicable to large-scale data migration: Snapshot backup is suitable for scenarios involving large amounts of data, especially when the data volume reaches GB, TB, or even PB levels.
- Cross-region and cross-account migration: With the cross-region replication function of OBS, data can be migrated across different regions and accounts.
- Controllable restoration process: During data restoration, you can restore specific indexes or all indexes and specify the cluster status to be restored.
- Controllable migration duration: The data migration rate can be configured based on the migration duration evaluation formula. Ideally, the data migration rate matches the file replication rate.

Impact on Performance

The essence of migrating data between clusters using backup and restoration is copying files at the data storage layer to back up data. This solution does not rely on any external Elasticsearch APIs. Hence it significantly reduces any impact on the performance of the source cluster. For clusters that are not particularly latency-sensitive, the performance impact of this method can be ignored.

Constraints

- The version of the destination cluster must not be earlier than that of the source cluster. For details, see [Snapshot version compatibility](#).
- The number of nodes in the destination cluster must be greater than half of that in the source cluster, and cannot be less than the number of shard replicas in the source cluster.
- The CPU, memory, and disk capacities of the destination cluster must not be lower than those of the source cluster.

Migration Duration

The number of nodes or index shards in the source and destination clusters determines how long the data migration will take. Data migration consists of two phases: data backup and restoration. The backup duration is determined by the source cluster and the restoration duration is determined by the destination cluster. The formula for calculating the total migration duration is as follows:

- If the number of index shards is greater than the number of nodes:
Total duration (in seconds) = (800 GB ÷ 40 MB ÷ Number of nodes in the source cluster + 800 GB ÷ 40 MB ÷ Number of nodes in the destination cluster) × Number of indexes

- If the number of index shards is smaller than the number of nodes:
Total duration (in seconds) = (800 GB ÷ 40 MB ÷ Number of index shards in the source cluster + 800 GB ÷ 40 MB ÷ Number of index shards in the destination cluster) × Number of indexes

 **NOTE**

The migration duration estimated using the formula is the minimal duration possible (if each node transmits data at the fastest speed, 40 MB/s). The actual duration also depends on factors such as the network and resources condition.

Prerequisites

- The destination cluster (**Es-2**) and source cluster (**Es-1**) are available. You are advised to migrate a cluster during off-peak hours.
- Ensure that the destination cluster (**Es-2**) and source cluster (**Es-1**) are in the same region.

If the cluster is deployed across regions or accounts, refer to the [Configuring Cross-Region Replication](#) section to copy the OBS bucket that stores snapshots in the source cluster to the OBS bucket in the destination cluster. During the migration, restore the data in the destination cluster.

Procedure

1. Log in to the Cloud Search Service management console.
2. Choose **Clusters > Elasticsearch**. On the displayed page, click the source cluster name **Es-1** to go to the basic information page.
3. In the navigation pane, choose **Cluster Snapshots**, and set basic snapshot configurations.

Table 1-7 Basic configurations for a cluster snapshot

Parameter	Description
OBS Bucket	Select an OBS bucket for storing cluster snapshots.
Backup Path	Storage path of the cluster snapshot in the OBS bucket. You can retain the default value.

Parameter	Description
IAM Agency	<p>To store snapshot data to an OBS bucket, you must have the required OBS access permissions. Select an IAM agency to grant the current account the permission to access and use OBS.</p> <ul style="list-style-type: none"> • If you are configuring an agency for the first time, click Automatically Create IAM Agency to create css-obs-agency. • If there is an IAM agency automatically created earlier, you can click One-click authorization to have the OBS Administrator permissions deleted automatically, and have the following custom policies added automatically instead to implement more refined permissions control. <pre> "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" </pre> • When OBS buckets use SSE-KMS encryption, the IAM agency must be granted KMS permissions. You can click Automatically Create IAM Agency and One-click authorization to have the following custom policies created automatically. <pre> "kms:cmk:create", "kms:dek:create", "kms:cmk:get", "kms:dek:decrypt", "kms:cmk:list" </pre> • To use Automatically Create IAM Agency and One-click authorization, the following minimum permissions are required: <pre> "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" </pre> • To use an IAM agency, the following minimum permissions are required: <pre> "iam:agencies:listAgencies", "iam:agencies:getAgency", "iam:permissions:listRolesForAgencyOnProject", "iam:permissions:listRolesForAgency" </pre>

4. Click **Create**. In the dialog box that is displayed, configure the parameters and click **OK** to manually create a snapshot.

Table 1-8 Snapshot creation parameters

Parameter	Description
Snapshot Name	User-defined snapshot name. You can retain the default value.
Index	Indexes that you want to back up using snapshots. The index names cannot contain spaces or uppercase letters, and cannot contain " <code>< >/?</code> ". Use commas (,) to separate different index names. If you do not specify this parameter, all indexes in the cluster are backed up by default. You can use the asterisk (*) to match multiple indexes. For example, if you enter index* , then the data of all indexes whose names are prefixed with index will be backed up. NOTE Run the GET /_cat/indices command in Kibana to query the names of all indexes in the cluster.
Description	Snapshot description.

In the snapshot management list, if the snapshot status is **Available**, the snapshot has been created.

- In the snapshot management list, click **Restore** in the **Operation** column of the snapshot and configure restoration parameters to restore data to destination cluster **Es-2**.

Table 1-9 Snapshot restoration parameters

Parameter	Description
Index	Enter the name of the index to be restored. If this parameter is not specified, all index data will be restored. You can use the asterisk (*) to match multiple indexes. For example, index* indicates that all indexes with the prefix index in snapshots are restored.
Rename Pattern	Index name matching rule. The Rename Pattern and Rename Replacement take effect only when they are configured at the same time. You can configure them to rename matched indexes in snapshots.
Rename Replacement	Rule for renaming an index name. The Rename Pattern and Rename Replacement take effect only when they are configured at the same time. The default value restored_index_\$1 indicates that restored_ is added in front of the names of all restored indexes.

Parameter	Description
Cluster	Select a destination cluster, for example, Es-2 . Select or deselect Overwrite same-name indexes in the destination cluster . By default, it is deselected. Data restoration using snapshots works by overwriting existing snapshot files. When there are same-name indexes in the destination cluster, you need to select this option in order to restore same-name, same-shard structure indexes. Indexes with a different shard structure cannot be restored. Exercise caution when performing this operation.

In the snapshot list, when the **Task Status** changes to **Restoration succeeded**, the data migration is complete.

6. After the data migration is complete, check the data consistency between the destination Elasticsearch cluster **Es-2** and source Elasticsearch cluster **Es-1**. For example, run the `_cat/indices` command in the source and destination clusters, separately, to check whether their indexes are consistent.

1.4 Migrating Data from an On-premises Elasticsearch Cluster to Huawei Cloud Using the S3 Plugin

This topic describes how to use the S3 plugin to migrate data from an on-premises Elasticsearch cluster to a Huawei Cloud Elasticsearch cluster using the snapshot mechanism.

Scenario

The S3 plugin (repository-s3) is purpose-developed for Elasticsearch. It allows users to store Elasticsearch snapshots to a storage service compatible with S3 APIs, such as Huawei Cloud OBS. The S3 plugin provides an efficient, flexible, and secure way to back up the data of Elasticsearch clusters.

The S3 plugin can be used to migrate data from an on-premises Elasticsearch cluster to a Huawei Cloud Elasticsearch cluster. Typical scenarios include:

- Cloud migration: Migrate an on-premises Elasticsearch service to the cloud to enjoy the benefits of cloud services, such as scalability, ease-of-maintenance, and cost-effectiveness.
- Cross-version migration: Migrate data from an on-premises Elasticsearch cluster of an earlier version to a Huawei Cloud Elasticsearch cluster of a later version.
- Cluster merge: Migrate data from multiple on-premises Elasticsearch clusters to a single Huawei Cloud Elasticsearch cluster to form a unified data platform for simplified management and higher data consistency.
- Unified technology stack: For a simplified, unified technology stack, companies that are already using some services on Huawei Cloud may choose to migrate their on-premises Elasticsearch clusters to Huawei Cloud.

Overview

Figure 1-6 Migration procedure

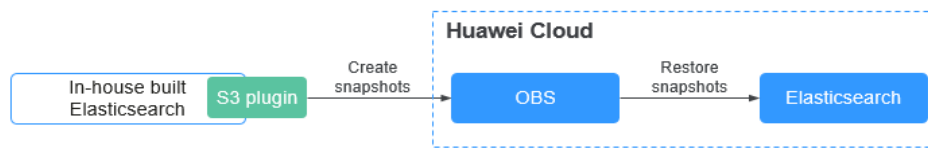


Figure 1-6 describes the procedure for migrating an on-premises Elasticsearch cluster (source cluster) to a Huawei Cloud Elasticsearch cluster (destination cluster) using the S3 plugin.

1. Install the repository-s3 plugin in the on-premises Elasticsearch cluster.
2. Back up the data of the on-premises Elasticsearch cluster to Huawei Cloud OBS.
3. Restore data from Huawei OBS to an Elasticsearch cluster in CSS.

Advantages

- High cross-version compatibility: Data can be migrated between Elasticsearch clusters of different versions, including upgrade from an earlier version to a later version.
- High availability and durability: S3 protects the security of data backups and minimizes the risk of data loss.
- Flexible backup policies: You can choose between full backup or incremental backup based on service requirements to optimize storage utilization and migration time.

Impact on Performance

The essence of migrating data between clusters using backup and restoration is copying files at the data storage layer to back up data. This solution does not rely on any external Elasticsearch APIs. Hence it significantly reduces any impact on the performance of the source cluster. For clusters that are not particularly latency-sensitive, the performance impact of this method can be ignored.

Constraints

- The version of the destination cluster must not be earlier than that of the source cluster. For details, see [Snapshot version compatibility](#).
- This method does not support incremental data synchronization. You need to pause data update before starting to back up data.
- An on-premises Elasticsearch cluster needs public network access to back up snapshots to OBS.

Prerequisites

- The source and destination Elasticsearch clusters are available.
- The OBS bucket (backup-obs) for storing snapshots has been prepared. The OBS bucket and the destination Elasticsearch cluster of CSS are in the same region.

- You have obtained the AK/SK of your account. For details, see [How Do I Obtain an Access Key \(AK/SK\)?](#).

Procedure

Step 1 Install the repository-s3 plugin in the on-premises Elasticsearch cluster.

1. Check the version of the on-premises Elasticsearch cluster and determine the version of the repository-s3 plugin that you need to install.

Run the following command to obtain cluster information:

```
curl http://x.x.x.x:9200 # Enter the IP address of the Elasticsearch cluster.
```

Find **version.number** in the command output. In the example below, the cluster version is 7.10.2.

```
{
  "name" : "es_cluster_migrate-ess-esn-1-1",
  "cluster_name" : "es_cluster_migrate",
  "cluster_uuid" : "1VbP7-39QNOx_R-lXKkTA",
  "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. Download the repository-s3 plugin installation package of the required version. 7.10.2 is used as an example here. Change the version number in the download address based on the actual cluster version.

Download address: <https://artifacts.elastic.co/downloads/elasticsearch-plugins/repository-s3/repository-s3-7.10.2.zip>

3. Log in to an instance node of the on-premises Elasticsearch cluster, and run the **cd** command to go to the plugins directory.
4. Run the following command to extract the repository-s3 plugin installation package to the plugins directory:
5. Add the AK/SK that enables access to Huawei Cloud to the repository-s3 plugin.

- a. Go to the installation path of the on-premises Elasticsearch cluster, and run the following command to create a keystore file:

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

NOTE

Use a common user account, rather than the root account, to create the keystore file. Otherwise, the configuration will not take effect.

- b. Add the AK/SK that enables access to Huawei Cloud to the keystore file.

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

access_key indicates the AK, and **secret_key** indicates the SK.

6. If the on-premises Elasticsearch cluster has multiple instance nodes, repeat [Step 1.3](#) to [Step 1.5](#) to install the S3 plugin and configure the AK/SK for all these nodes.
7. After the AK/SK is added, restart the Elasticsearch cluster for the configuration to take effect.
8. After the restart, run the following command to check whether the S3 plugin has been successfully installed.

```
curl http://x.x.x.x:9200/_cat/plugins # Enter the IP address of the Elasticsearch cluster.
```

If the correct version information of the repository-s3 plugin is returned, as shown below, the plugin has been successfully installed.

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

Step 2 Back up the data of the on-premises Elasticsearch cluster to Huawei Cloud OBS.

1. Log in to the Kibana console of the on-premises Elasticsearch cluster, and choose **Dev Tools** to go to the CLI.
2. Run the following command to create a snapshot repository **my_backup**, and connect it to OBS bucket **backup-obs** for storing snapshots in Huawei Cloud OBS.

```
PUT /_snapshot/my_backup
{
  "type": "s3",
  "settings": {
    # OBS bucket name
    "bucket": "backup-obs",
    # Public network address for accessing OBS
    "endpoint": "obs.xxx.example.com",
    "base_path": "snapshot",
    "max_snapshot_bytes_per_sec": "1000mb"
  }
}
```

3. Run the following command. If correct information about the snapshot repository is returned, it has been successfully created.

```
GET /_snapshot/my_backup
```

4. Run the following command to create a snapshot.

Exclude system indexes whose name starts with `.`, and create a full data snapshot for all other indexes. Setting **wait_for_completion** to **true** means asynchronous task execution.

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

5. After the snapshot is created, run the following command to check the snapshot information:

```
# Display all existing snapshots.
GET _cat/repositories
# Display information about all snapshots.
GET _snapshot/my_backup/_all
# Check the information of a snapshot by specifying its name. snapshot_all indicates the snapshot name.
GET _snapshot/my_backup/snapshot_all
# Check the status of a snapshot by specifying its name. snapshot_all indicates the snapshot name.
GET _snapshot/my_backup/snapshot_all/_status
```

Step 3 Restore data from OBS to an Elasticsearch cluster of CSS.

1. Log in to the CSS management console.
2. In the navigation pane on the left, choose **Clusters > Elasticsearch** to go to the **Clusters** page.

3. Locate the destination cluster and click **Access Kibana** in the **Operation** column to log in to Kibana.
4. Click **Dev Tools** in the navigation tree on the left.
5. Run the following command to create a snapshot repository called **my_backup_all**.

```
PUT _snapshot/my_backup_all/
{
  "type": "obs",
  "settings": {
    # Private domain name of OBS.
    "endpoint": "obs.xxx.example.com",
    "region": "cn-south-1",
    # Username and password for accessing OBS.
    "access_key": "xxx",
    "secret_key": "xxx",
    # OBS bucket name, which must be the same as the destination OBS bucket name in the
    previous step.
    "bucket": "backup-obs",
    "compress": "false",
    "chunk_size": "1g",
    "base_path": "snapshot",
    "max_restore_bytes_per_sec": "1000mb",
    "max_snapshot_bytes_per_sec": "1000mb"
  }
}
```

6. Run the following command to restore data from the snapshot on OBS to the destination cluster.

- Check all existing snapshots. In the command below, **my_backup_all** indicates the repository name.

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

- Restore data from the **snapshot_all** snapshot in the **my_backup_all** snapshot repository to the destination Elasticsearch cluster.

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

- Restore specified indexes to the destination Elasticsearch cluster. In this example, **my_index1** and **my_index2** are the names of the indexes to be restored.

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

----End

1.5 Migrating Data from a Third-Party Elasticsearch Cluster to Huawei Cloud Using Backup and Restoration

You can use backup and restoration to migrate data from a third-party Elasticsearch cluster to a Huawei Cloud Elasticsearch cluster.

Scenario

Data migration between a third-party Elasticsearch cluster and a Huawei Cloud Elasticsearch cluster through backup and restoration depends on storage repositories. Typical application scenarios include:

- Changing the service provider: An enterprise currently using a third-party Elasticsearch service wishes to switch to Huawei Cloud for reasons such as cost, performance, or other strategic considerations.
- Cluster merge: Data scattered across different third-party Elasticsearch clusters can be migrated to Huawei Cloud Elasticsearch clusters for centralized management, enabling more efficient data analysis and querying.
- Cross-version migration: Data is migrated from a third-party Elasticsearch cluster of an earlier version to a Huawei Cloud Elasticsearch cluster of a later version.
- Unified technology stack: For a simplified, unified technology stack, companies that are already using some services on Huawei Cloud may choose to migrate their in-house built Elasticsearch clusters to Huawei Cloud.

Overview

Figure 1-7 Migration procedure

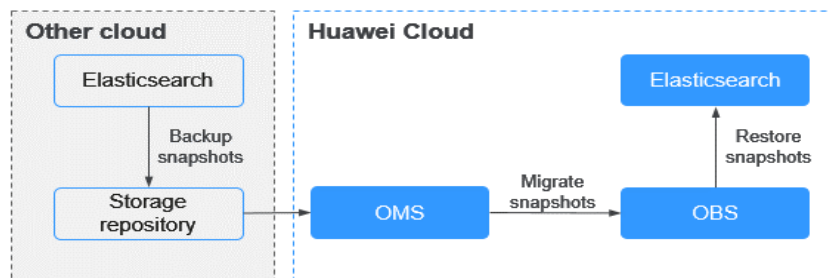


Figure 1-7 shows the process of migrating a third-party Elasticsearch cluster (source cluster) to Huawei Cloud Elasticsearch cluster (destination cluster).

1. Back up third-party Elasticsearch data to a third-party shared repository.
2. Use OMS to migrate data from the shared repository to Huawei Cloud OBS.
3. Use Huawei Cloud OBS to restore data to a Huawei Cloud Elasticsearch cluster.

Advantages

- Easy to use and manage: The snapshot and restoration APIs provided by Elasticsearch are easy to use and manage, and can be used to automate some of the tasks.
- Suitable for large-scale data migration: Snapshots can be used to migrate GB, TB, or even PB of data.
- Controllable restoration process: During data restoration, you can restore specific indexes or all indexes and specify the cluster status to be restored.

Impact on Performance

The essence of migrating data between clusters using backup and restoration is copying files at the data storage layer to back up data. This solution does not rely on any external Elasticsearch APIs. Hence it significantly reduces any impact on

the performance of the source cluster. For clusters that are not particularly latency-sensitive, the performance impact of this method can be ignored.

Constraints

- The version of the destination cluster must not be earlier than that of the source cluster. For details, see [Snapshot version compatibility](#).
- This method does not support incremental data synchronization. You need to pause data update before starting to back up data.
- Snapshots can be migrated only when public access is enabled for third-party storage repositories.

Prerequisites

- The source and destination Elasticsearch clusters are available.
- The OBS bucket **esbak** for storing snapshots has been prepared. The OBS bucket and the destination Elasticsearch cluster of CSS are in the same region.

Procedure

Step 1 Log in to the third-party cloud where Elasticsearch is located and create a shared repository that supports the S3 protocol. For example, log in to Alibaba Cloud and go to the OSS service to create the **patent-esbak** directory, or log in to Tencent Cloud and go to the COS service to create the **patent-esbak** directory.

Step 2 Create a snapshot backup repository in the third-party Elasticsearch cluster to store Elasticsearch snapshot data.

For example, create a backup repository named **my_backup** in Elasticsearch and associate it with the OSS repository.

```
PUT _snapshot/my_backup
{
  # Repository type.
  "type": "oss",
  "settings": {
    # # Private network domain name of the repository in step 1.
    "endpoint": "http://oss-xxx.example.com",
    # User ID and password of the repository. Hard-coded or plaintext access keys (AK/SK) are risky. For
    # security purposes, encrypt your access keys and store them in the configuration file or environment
    # variables. In this example, access keys are stored in the environment variables for identity authentication.
    # Before running the code in this example, configure the AK and SK in environment variables.
    "access_key_id": "ak",
    "secret_access_key": "sk",
    # Bucket name of the repository created in step 1.
    "bucket": "patent-esbak",
    # # Whether to enable snapshot file compression.
    "compress": false,
    # If the size of the uploaded snapshot data exceeds the value of this parameter, the data will be
    # uploaded as blocks to the repository.
    "chunk_size": "1g",
    # Start position of the repository. The default value is the root directory.
    "base_path": "snapshot/"
  }
}
```

Step 3 Create a snapshot in the third-party Elasticsearch cluster.

- Create a snapshot for all indexes.
For example, create a snapshot named **snapshot_1**.

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

- Create a snapshot for specified indexes.

For example, create a snapshot named **snapshot_test** that contains indexes **patent_analyse** and **patent**.

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

Step 4 View the snapshot creation progress in the third-party Elasticsearch cluster.

- Run the following command to view information about all snapshots:

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

- Run the following command to view information about **snapshot_1**:

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

Step 5 Use the Object Storage Migration Service (OMS) to migrate the snapshot data from the repository to the OBS bucket **esbak**.

OMS supports data migration from multiple cloud vendors to OBS. For details, see [Migration from Other Clouds to Huawei Cloud](#).

 **NOTE**

When creating a migration task on OMS, set **Object Metadata** to **Migrate**. Otherwise, data migration may be abnormal.

Step 6 Create a repository in the Elasticsearch cluster of CSS and associate it with OBS. This repository will be used for restoring the snapshot data of the third-party Elasticsearch cluster.

For example, create a repository named **my_backup_all** in the CSS Elasticsearch cluster and associate it with the OBS bucket **esbak**.

```
PUT _snapshot/my_backup_all/
{
  "type" : "obs",
  "settings" : {
    # Private network domain name of OBS
    "endpoint" : "obs.xxx.example.com",
    "region" : "xxx",
    # Username and password for accessing OBS. Hard-coded or plaintext access keys (AK/SK) are risky.
    # For security purposes, encrypt your access keys and store them in the configuration file or environment
    # variables. In this example, access keys are stored in the environment variables for identity authentication.
    # Before running the code in this example, configure the AK and SK in environment variables.
    "access_key": "ak",
    "secret_key": "sk",
    # OBS bucket name, which must be the same as the destination OBS bucket name in the previous
    # step.
    "bucket" : "esbak",
    "compress" : "false",
    "chunk_size" : "1g",
    #Note that there is no slash (/) after snapshot.
    "base_path" : "snapshot",
    "max_restore_bytes_per_sec": "100mb",
    "max_snapshot_bytes_per_sec": "100mb"
  }
}
```

Step 7 Restore the snapshot data to the Elasticsearch cluster of CSS.

1. Check information about all snapshots.

```
GET _snapshot
```

2. Restore data using snapshots.

- Restore all the indexes from a snapshot. For example, to restore all the indexes from **snapshot_1**, run the following command:

```
POST _snapshot/my_backup_all/snapshot_1/_restore?wait_for_completion=true
```

- Restores some indexes from a snapshot. For example, in the snapshot named **snapshot_1**, restore only the indexes that do not start with a period (.).

```
POST _snapshot/my_backup/snapshot_1/_restore
{"indices": "*", "-monitoring*", "-security*", "-kibana*", "ignore_unavailable": "true"}
```

- Restore a specified index from a snapshot and renames the index. For example, in **snapshot_1**, restore **index_1** to **restored_index_1** and **index_2** to **restored_index_2**.

```
POST /_snapshot/my_backup/snapshot_1/_restore
{
  # Restore only indexes index_1 and index_2 and ignore other indexes in the snapshot.
  "indices": "index_1,index_2"
  # Search for the index that is being restored. The index name must match the provided
  # template.
  "rename_pattern": "index_(.+)",
  # Rename the found index.
  "rename_replacement": "restored_index_$1"
}
```

Step 8 View the snapshot restoration result.

- Run the following command to view the restoration results of all snapshots:

```
GET /_recovery/
```

- Run the following command to check the snapshot restoration result of a specified index:

```
GET {index_name}/_recovery
```

----End

1.6 Migrating Data Between Huawei Cloud Elasticsearch Clusters Using the Read/Write Splitting Plugin

This topic describes how to migrate data between Huawei Cloud Elasticsearch clusters using the read/write splitting plugin.

Scenario

By default, the read/write splitting plugin of CSS is installed in Elasticsearch 7.6.2 and Elasticsearch 7.10.2 clusters. You can configure read/write splitting to perform near-real-time synchronization of index data between Elasticsearch clusters.

The read/write splitting plugin can be used for data migration only if both the source and destination clusters were created in CSS. Typical application scenarios include:

- Cross-region or cross-account migration: Migrate the data of an Elasticsearch cluster in another region or under another account to the current cluster.
- Cluster merge: Merge the index data of two Elasticsearch clusters.

Overview

Figure 1-8 Migration procedure

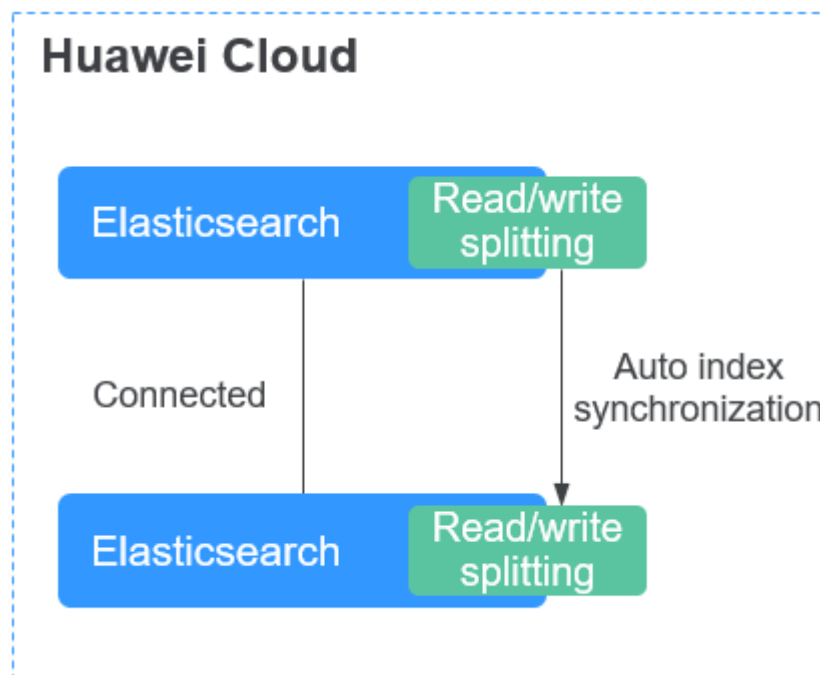


Figure 1-8 illustrates the general procedure for migrating data from one Huawei Cloud Elasticsearch cluster to another using the read/write splitting plugin of CSS.

1. Use the read/write splitting plugin to set up a connection between the source (primary) and destination (secondary) clusters.
2. Configure automatic index synchronization in the destination cluster to have data automatically synchronized from the source to the destination cluster. The synchronization interval is 30 seconds by default and can be changed.
3. Check the status of auto synchronization to see if data migration has completed.

For more information about the read/write splitting feature of CSS, see [Configuring Read/Write Splitting](#).

Advantages

- High data consistency: The read/write splitting feature enables data replication between a pair of primary/secondary clusters and ensures data synchronization between different shards.
- Fast migration: The speed of data synchronization depends on the available bandwidth, not the source or destination cluster.
- Configurable synchronization interval: The default data synchronization interval is 30 seconds. You can change it to reduce the delay of data synchronization.

Impact on Performance

When you use the read/write splitting plugin to migrate data between clusters, you are essentially just synchronizing data by copying files at the bottom layer.

This method does not rely on any external Elasticsearch APIs. Hence any impact on the performance of the source cluster is minimized.

Constraints

The versions of the source and destination clusters must both be 7.6.2 or 7.10.2.

Prerequisites

- The source and destination Elasticsearch clusters are available and both have the read/write splitting plugin installed. (By default, the read/write splitting plugin is installed in Elasticsearch 7.6.2 and 7.10.2.)
- The network between the clusters is connected.

If the source and destination clusters are in different VPCs, establish a VPC peering connection between them. For details, see [VPC Peering Connection Overview](#).

Procedure

- Step 1** Obtain information about the Elasticsearch clusters to configure the data migration task.

Table 1-10 Required Elasticsearch cluster information

Cluster		Required Information	How to Obtain
Source cluster (primary cluster)	Huawei Cloud Elasticsearch cluster	<ul style="list-style-type: none"> • Access address of the source cluster • Username and password for accessing the source cluster (only for security-mode clusters) 	<ul style="list-style-type: none"> • For details about how to obtain the cluster address, see Step 1.3. • Contact the service administrator to obtain the username and password.
	In-house built Elasticsearch cluster	<ul style="list-style-type: none"> • Public network address of the source cluster • Username and password for accessing the source cluster (only for security-mode clusters) 	Contact the service administrator to obtain the information.
	Third-party Elasticsearch cluster	<ul style="list-style-type: none"> • Access address of the source cluster • Username and password for accessing the source cluster (only for security-mode clusters) 	Contact the service administrator to obtain the information.

Cluster		Required Information	How to Obtain
Destination cluster (secondary cluster)	Huawei Cloud Elasticsearch cluster	<ul style="list-style-type: none"> Access address of the destination cluster Username and password for accessing the destination cluster (only for security-mode clusters) 	<ul style="list-style-type: none"> For details about how to obtain the access address, see Step 1.3. Contact the service administrator to obtain the username and password.

1. Log in to the CSS management console.
2. In the navigation pane on the left, choose **Clusters > Elasticsearch**.
3. In the Elasticsearch cluster list, obtain the cluster access address.

Figure 1-9 Obtaining cluster information

Name/ID	Cluster Status	Task Status	Version	Created	Enterprise Project	Private Network Address
css-d6e2a884-d86a-4...	Available		7.10.0	Jul 19, 2024 10:	default	192.168.0.130:9600,192.16...

Step 2 Log in to the Kibana console of the destination Elasticsearch cluster.

1. In the Elasticsearch cluster list, select the destination cluster, and click **Access Kibana** in the **Operation** column to log in to the Kibana console.
2. Click **Dev Tools** in the navigation tree on the left.

Step 3 Use the read/write splitting plugin to set up a connection between the source and destination clusters.

Run the following command to configure information about the source cluster in the destination cluster:

```
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": "*****"
        }
      }
    }
  }
}
```

Table 1-11 Request body parameters

Parameter	Description
<i>leader1</i>	Name of the configuration task, which is user-defined and will be used for configuring read/write splitting later.
seeds	Access address of the source cluster. When HTTPS is enabled for the cluster, the URI schema must use HTTPS.
username	Username of the source cluster (primary cluster). This parameter is required only when security mode is enabled for the primary cluster.
password	Password of the source cluster (primary cluster). This parameter is required only when security mode is enabled for the primary cluster.

If the value of **acknowledged** is **true** in the command output, the configuration is successful.

Step 4 Configure automatic index synchronization in the destination cluster to have data automatically synchronized from the source to the destination cluster.

Run the following command in the destination cluster to create a pattern-matching index synchronization policy, which synchronizes matched indexes from the source cluster to the destination cluster.

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

Table 1-12 Request body parameters

Parameter	Description
pattern1	Name of the pattern for index matching.
remote_cluster	Configuration task name, for example, leader1 in the previous step.
remote_index_patterns	Pattern for matching indexes to be synchronized in the source cluster. The wildcard (*) is supported.
local_index_pattern	Index pattern in the destination cluster. The index template can be replaced. For example, if this parameter is set to {{remote_index}} , the index log1 remains after synchronization.

Parameter	Description
apply_exist_index	Whether to synchronize existing indexes in the source cluster. The default value is true .

Step 5 Check the status of automatic synchronization in the destination cluster. The default synchronization interval is 30 seconds.

Run the following command to obtain the synchronization status of the matched indexes:

```
GET auto_sync/stats
```

If the value of **failed_count** is **0** in the command output, the synchronization is complete.

Step 6 After the synchronization is complete, check the data consistency between the source and destination Elasticsearch clusters.

For example, access the Kibana console of the source and destination clusters separately, go to **Dev Tools**, and run the **GET cat/indices** command to check whether their indexes are consistent.

----End

1.7 Migrating Data Between Elasticsearch Clusters Using the Reindex API

You can use the reindex API to migrate data between Elasticsearch clusters.

Scenario

As an open-source search engine, Elasticsearch provides the reindex API to support index migration between clusters. This API is also provided in CSS to support data migration between Elasticsearch clusters. Below are some scenarios where you might use the reindex API for data migration.

- Cluster merging: The reindex API can be used to merge index data scattered across multiple Elasticsearch clusters into a single cluster for centralized data management and analysis.
- Cloud migration: Migrate an on-premises Elasticsearch service to the cloud to enjoy the benefits of cloud services, such as scalability, ease-of-maintenance, and cost-effectiveness.
- Changing the service provider: If you are using a third-party Elasticsearch service, and you want to switch to Huawei Cloud or another service provider for some reason (cost, performance, or other concerns), you can use the reindex API for data migration.

The reindex API supports the following:

- Full migration: Migrate the full amount of index data between clusters. During the migration, all writes to the source cluster must be stopped, ensuring data consistency between the source and destination clusters.
- Incremental migration: For indexes that have a timestamp field, the reindex API can be used to execute an incremental migration based on this field.

During the workload switchover phase, after the full migration is completed, you must stop all writes to the source cluster, and then use the reindex API to execute a quick incremental migration based on the latest update time. Then you can finally switch all services to the destination cluster.

- Reorganizing indexes: The reindex API can be used to restructure indexes while migrating them, including changing mappings, analyzers, and sharding.

Overview

Figure 1-10 Migration procedure

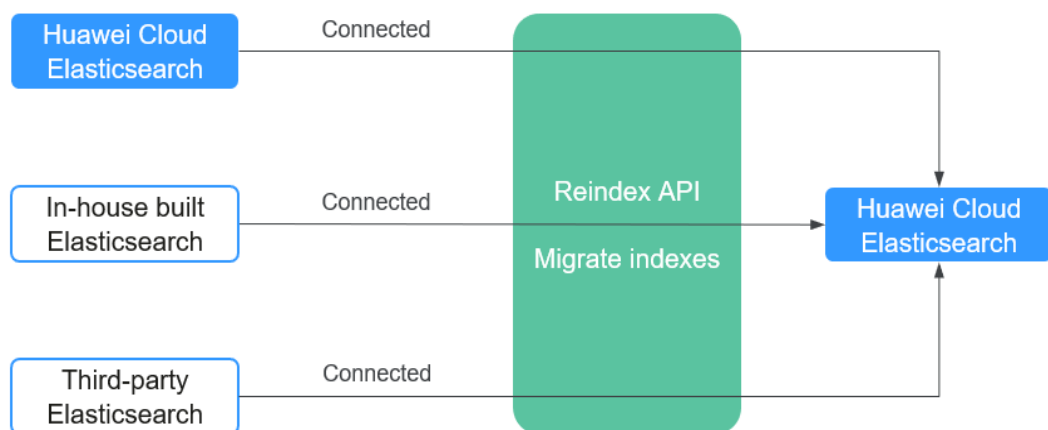


Figure 1-10 shows how to migrate data between Elasticsearch clusters using the reindex API.

1. Configure the reindex remote access whitelist in the destination cluster to connect the source and destination clusters.
2. Use the reindex API to migrate indexes from the source cluster to the destination cluster.

Advantages

- Easy to use: As a built-in function of Elasticsearch, the reindex API offers an easy way to migrate data without complex settings or additional tools.
- Flexible data processing: Indexes can be restructured or rebuilt during migration, such as changing mappings, analyzers, and sharding.
- Performance control: During the migration, you can tune the parameters of the scroll API to control the data migration speed for optimal cluster performance.

Impact on Performance

Using the reindex API for data migration between clusters relies on the scroll API. The scroll API can efficiently retrieve index data from the source cluster and synchronize the data to the destination cluster in batches. This process may impact the performance of the source cluster. The specific impact depends on how fast data is retrieved from the source cluster, and the data retrieval speed depends on the **size** and **slice** settings of the scroll API. For details, see the [Reindex API](#) document.

Reindex tasks are asynchronous in Elasticsearch clusters, so their impact on the performance of the source cluster is manageable when task concurrency is low. If the source cluster has a high resource usage, it is advisable to tune the size parameter of the scroll API to slow down the data retrieval speed or perform the migration during off-peak hours, reducing impact on the performance of the source cluster.

Constraints

- During cluster migration, do not add, delete, or modify the index data of the source cluster. Otherwise, the data in the source cluster will be inconsistent with that in the destination cluster after the migration.
- The source and destination clusters must use the same version.

Prerequisites

- The source and destination Elasticsearch clusters are available.
- The network between the clusters is connected.
 - If the source and destination clusters are in different VPCs, establish a VPC peering connection between them. For details, see [VPC Peering Connection Overview](#).
 - To migrate an in-house built Elasticsearch cluster to Huawei Cloud, you can configure public network access for this cluster.
 - To migrate a third-party Elasticsearch cluster to Huawei Cloud, you need to establish a VPN or Direct Connect connection between the third party's internal data center and Huawei Cloud.
- Ensure that `_source` has been enabled for indexes in the cluster.
By default, `_source` is enabled. You can run the `GET {index}/_search` command to check whether it is enabled. If the returned index information contains `_source`, it is enabled.

Obtaining Information About the Source Elasticsearch Cluster

Before data migration, obtain necessary information about the source cluster for configuring a migration task.

Table 1-13 Required information about the source Elasticsearch cluster

Cluster Type	Required Information	How to Obtain
Huawei Cloud Elasticsearch cluster	<ul style="list-style-type: none"> • Access address of the source cluster • Username and password for accessing the source cluster (only for security-mode clusters) • Index structure 	<ul style="list-style-type: none"> • For details about how to obtain the cluster name and address, see 3. • Contact the service administrator to obtain the username and password. • For details about how to query the index structure, see 6.

Cluster Type	Required Information	How to Obtain
In-house built Elasticsearch cluster	<ul style="list-style-type: none"> Public network address of the source cluster Username and password for accessing the source cluster (only for security-mode clusters) Index structure 	Contact the service administrator to obtain the information.
Third-party Elasticsearch cluster	<ul style="list-style-type: none"> Access address of the source cluster Username and password for accessing the source cluster (only for security-mode clusters) Index structure 	Contact the service administrator to obtain the information.

The method of obtaining the cluster information varies depending on the source cluster. This section describes how to obtain information about a Huawei Cloud Elasticsearch cluster.

1. Log in to the CSS management console.
2. In the navigation pane on the left, choose **Clusters > Elasticsearch**.
3. In the Elasticsearch cluster list, obtain the cluster name and address.

Figure 1-11 Obtaining cluster information

Name/ID	Cluster Status	Task Status	Version	Created	Enterprise Project	Private Network Address
css-d6e2a884-d68a-4...	Available		7.10.0	Jul 19, 2024 10:...	default	192.168.0.130-9600,192.16...

4. Click **Access Kibana** in the **Operation** column to log in to the Kibana console.
5. Click **Dev Tools** in the navigation tree on the left.
6. Run the following command to query the index structure of the source cluster:

```
GET {index_name}
```

index_name indicates the name of the index to be migrated.

Configuring the Reindex Remote Access Whitelist

In the destination Elasticsearch cluster, configure the reindex whitelist.

1. Log in to the CSS management console.
2. In the navigation pane on the left, choose **Clusters > Elasticsearch**.
3. In the Elasticsearch cluster list, click the destination cluster. The cluster information page is displayed.
4. In the navigation pane on the left, click **Parameter Configurations**. Then click **Edit**, and expand **Reindexing**.

- Set **reindex.remote.whitelist**.
- Value: Enter the address of the source cluster obtained in [Obtaining Information About the Source Elasticsearch Cluster](#).

If the source Elasticsearch cluster uses HTTPS, expand **Customize**, and add a custom parameter to ignore SSL authentication.

- Parameter: `reindex.ssl.verification_mode`
 - Value: `none`
5. Click **Submit**. In the displayed confirmation dialog box, confirm the parameter settings, select **I understand that the modification will take effect after the cluster is restarted**, then click **OK**.
 6. Return to the Elasticsearch cluster list, and locate the destination cluster. Choose **More > Restart** in the **Operation** column to restart the cluster and make the change take effect.

Migrating Indexes Using the Reindex API

1. In the destination cluster, create an index structure identical to that in the source cluster.
 - a. Log in to the CSS management console.
 - b. In the navigation pane on the left, choose **Clusters > Elasticsearch**.
 - c. In the Elasticsearch cluster list, locate the destination cluster, and click **Access Kibana** in the **Operation** column to log in to the Kibana console.
 - d. Click **Dev Tools** in the navigation tree on the left.
 - e. Run the following command to create an index structure that is identical to that in the source cluster:

```
PUT {index_name}
{
  Index structure of the source cluster
}
```

index_name indicates the index name after the migration. For the index structure of the source cluster, see [Obtaining Information About the Source Elasticsearch Cluster](#).

2. Run the following command to migrate data using the reindex API:
 - **Full migration:** Migrate the full amount of index data in the source cluster to the destination cluster.

On the Kibana console of the destination cluster, run the following command:

```
POST _reindex?wait_for_completion=false
{
  "source": {
    "remote": {
      "host": "http://xx.xx.xx.xx:9200", //Address of the source cluster. If the source cluster uses
      HTTPS, use https://xx.xx.xx.xx:9200.
      "username": "xxx", //Username for accessing the source cluster. It is needed for a security-
      mode cluster only.
      "password": "*****" //Password for accessing the source cluster. It is needed for a security-
      mode cluster only.
    },
    "index": "{index_name}", //Index name in the source cluster
    "size": 3000
  },
  "dest": {
    "index": "{index_name}" //Index name in the destination cluster
  }
}
```



```
}  
}
```

- **Incremental migration:** Migrate new/changed index data from the source cluster to the destination cluster based on timestamps. This method can also be used to migrate an oversized index one chunk at a time.

On the Kibana console of the destination cluster, run the following command:

```
POST _reindex?wait_for_completion=false  
{  
  "source": {  
    "remote": {  
      "host": "http://xx.xx.xx.xx:9200", //Address of the source cluster. If the source cluster uses  
      HTTPS, use https://xx.xx.xx.xx:9200.  
      "username": "xxx", //Username for accessing the source cluster. It is needed for a security-  
      mode cluster only.  
      "password": "*****" //Password for accessing the source cluster. It is needed for a security-  
      mode cluster only.  
    },  
    "query": {  
      "range": {  
        "timestamps": { //The time field  
          "gte": "xxx", //Start time of the incremental data.  
          "lte": "xxx" //End time of the incremental data.  
        }  
      }  
    },  
    "index": "index_name", //Index name in the source cluster  
    "size": 3000  
  },  
  "dest": {  
    "index": "index_name" //Index name in the destination cluster  
  }  
}
```

- **Index reorganization in the same cluster:** Use the reindex API to restructure indexes during migration.

On the Kibana console of the destination cluster, run the following command:

```
POST _reindex?wait_for_completion=false  
{  
  "source": {  
    "index": "index_name", //Index name in the source cluster  
    "size": 3000  
  },  
  "dest": {  
    "index": "index_name" //Index name in the destination cluster  
  }  
}
```

FAQ: What Do I Do If It Is Slow to Migrate an Oversized Index?

It may take a long time to migrate an oversized index. To speed up the migration, use the following methods:

- The reindex API relies on the scroll API to read data from the source cluster and write data into the destination cluster. You can set the size and slice parameters of the scroll API to increase migration concurrency and speed. For details, see [Reindex API](#).
- Before the migration, set the number of replicas to 0 for the destination index. After the migration is completed, change the number of replicas to the original value.

- Use snapshots to migrate large amounts of data. See [Migrating Data Between Huawei Cloud Elasticsearch Clusters Using Backup and Restoration](#), [Migrating Data from an On-premises Elasticsearch Cluster to Huawei Cloud Using the S3 Plugin](#), and [Migrating Data from a Third-Party Elasticsearch Cluster to Huawei Cloud Using Backup and Restoration](#) for examples.
- If the index has a time field, use incremental migration to migrate the index one chunk at a time.

1.8 Migrating Data Between Elasticsearch Clusters Using ESM

Scenario

The Elasticsearch Migration Tool (ESM) is an open-source tool designed for migrating data between Elasticsearch clusters, including between those of different versions. When using ESM to migrate data, you can adjust the migration speed by tuning the parameters of the scroll API to accommodate various network conditions and service requirements. Below are some scenarios where you might use ESM for data migration.

- Cross-version data migration: Use ESM to seamlessly migrate data when upgrading an Elasticsearch cluster to a new version, ensuring data integrity and availability during and after the upgrade.
- Cluster merging: Merge index data scattered across multiple Elasticsearch clusters into a single cluster for centralized data management and analysis.
- Cloud migration: Migrate an on-premises Elasticsearch service to the cloud to enjoy the benefits of cloud services, such as scalability, ease-of-maintenance, and cost-effectiveness.
- Changing the service provider: Switch from a third-party Elasticsearch service to Huawei Cloud for reasons related to cost, performance, or other strategic considerations.

Overview

Figure 1-12 Migration procedure

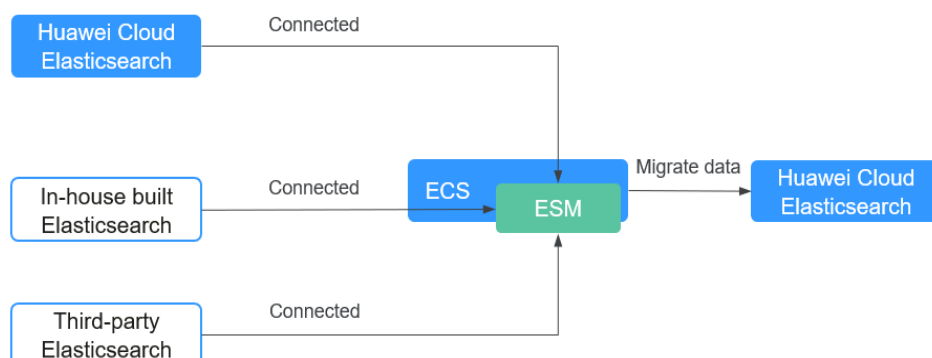


Figure 1-12 shows how to migrate data between Elasticsearch clusters using ESM.

1. Install ESM on a Linux VM.
2. Run ESM commands to migrate indexes from the source cluster to the destination cluster.

Advantages

- Cross-version data migration: You can use ESM to migrate data between Elasticsearch clusters of different versions, including from an earlier version to a later version.
- Easy to use: ESM is easy to use and is developed using the Go language. It quickly becomes available after being installed using an installation package.
- Performance control: During the migration, you can tune the parameters of the scroll API to control the data migration speed for optimal cluster performance.
- Flexible migration options: ESM supports both full migration and incremental migration, accommodating different needs.
- Free: As an open-source tool, the ESM code is hosted on GitHub and accessible to all users for free.

Impact on Performance

Using ESM for data migration between clusters relies on the scroll API. The scroll API can efficiently retrieve index data from the source cluster and synchronize the data to the destination cluster in batches. This process may impact the performance of the source cluster. The specific impact depends on how fast data is retrieved from the source cluster, and the data retrieval speed depends on the **size** and **slice** settings of the scroll API. For details, see the [Reindex API](#) document.

ESM can quickly read data from the source cluster, potentially impacting its performance. Therefore, it is advisable to perform the data migration during off-peak hours. Additionally, you may need to monitor changes in the CPU and memory metrics of the source cluster. By tuning the migration speed and choosing an appropriate time window for the migration, you can keep the performance impact under control. If you have large amounts of data to migrate or if the source cluster has a high resource usage, you should perform the migration during off-peak hours, reducing impact on the performance of the source cluster.

Constraints

During cluster migration, do not add, delete, or modify the index data of the source cluster. Otherwise, the migration may fail, or data in the source cluster may be inconsistent with that in the destination cluster after the migration.

Prerequisites

- The source and destination Elasticsearch clusters are available.
- The network between the clusters is connected.
 - If the source and destination clusters are in different VPCs, establish a VPC peering connection between them. For details, see [VPC Peering Connection Overview](#).
 - To migrate an in-house built Elasticsearch cluster to Huawei Cloud, you can configure public network access for this cluster.

- To migrate a third-party Elasticsearch cluster to Huawei Cloud, you need to establish a VPN or Direct Connect connection between the third party's internal data center and Huawei Cloud.
- Ensure that **_source** has been enabled for indexes in the cluster.
By default, **_source** is enabled. You can run the **GET {index}/_search** command to check whether it is enabled. If the returned index information contains **_source**, it is enabled.

Obtaining Elasticsearch Cluster Information

Before data migration, obtain necessary information about the source and destination clusters for configuring a migration task.

Table 1-14 Required Elasticsearch cluster information

Cluster Type		Required Information	How to Obtain
Source cluster	Huawei Cloud Elasticsearch cluster	<ul style="list-style-type: none"> • Access address of the source cluster • Username and password for accessing the source cluster (only for security-mode clusters) 	<ul style="list-style-type: none"> • For details about how to obtain the cluster name and address, see 3. • Contact the service administrator to obtain the username and password.
	In-house built Elasticsearch cluster	<ul style="list-style-type: none"> • Public network address of the source cluster • Username and password for accessing the source cluster (only for security-mode clusters) 	Contact the service administrator to obtain the information.
	Third-party Elasticsearch cluster	<ul style="list-style-type: none"> • Access address of the source cluster • Username and password for accessing the source cluster (only for security-mode clusters) 	Contact the service administrator to obtain the information.
Destination cluster	Huawei Cloud Elasticsearch cluster	<ul style="list-style-type: none"> • Access address of the destination cluster • Username and password for accessing the destination cluster (only for security-mode clusters) 	<ul style="list-style-type: none"> • For details about how to obtain the access address, see 3. • Contact the service administrator to obtain the username and password.

The method of obtaining the cluster information varies depending on the source cluster. This section describes how to obtain information about a Huawei Cloud Elasticsearch cluster.

1. Log in to the CSS management console.
2. In the navigation pane on the left, choose **Clusters > Elasticsearch**.
3. In the Elasticsearch cluster list, obtain the cluster name and address.

Figure 1-13 Obtaining cluster information

Name/ID	Cluster Status	Task Status	Version	Created	Enterprise Project	Private Network Address
CSS- d6e2a884-d68a-4...	Available		7.10.0	Jul 19, 2024 10:	default	192.168.0.130:9600,192.16...

Preparing the VM Used for the Migration

Create an ECS where you install ECM for Elasticsearch cluster migration.

1. Buy a Linux ECS, select the CentOS 7 image, and set the VPC to that of the destination cluster. For details, see [Purchasing and Using a Linux ECS](#).
2. Test the connectivity between the ECS and the source and destination clusters.

Run the following command on the ECS to test connectivity. If the correct cluster information is returned, the connection is ready.

```
# Non-security mode cluster
curl -ik http://ip:9200
# Security-mode cluster + HTTPS
curl -ik https://ip:9200 -u[Username]:[password]
```

Migrating Data Using ESM

1. Visit [the ESM download address](#), and download the **migrator-linux-amd64** installation package.
2. Use SCP to upload the downloaded **migrator-linux-amd64** to a path on the Linux ECS.
3. Run the following command in the above path on the Linux ECS to migrate index structures and data from the source cluster to the destination cluster:

```
# Full migration
./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

# Incremental migration
./migrator-linux-amd64 -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\"]"
```

For commonly used parameters in the migration command, see [Table 1-15](#). For even more information, see [ESM documents](#).

Table 1-15 Commonly used parameters in ESM migration commands

Parameter	Example	Description
-s, --source=	http://source:9200	Address for accessing the source Elasticsearch cluster

Parameter	Example	Description
-d, --dest=	http://dest:9200	Address for accessing the destination Elasticsearch cluster
-x, --src_indexes=	index_name index1,index2	The names of the indexes in the source cluster waiting to be migrated. A regular expression can be used to specify indexes. Separate multiple indexes using a comma (,).
-y, --dest_index=	index_name_rename	Index name in a destination cluster. You may specify just a single index name. If left blank, the source index names will be used.
-m, --source_auth=	admin:password	Username and password for accessing the source Elasticsearch cluster (only for security-mode clusters)
-n, --dest_auth=	admin:password	Username and password for accessing the destination Elasticsearch cluster (only for security-mode clusters)
-w, --workers=	5	The number of concurrent threads for bulk data reading. This parameter controls how fast data will be read from the source cluster. Default value: 1
-b, --bulk_size=	10	The bulk size for data reading. This parameter also controls how fast data will be read from the source cluster. Default value: 5 MB
--sliced_scroll_size	4	Size of sliced scroll. This parameter also controls how fast data will be read from the source cluster. Default value: 1
--copy_settings	-	Whether to migrate index settings in the source cluster
--copy_mappings	-	Whether to migrate index mappings in the source cluster
--buffer_count=	-	Number of files to be cached in the memory of the VM that hosts ESM. Default value: 100,000

4. After the data migration is completed, check data consistency by comparing the number of files.

```
# Non-security mode cluster
curl -ik http://ip:9200/{index name}/_count
# Security-mode cluster + HTTPS
curl -ik https://ip:9200 -u[Username]:[password]/{ index name}/_count
```

FAQ

- **How do I handle the error message "out of memory" displayed during migration?**

The error message "out of memory", if it is displayed during the migration, indicates a memory overflow on the ESM ECS. Handle it using one of the following methods:

- Use a larger flavor the ECS. For details, see [Modifying Individual ECS Specifications](#).
- Reduce the value of **buffer_count** used in the ESM migration command to reduce the number of files that can be cached in the memory of the ECS.

- **Why is the total size of index data inconsistent in the source and destination clusters after the migration?**

This is normal when ESM is used to migrate Elasticsearch clusters. In a typical Elasticsearch cluster, multiple shards are used to store data, and each shard has multiple segments. After data is migrated from the source cluster to the destination cluster using ESM, new segments and shards are generated in the destination cluster. Different numbers of segments and shards in the source and destination clusters lead to different data expansion rates, hence different data sizes. To check data consistency, compare the number of files, rather than the data size.

1.9 Migrating Kibana Saved Objects Between Elasticsearch Clusters

Scenarios

You may want to migrate Kibana's saved objects between Elasticsearch clusters in the following scenarios:

- **Data migration:** When migrating data from one Elasticsearch cluster to another, migrating Kibana's saved objects is a key step to ensure service continuity. Exporting Kibana's saved objects (such as visualizations and dashboards) from the source cluster and importing them to the destination cluster ensures consistency in the user interface and monitoring dashboards.
- **Environment replication:** When replicating an Elasticsearch environment between the development, testing, and production environments, migrating Kibana's saved objects ensures consistency across different environments, thus improving development and testing efficiency.
- **Service failure or data loss:** In the event of a service failure or data loss, migrating Kibana's saved objects to a backup cluster helps to quickly restore data monitoring and analytics capabilities.

- **Multi-cluster management:** In a multi-cluster environment, consolidating data and other objects from different Elasticsearch clusters into a single cluster enables cross-cluster data analysis and management.

These scenarios highlight the importance of migrating Kibana's saved objects between Elasticsearch clusters. It is not just about the movement of data; it's a crucial method to ensure service continuity, improve efficiency, and guarantee compliance.

Solution Architecture

Figure 1-14 Migrating Kibana saved objects between Elasticsearch clusters

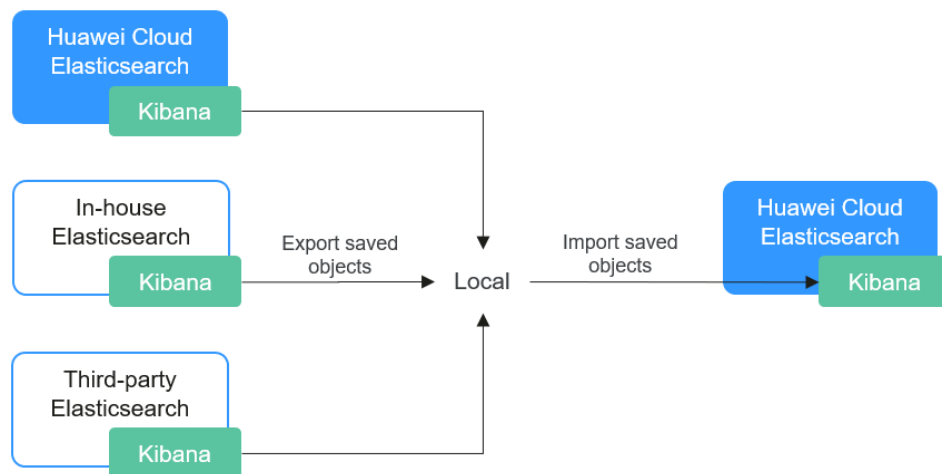


Figure 1-14 illustrates the process of migrating Kibana's saved objects between Elasticsearch clusters.

- The first step is exporting Kibana's saved objects from the source Elasticsearch cluster.
- The second step is importing these objects in the destination Elasticsearch cluster.

Advantages

- **Service continuity:** During a cluster upgrade or migration, migrating Kibana's saved objects ensures the continuity of cluster monitoring and analytics tasks.
- **Cross-environment consistency:** Replicating Kibana's saved objects across different environments (development, testing, and production) helps ensure environment consistency, improving development and testing efficiency.
- **Quick recovery:** In a fault recovery scenario, migrating Kibana's saved objects helps quickly restore cluster monitoring and analytics capabilities, mitigating the impact of any downtime.
- **Data consolidation:** In a multi-cluster environment, consolidating the data from different clusters by migrating Kibana's saved objects enables centralized data management and analysis.

Constraints

To avoid migration errors or failures, the versions of the source and destination Elasticsearch clusters must be close. If a version incompatibility error occurs during

migration, handle it by referring to [FAQ: How Do I Handle a Version Incompatibility Error Reported During the Migration of Kibana's Saved Objects?](#).

Prerequisites

The source and destination Elasticsearch clusters are available.

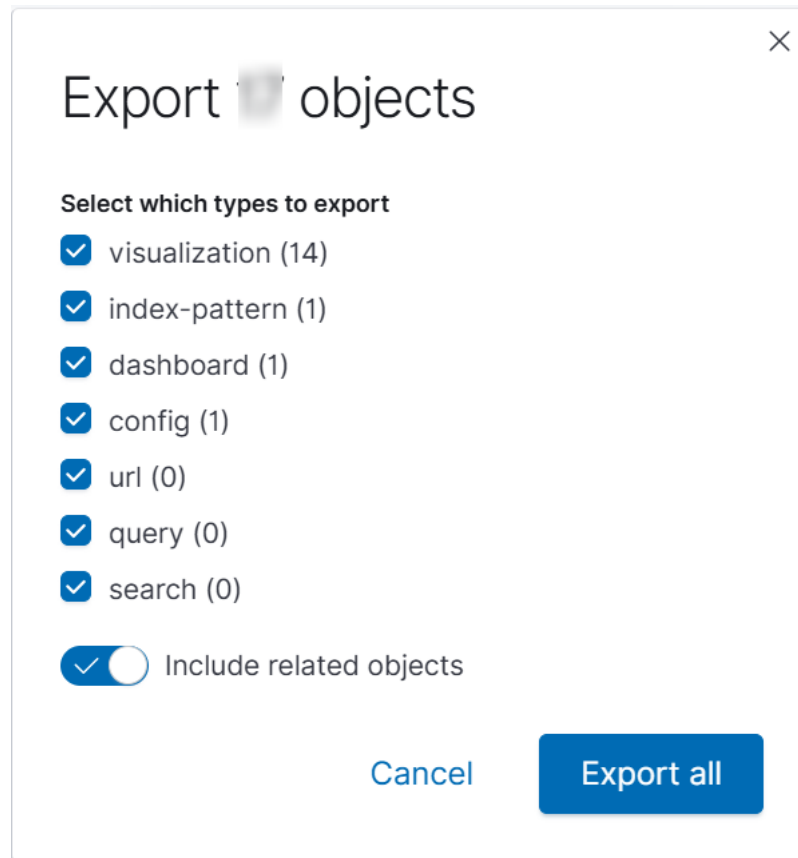
Procedure

NOTE

The Kibana UI may vary depending on the Kibana version. The following uses version 7.10.2 as an example.

- Step 1** Export Kibana's saved objects from the source Elasticsearch cluster to a local PC. In our example, the source cluster is an Elasticsearch cluster on Huawei Cloud.
1. Log in to the CSS management console.
 2. In the navigation pane on the left, choose **Clusters > Elasticsearch**.
 3. In the Elasticsearch cluster list, locate the source cluster, and click **Access Kibana** in the **Operation** column to log in to the Kibana console.
 4. In the navigation tree on the left, choose **Stack Management > Saved Objects**.
 5. On the **Saved Objects** page, click **Export xx objects**. In the displayed dialog box, select the types of objects you want to export, and click **Export all**. An **export.ndjson** file is exported to the local PC.

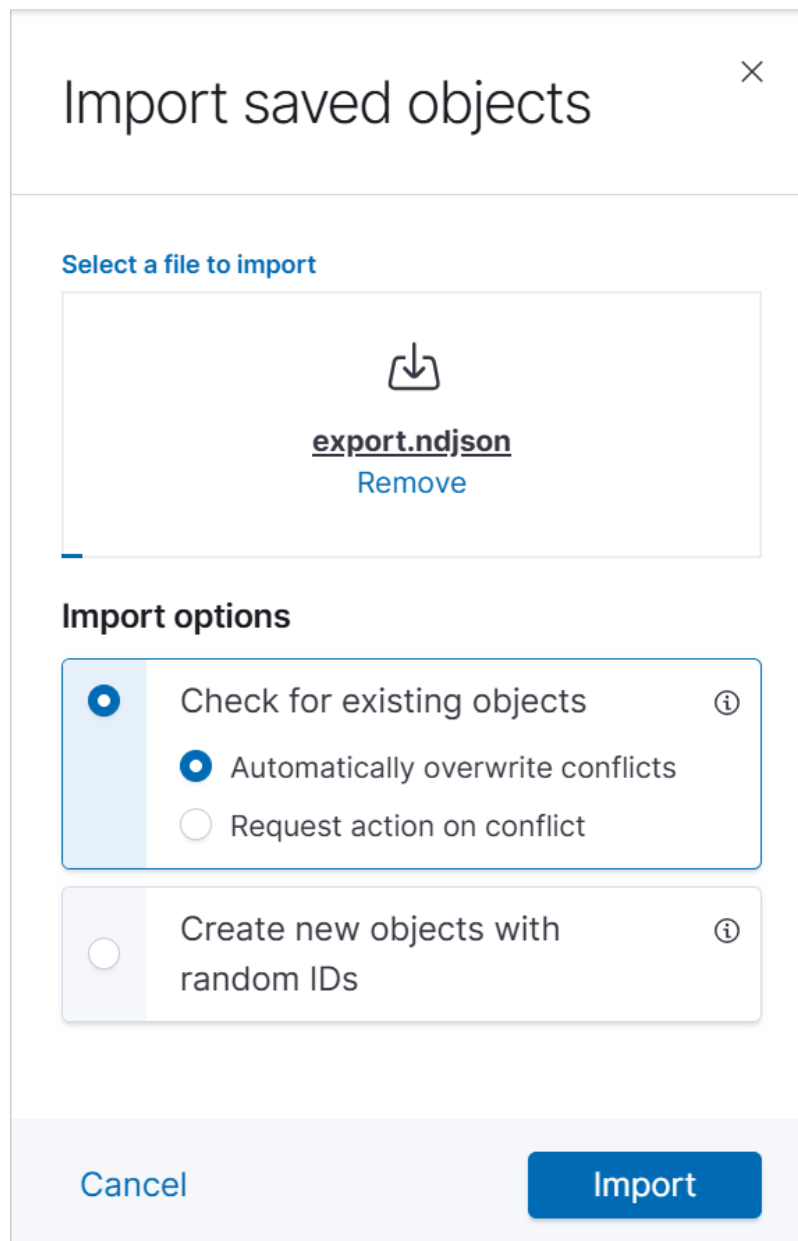
Figure 1-15 Exporting Kibana objects



Step 2 Import Kibana's saved objects exported in the previous step to the destination Elasticsearch cluster.

1. On the CSS management console, choose **Clusters > Elasticsearch**.
2. In the Elasticsearch cluster list, locate the destination cluster, and click **Access Kibana** in the **Operation** column to log in to the Kibana console.
3. In the navigation tree on the left, choose **Stack Management > Saved Objects**.
4. On the **Saved Objects** page, click **Import**. In the displayed dialog box, select the source cluster's **export.ndjson** file saved on the local PC. Select **Automatically overwrite conflicts** for **Import options**, and click **Import**.

Figure 1-16 Importing Kibana objects



5. Click Done when the import is finished.

----End

FAQ: How Do I Handle a Version Incompatibility Error Reported During the Migration of Kibana's Saved Objects?

If the following error message is displayed when you import Kibana's saved objects, the source and destination clusters have incompatibility issues.

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]"

In this case, you can modify the version information in the **export.ndjson** file on the local PC to keep the versions consistent. In this example, you need to change [7.13.0] to [7.9.0] in the code. Then, save the change and import the file again. If

the import still fails, you will have to manually rebuild the objects in the destination cluster.

2 Optimizing the Performance of Elasticsearch and OpenSearch Clusters

2.1 Optimizing the Write Performance of Elasticsearch and OpenSearch Clusters

Before using an Elasticsearch or OpenSearch cluster in CSS, you are advised to optimize the cluster's write performance to improve efficiency.

Data Write Process

Figure 2-1 Data write process

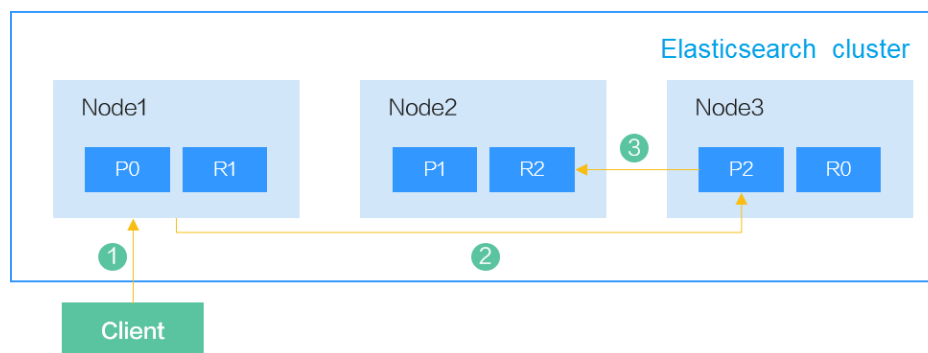


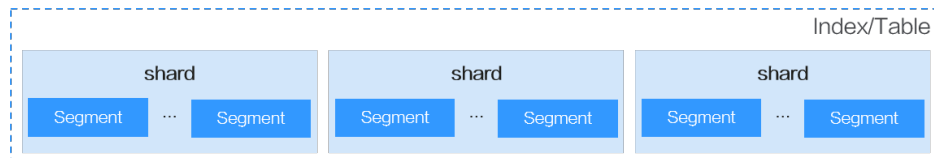
Figure 2-1 shows how a client writes data to an Elasticsearch or OpenSearch cluster. In the preceding figure, **P** indicates the primary shard, and **R** indicates the replica shard. The primary and replica shards are randomly allocated in data nodes, but cannot be in the same node.

1. The client sends a data write request to Node1. Here Node1 is the coordinator node.
2. Node1 routes the data to shard 2 based on the `_id` of the data. In this case, the request is forwarded to Node3 and the write operation is performed.
3. After data is written to the primary shard, the request is forwarded to the replica shard of Node2. After the data is written to the replica, Node3 reports

the write success to the coordinator node, and the coordinator node reports it to the client.

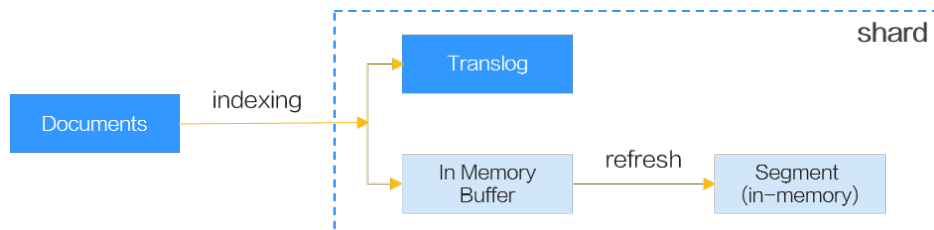
An index in Elasticsearch consists of one or more shards. Each shard contains multiple segments, and each segment is an inverted index.

Figure 2-2 Elasticsearch index composition



As shown in [Figure 2-3](#), when a document is inserted into Elasticsearch, it is written to Buffer and Translog, and then periodically refreshed to Segment. The refresh frequency is specified by the **refresh_interval** parameter. By default, data is refreshed every second. For more information about write performance, see [Near Real-Time Search](#).

Figure 2-3 Process of inserting a document into Elasticsearch



Improving Write Performance

In the Elasticsearch data write process, the following solutions can be used to improve performance:

Table 2-1 Improving write performance

Solution	Description
Use SSDs or improve cluster configurations.	Using SSDs can greatly speed up data write and merge operations. For CSS, you are advised to select the ultra-high I/O storage or ultra-high I/O servers.
Use Bulk APIs.	The client writes data in batches. You are advised to write 1 MB to 10 MB data in each batch.
Randomly generate _id .	If _id is specified, a query operation will be triggered before data is written, affecting data write performance. In scenarios where data does not need to be retrieved using _id , you are advised to use a randomly generated _id .

Solution	Description
Set a proper number of segments.	You are advised to set the number of shards to a multiple of the number of cluster data nodes. Ensure each shard is smaller than 50 GB.
Close replicas.	<p>Data write and query are performed in off-peak hours. Close data copies during writing and open them afterwards.</p> <p>The command for disabling replicas in Elasticsearch 7.x is as follows:</p> <pre data-bbox="603 562 1430 667">PUT {index}/_settings { "number_of_replicas": 0 }</pre>
Adjust the index refresh frequency.	<p>During batch data writing, you can set refresh_interval to a large value or -1 (indicating no refresh), improving the write performance by reducing refresh.</p> <p>In Elasticsearch 7.x, run the following command to set the update time to 15s:</p> <pre data-bbox="603 869 1430 974">PUT {index}/_settings { "refresh_interval": "15s" }</pre>
Change the number of write threads and the size of the write queue.	<p>You can increase the number of write threads and the size of the write queue, or error code 429 may be returned for unexpected traffic peaks.</p> <p>In Elasticsearch 7.x, you can modify the following parameters to optimize write performance: thread_pool.write.size and thread_pool.write.queue_size</p>
Set a proper field type.	<p>Specify the type of each field in the cluster, so that Elasticsearch will not regard the fields as a combination of keywords and texts, which unnecessarily increase data volume. Keywords are used for keyword search, and texts used for full-text search.</p> <p>For the fields that do not require indexes, you are advised to set index to false.</p> <p>In Elasticsearch 7.x, run the following command to set index to false for field1:</p> <pre data-bbox="603 1518 1430 1796">PUT {index} { "mappings": { "properties": { "field1": { "type": "text", "index": false } } } }</pre>

Solution	Description
Optimize the shard balancing policy.	<p>By default, Elasticsearch uses the load balance policy based on disk capacity. If there are multiple nodes, especially if some of them are newly added, shards may be unevenly allocated on the nodes. To avoid such problems, you can set the index-level parameter routing.allocation.total_shards_per_node to control the distribution of index shards on each node. You can set this parameter in the index template, or modify the setting of an existing index to make the setting take effect.</p> <p>Run the following command to modify the setting of an existing index:</p> <pre>PUT {index}/_settings { "index": { "routing.allocation.total_shards_per_node": 2 } }</pre>

2.2 Optimizing the Query Performance of Elasticsearch and OpenSearch Clusters

Before using an Elasticsearch or OpenSearch cluster in CSS, you are advised to optimize the cluster's query performance to improve efficiency.

Data Query Process

Figure 2-4 Data query process

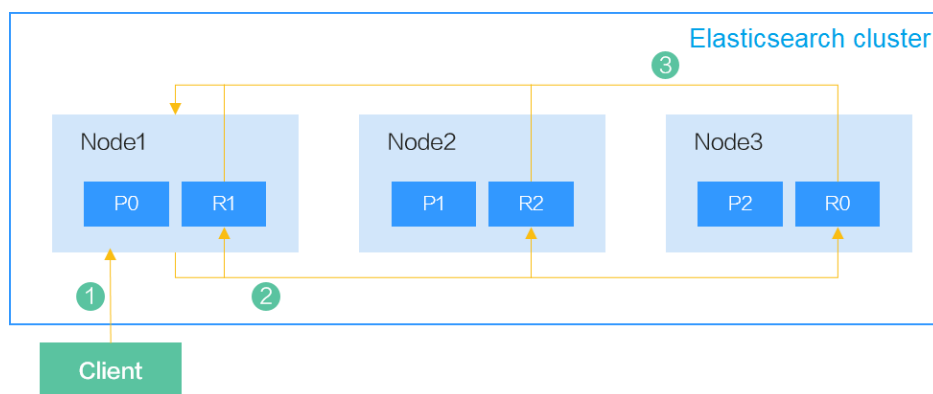


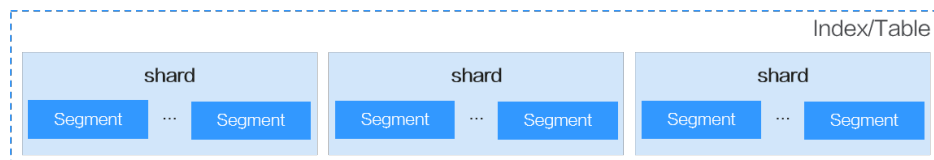
Figure 2-4 shows how a client sends a query request to an Elasticsearch or OpenSearch cluster. In the preceding figure, **P** indicates the primary shard, and **R** indicates the replica shard. The primary and replica shards are randomly allocated in data nodes, but cannot be in the same node.

1. The client sends a data query request to Node1. Here Node1 is the coordinator node.
2. Node1 selects a shard based on the shard distribution and the index specified in the query, and then forwards the request to Node1, Node2, and Node3.

- Each shard executes the query task. After the query succeeds on the shards, the query results are aggregated to Node1, which returns the results to the client.

For a query request, five shards can be queried concurrently on a node by default. If there are more than five shards, the query will be performed in batches. In a single shard, the query is performed by traversing each segment one by one.

Figure 2-5 Composition of an Elasticsearch index



Improving Query Performance

In the Elasticsearch data query process, the following solutions can be used to improve performance:

Table 2-2 Improving query performance

Solution	Description
Use _routing to reduce the number of shards scanned during retrieval.	<p>During data import, configure routing to route data to a specific shard instead of all the shards of the related index, improving the overall throughput of the cluster.</p> <p>In Elasticsearch 7.x, run the following commands:</p> <ul style="list-style-type: none"> Insert data based on a specified routing. <pre>PUT /{index}/_doc/1?routing=user1 { "title": "This is a document" }</pre> Query data based on a specified routing. <pre>GET /{index}/_doc/1?routing=user1</pre>

Solution	Description
<p>Use index sorting to reduce the number of segments to be scanned.</p>	<p>When a request is processed on a shard, the segments of the shard are traversed one by one. By using index sorting, the range query or sorting query can be terminated in advance (early-terminate).</p> <p>For example, in Elasticsearch 7.x, run the following commands:</p> <pre>// Assume the date field needs to be frequently used for range query. PUT {index} { "settings": { "index": { "sort.field": "date", "sort.order": "desc" } }, "mappings": { "properties": { "date": { "type": "date" } } } }</pre>
<p>Add query cache to improve cache hit.</p>	<p>When a filter request is executed in a segment, the bitset is used to retain the result, which can be reused for later similar queries, thus reducing the overall query workloads.</p> <p>You can add query cache by increasing the value of indices.queries.cache.size. For details, see Configuring Parameters. Restart the cluster for the modification to take effect.</p>
<p>Perform forcemerge in advance to reduce the number of segments to be scanned.</p>	<p>For read-only indexes that are periodically rolled, you can periodically execute forcemerge to combine small segments into large segments and permanently delete indexes marked as deleted.</p> <p>In Elasticsearch 7.x, a configuration example is as follows:</p> <pre>// Assume the number of segments after index forcemerge is set to 10. POST /{index}/_forcemerge?max_num_segments=10</pre>

3 Testing the Performance of CSS's Elasticsearch Vector Search

Scenarios

CSS's vector search engine provides a fully managed, high-performance distributed vector database service. To facilitate performance/pressure testing for the vector search service and provide accurate references for product selection and resource configuration, this document describes the performance testing solutions for CSS's Elasticsearch vector search service based on open-source datasets and open-source pressure testing tools.

Preparations

- Create an Elasticsearch vector database. For details, see [Creating an Elasticsearch Cluster](#).

Set the node quantity to 3, and node specifications to 4 vCPUs | 16GB under General computing. (The test data volume is relatively small, and the CPU specifications need to be kept close to those used in third-party performance benchmarking.) Select ultra-high I/O for node storage, and keep the security mode disabled.

- Obtain test datasets.
 - **sift-128-euclidean**: 128 dimensions, 1 million base records, measured using the Euclidean distance.
 - **cohere-768-cosine**: 768 dimensions, 1 million base records, measured using the cosine distance.
 - **gist-960-euclidean**: 960 dimensions, 1 million base records, measured using the Euclidean distance.

You can download **sift-128-euclidean** and **gist-960-euclidean** at <https://github.com/erikbern/ann-benchmarks>. To use the **cohere-768-cosine** dataset, submit a service ticket.

Figure 3-1 Downloading sift-128-euclidean and gist-960-euclidean
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
MNIST	784	5,500,000	10,000	100	Angular	HDF5 (2.4GB)
Fashion MNIST	784	60,000	10,000	100	Euclidean	HDF5 (217MB)
GIST	960	1,000,000	1,000	100	Euclidean	HDF5 (3.6GB)
USPS	256	1,100,000	10,000	100	Angular	HDF5 (127MB)
USPS	512	1,100,000	10,000	100	Angular	HDF5 (270MB)
USPS	1024	1,100,000	10,000	100	Angular	HDF5 (460MB)
USPS	2048	1,100,000	10,000	100	Angular	HDF5 (870MB)
ImageNet	224x224	76,000	500	100	Inward	HDF5 (27MB)
MNIST	784	60,000	10,000	100	Euclidean	HDF5 (217MB)
MNIST	48,128	60,000	500	100	Inward	HDF5 (67MB)
SIFT	256	200,000	10,000	100	Angular	HDF5 (207MB)
SIFT	128	1,000,000	10,000	100	Euclidean	HDF5 (501MB)

- Prepare the testing tools.
 - Prepare the data writing and recall testing scripts. For details, see [Script base_test_example.py](#).
 - Download the open-source pressure testing tool wrk at <https://github.com/wg/wrk/tree/master>.

Performance Testing Procedure

1. Create an ECS for installing the pressure testing tool and executing test scripts. For details, see [Purchasing and Using a Linux ECS](#).
 - This ECS and the Elasticsearch cluster created for the testing purpose must be within the same VPC and security group.
 - You may use another client server instead. No matter what server you use, ensure that it is in the same VPC as the Elasticsearch cluster.

2. Upload the test dataset to the ECS.

3. Upload the data writing and recall testing scripts to the ECS, and run the following command:

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

python3 base_test_example.py
```

This command creates a vector index for testing, writes in the test data, and returns the average recall for the queries.

4. Install wrk on the ECS.
5. On the ECS, prepare the query request file used for the pressure testing to simulate real-world traffic. See [Script prepare_query.py](#) for an example.

```
pip install h5py

python3 prepare_query.py
```

6. Prepare the wrk pressure testing configuration script on the ECS. See [Script perf.lua](#) for an example. Modify the query request file name, cluster address, and index name in the script as needed.
7. Run the following command on the ECS to perform pressure testing on CSS's vector search service:

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

- **t** indicates the number of pressure testing threads.
- **c** indicates the number of server connections.
- **d** indicates the pressure testing duration. **10m** indicates 10 minutes.
- **s** indicates the pressure testing configuration script for wrk.
- **x.x.x.x** indicates the address of the Elasticsearch cluster.

Obtain the test result from the command output, where **Requests/sec** indicates the query throughput in QPS.

Figure 3-2 Test result example

```
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
```

Performance Testing Solutions

- **GRAPH indexes**

For a database of millions of records, GRAPH indexing is recommended.

- **Testing solution 1:** Use datasets of varying numbers of data dimensions to test the maximum QPS supported by the vector database when the top-10 recall rate reaches 99%. Perform the test on each dataset using both default parameters and performance-tuned ones. By tuning the build parameters, you can optimize the graph index structure to enhance query performance while maintaining the same recall rate.

Test result:

Table 3-1 GRAPH index test result 1

Dataset	Build Parameters		Query Parameters		Metrics	
	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

Dataset	Build Parameters		Query Parameters		Metrics	
	efc	shrink	ef	max_scan_num	QPS	Recall
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

Conclusion: On all these datasets, the vector search service can reach a recall rate of 99% or higher using default parameters. By further tuning the build and query parameters, the index building overhead increases slightly, but this also results in improved query performance.

- **Testing solution 2:** Use the same dataset to test the vector search service's query performance under different recall rates by tuning index parameters. This solution uses the Cohere dataset to test the maximum QPS of the cluster under different top-10 recall rates—99%, 98%, and 95%, respectively.

Test result:

Table 3-2 GRAPH index test result 1

Dataset	Build Parameter	Query Parameter	Metrics	
	efc	ef	QPS	Recall
cohere-768-cosine	500	128	3687	0.99
	500	80	5320	0.98
	500	36	9028	0.95

Conclusion: With fixed index building parameters for the same cluster, tuning the ef parameter can achieve different query precisions. Slightly sacrificing the recall rate can significantly boost QPS.

- **GRAPH_PQ indexes**

Graph-based indexing typically requires residual memory to ensure query performance. When dealing with a large number of vector dimensions or high data volumes, memory resources become a crucial factor affecting costs and performance. Specifically, high-dimensional vectors and large datasets demand significantly more memory, increasing storage costs and directly

impacting the efficiency and response times of the indexing algorithm. In this scenario, the GRAPH_PQ indexing algorithm is recommended.

Testing solution: Use the COHERE and GIST datasets with high data dimensions to test the cluster's maximum QPS under a top-10 recall rate of 95%. Compare the residual memory overhead with that of GRAPH indexes.

Test result:

Table 3-3 GRAPH_PQ index test result

Dataset	Build Parameters		Query Parameters		Metrics		Memory Overhead	
	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

Conclusion: The result shows that GRAPH_PQ indexing can achieve the same or similar precision and QPS as GRAPH indexing while cutting the memory overhead more than 10 times. By integrating graph indexing and quantization, the GRAPH_PQ indexing algorithm used by CSS's vector search service significantly reduces the memory overhead and increases per-node data capacity.

Table 3-4 describes the indexing parameters used above. For more information about the build parameters, see [Creating Vector Indexes in an Elasticsearch Cluster](#). For more information about the query parameters, see [Using Vector Indexes for Data Search in an Elasticsearch Cluster](#).

Table 3-4 Description of index parameters

Type	Parameter	Description
Build parameter	efc	Queue size of the neighboring node during HNSW build. The default value is 200 . A larger value indicates a higher precision and slower build speed.
	shrink	Cropping coefficient during HNSW build. The default value is 1.0f .

Type	Parameter	Description
	fragment_num	Number of fragments. The default value is 0 . The plugin automatically sets the number of fragments based on the vector length.
Query parameter	ef	Queue size of the neighboring node during the query. A larger value indicates a higher query precision and slower query speed. The default value is 200 .
	max_scan_num	Maximum number of scanned nodes. A larger value indicates a higher query precision and slower query speed. The default value is 10000 .
	topk	The number of top-k records returned for a query.

Script 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)

# For more information about the index parameters, see Creating Vector Indexes in an Elasticsearch Cluster.
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")

# For more information about the query parameters, see Using Vector Indexes for Data Search in an Elasticsearch Cluster.
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")
    # Adjust the number of shards and replicas, indexing algorithm, and index parameters based on the
    # testing requirements. In our example here, one shard and two replicas are configured for performance
```

```
testing.  
    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__":  
    # Change the value to the address of the CSS cluster.  
    client = get_client(['http://x.x.x.x:9200'])  
    test_sift(client)
```

Script 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):  
    """  
    This function is used to read query vectors from the source data files in HDF5 format and generate a  
    complete query request body for performance testing.  
    :param src: path of the source data files in HDF5 format.  
    :param dst: destination file path.  
    :param size: number of query results returned.  
    :param k: number of top-k similar results returned by querying a segment-level index.  
    :param ef: specifies the queue size used during a query.  
    :param msn: specifies max_scan_num.  
    :param metric: metric used for result rescoring, such as euclidean, cosine, and inner_product.  
    :param rescore: whether to use rescoring. You can enable rescoring for GRAPH_PQ indexes.  
    :param use_base64: whether to use Base64-encoded vector data.  
    """  
    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__":
```

```
# Change the value to the data file address.
prepare_query(r"/path/to/sift-128-euclidean.hdf5", r"requests.txt")
```

Script perf.lua

```
local random = math.random

local reqs = {}
local cnt = 0

-- Rename the query request file used for pressure testing as needed.
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
end

if #addrs == 0 then
    -- Change the value to the cluster address.
    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

-- Change the index name as needed.
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 Using Elasticsearch to Accelerate Query and Analysis for Relational Databases

This section explains how to synchronize data from a MySQL database to an Elasticsearch cluster in CSS to enable full-text search, ad hoc queries, and statistical analysis of the database.

Scenario

Using Elasticsearch to accelerate relational databases can overcome some limitations of relational databases and enable more efficient and intelligent data processing and analysis. It is typically used in the following scenarios:

- E-commerce platform: Quickly search for commodities, provide personalized recommendations, and monitor user behavior and transaction data in real time.
- Content management system: Efficiently retrieve a large number of documents and contents, supporting complex queries and data analysis.
- Financial services: Monitor transaction data in real time for risk analysis and fraud detection.
- Social media analysis: Perform sentiment analysis, trend analysis, and influence evaluation on user-generated content.
- Customer relationship management: Quickly search for customer information, analyze customer behavior, and provide customized services.
- Log and event monitoring: Collect and analyze a large amount of log data to monitor system status and security events in real time.
- Healthcare records: Quickly retrieve and analyze patient records to support clinical decision-making and research.

Overview

Figure 4-1 Architecture of the Elasticsearch-accelerated relational database solution



1. Service data is stored in the MySQL database.
2. DRS synchronizes data from MySQL databases to the CSS Elasticsearch cluster in real time.
3. Users perform full-text search, Ad Hoc query, and statistics analysis in the Elasticsearch cluster.

Advantages

The advantages of using Elasticsearch to accelerate relational databases are as follows:

- Improved full-text search capability: Elasticsearch is a search engine that provides powerful full-text search functions. Relational databases are usually not good at full-text retrieval, but Elasticsearch can effectively solve this problem.
- High-concurrency Ad Hoc query: Elasticsearch is designed to process a large number of concurrent query requests, especially in Ad Hoc query scenarios. It can provide fast response to meet query requirements in high-concurrency environments.
- Real-time data synchronization: Data Replication Service (DRS) synchronizes data from the MySQL database to Elasticsearch in real time, ensuring data consistency and real-time performance.
- Simplified data migration and index creation: In an Elasticsearch cluster, you can create indexes that match the MySQL database table structure, simplifying data migration and index management.
- Flexible query language: Elasticsearch provides flexible query languages to support complex query construction, such as range queries, fuzzy queries, and aggregation queries. This may require more complex SQL statements in relational databases.
- Statistical analysis: The aggregation function of Elasticsearch can quickly collect and analyze data, such as age distribution statistics, which may require more computing resources and time in relational databases.
- Security and stability: You can configure security-mode Elasticsearch clusters and MySQL databases and use SSL connections between them to ensure data transmission security and system stability.
- Easy monitoring and maintenance: Elasticsearch provides various monitoring tools and APIs to facilitate system maintenance and performance monitoring.
- Scalability: The Elasticsearch cluster can be horizontally expanded based on service requirements. More nodes can be added to process larger data volumes and query loads.

These advantages make the Elasticsearch cluster an effective supplement to relational databases in processing full-text retrieval and high-concurrency Ad Hoc queries.

Prerequisites

- A MySQL database and an Elasticsearch cluster in security mode are available, and they are in the same VPC and security group.
- Data to be synchronized exists in the MySQL database.

This document uses the following table structure and initial data as an example.

- a. Create a student information table in MySQL.

```
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. Insert the initial data of three students into the MySQL database.

```
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');
```

- An index has been created in the Elasticsearch cluster and matches the table indexes in the MySQL database.

Run the following command to create an index for the Elasticsearch cluster:

```
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"  
      }  
    }  
  }  
}
```

Configure **number_of_shards** and **number_of_replicas** as needed.

Procedure

- Step 1** Use DRS to synchronize MySQL data to CSS in real time. For details, see [From MySQL to CSS/ES](#).

In this example, configure the parameters by following the suggestions in [Table 4-1](#).

Table 4-1 Synchronization parameters

Module	Parameter	Suggestion
Create Synchronization Instance > Synchronize Instance Details	Network Type	Select VPC .
	Source DB Instance	Select the RDS for MySQL instance to be synchronized, that is, the MySQL database that stores service data.
	Synchronization Instance Subnet	Select the subnet where the synchronization instance is located. You are advised to select the subnet where the database instance and the Elasticsearch cluster are located.
Configure Source and Destination Databases > Destination Database	VPC	Select the same VPC as the Elasticsearch cluster.
	Subnet	Select the same subnet as the Elasticsearch cluster.
	IP Address or Domain Name	Enter the IP address of the Elasticsearch cluster. For details, see Obtaining the IP address of a CSS cluster .
	Database Username	Enter the administrator username (admin) and password of the Elasticsearch cluster.
	Database Password	Enter the administrator password of the Elasticsearch cluster.
	Encryption Certificate	Select the security certificate of the Elasticsearch cluster. If SSL Connection is not enabled, you do not need to select any certificate. For details, see Obtaining the security certificate of a CSS cluster .
Set Synchronization Task	Flow Control	Select No .
	Synchronization Object Type	Deselect Table structure , because the indexes matching MySQL tables have been created in the Elasticsearch cluster.
	Synchronization Object	Select Tables . Select the database and table name corresponding to the Elasticsearch cluster. NOTE Ensure the type name in the configuration item is the same as the index name, that is, _doc .
Process Data	-	Click Next .

After the synchronization task is started, wait until the **Status** of the task changes from **Full** synchronization to **Incremental**, indicating real-time synchronization has started.

Step 2 Check the synchronization status of the database.

1. Verify full data synchronization.

Run the following command in Kibana of the Elasticsearch cluster to check whether full data has been synchronized to CSS:

```
GET student/_search
```

2. Insert new data in the source cluster and check whether the data is synchronized to Elasticsearch.

For example, insert a record whose **id** is **4** in the source cluster.

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

Run the following command in Kibana of the Elasticsearch cluster to check whether new data has been synchronized to CSS:

```
GET student/_search
```

3. Update data in the source cluster and check whether the data is synchronized to Elasticsearch.

For example, in the record whose **id** is **4**, change the value of **age** from **50** to **55**.

```
UPDATE student set age='55' WHERE id=4;
```

Run the following command in Kibana of the Elasticsearch cluster to check whether the data is updated in CSS:

```
GET student/_search
```

4. Delete data from the source cluster and check whether the data is deleted synchronously from Elasticsearch.

For example, delete the record whose **id** is **4**.

```
DELETE FROM student WHERE id=4;
```

Run the following command in Kibana of the Elasticsearch cluster to check whether the data is deleted synchronously from CSS:

```
GET student/_search
```

Step 3 Verify the full-text search capability of the database.

For example, run the following command to query the data that contains **avatar** in **dsc** in the Elasticsearch cluster:

```
GET student/_search
{
  "query": {
    "match": {
      "dsc": "avatar"
    }
  }
}
```

Step 4 Verify the ad hoc query capability of the database.

For example, query **philanthropist** whose age is greater than **40** in the Elasticsearch cluster.


```
GET student/_search
{
  "query": {
    "bool": {
      "must": [
        {
          "match": {
            "dsc": "philanthropist"
          }
        },
        {
          "range": {
            "age": {
              "gte": 40
            }
          }
        }
      ]
    }
  }
}
```

Step 5 Verify the statistical analysis capability of the database.

For example, collect statistics on the age distributions of all users in the Elasticsearch cluster.

```
GET student/_search
{
  "size": 0,
  "query": {
    "match_all": {}
  },
  "aggs": {
    "age_count": {
      "terms": {
        "field": "age",
        "size": 10
      }
    }
  }
}
```

----End

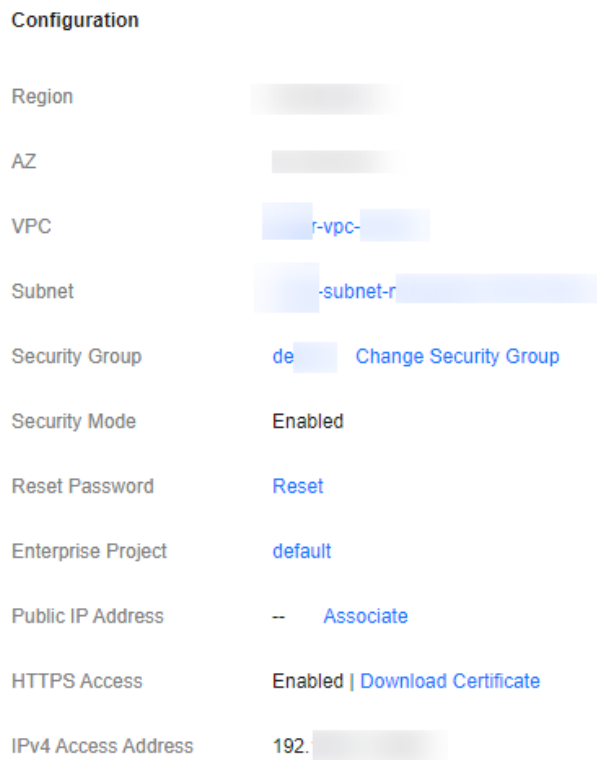
Other Operations

- **Obtaining the IP address of a CSS cluster**
 - a. In the navigation pane on the left, choose **Clusters**.
 - b. In the cluster list, obtain the IP address of a CSS cluster from the **Private Network Address** column. Generally, the IP address format is *<host>:<port>* or *<host>:<port>,<host>:<port>*.

If the cluster has only one node, the IP address and port number of only one node are displayed, for example, **10.62.179.32:9200**. If the cluster has multiple nodes, the IP addresses and port numbers of all nodes are displayed, for example, **10.62.179.32:9200,10.62.179.33:9200**.
- **Obtaining the security certificate of a CSS cluster**
 - a. Log in to the CSS management console.
 - b. In the navigation pane, choose **Clusters**. The cluster list is displayed.
 - c. Click a cluster name to go to the cluster details page.

- d. In the **Configuration** area, click **Download Certificate** next to **HTTPS Access**.

Figure 4-2 Download Certificate



Configuration	
Region	[Redacted]
AZ	[Redacted]
VPC	[Redacted] r-vpc-
Subnet	[Redacted] -subnet-r [Redacted]
Security Group	de Change Security Group
Security Mode	Enabled
Reset Password	Reset
Enterprise Project	default
Public IP Address	-- Associate
HTTPS Access	Enabled Download Certificate
IPv4 Access Address	192. [Redacted]

5 Using Elasticsearch, In-House Built Logstash, and Kibana to Build a Log Management Platform

A unified log management platform built using a CSS Elasticsearch cluster can manage logs in real time in a unified and convenient manner, enabling log-driven O&M and improving service management efficiency.

Scenarios

This document utilizes Elasticsearch, Filebeat, Logstash, and Kibana to illustrate the construction of a unified log management platform. Filebeat collects ECS logs and forwards them to Logstash for data processing. The processed data is stored in Elasticsearch and can be queried, analyzed, and visualized using Kibana. This solution is applicable in the following scenarios:

- **Log Management:** Centrally manage application and system logs to quickly identify faults.
- **Security Monitoring:** Detect and respond to security threats, detect intrusions, and analyze abnormal behaviors.
- **Service Analysis:** Analyze user behaviors to optimize products and services.
- **Performance Monitoring:** Monitor system and application performance in real-time to detect bottlenecks.

Overview

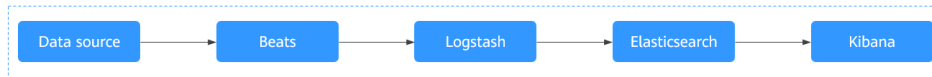
Elasticsearch, Logstash, Kibana, and Beats (ELKB) provides a complete set of log solutions and is a mainstream log system.

- Elasticsearch is an open-source, distributed search and analytics engine used to store, search, and analyze large volumes of data.
- Logstash is a server-side data pipeline that collects, parses, and enriches data before sending it to Elasticsearch.
- Kibana provides an open-source data analysis and visualization platform for Elasticsearch, enabling users to search, view, and interact with the data stored in Elasticsearch.

- Beats, such as Filebeat and Metricbeat, are lightweight data collectors installed on servers to collect and forward data to Logstash.

Figure 5-1 shows the architecture of the log management platform using Elasticsearch and Logstash.

Figure 5-1 ELKB architecture



1. Collection

- As a data collector, Beats gather data from various sources and send it to Logstash.
- Logstash can independently collect data or receive it from Beats to filter, transform, and enhance the data.

2. Data processing

Before sending data to Elasticsearch, Logstash performs necessary processing, such as parsing structured logs and filtering out irrelevant information.

3. Data storage

As a core storage component, Elasticsearch indexes and stores data from Logstash, providing quick search and data retrieval functions.

4. Data analysis and visualization

Kibana is used to analyze and visualize data in Elasticsearch, allowing users to create dashboards and reports to visualize the data.

For details about the version compatibility of ELKB components, see https://www.elastic.co/support/matrix#matrix_compatibility.

Advantages

- Real-time: Provide real-time data collection and analysis capabilities.
- Flexibility: Support various data sources and flexible data processing flows.
- Ease of use: The user-friendly interface simplifies data operations and visualization.
- Scalability: Offer strong horizontal expansion capabilities, enabling the processing of petabyte-level data.

Prerequisites

- You have created an Elasticsearch cluster in non-security mode. For details, see [Creating an Elasticsearch Cluster](#).
- You have applied for an ECS and installed the Java environment. For details about how to purchase an ECS, see [Purchasing and Logging In to a Linux ECS](#).

Procedure

Step 1 Log in to the ECS, deploy and configure Filebeat.

1. Download Filebeat. The recommended version is 7.6.2. Download it at <https://www.elastic.co/downloads/past-releases#filebeat-oss>.
2. Configure the Filebeat configuration file **filebeat.yml**.

For example, to collect all the files whose names end with **log** in the **/root/** directory, configure the **filebeat.yml** file is as follows:

```
filebeat.inputs:
- type: log
  enabled: true
  # Path of the collected log file
  paths:
  - /root/*.log

filebeat.config.modules:
  path: ${path.config}/modules.d/*.yml
  reload.enabled: false
# Logstash hosts information
output.logstash:
  hosts: ["192.168.0.126:5044"]

processors:
```

Step 2 Deploy and configure Logstash in-house.

NOTE

To achieve better performance, you are advised to set the JVM parameter to half of the ECS or docker memory for in-house built Logstash.

1. Download Logstash. The recommended version is 7.6.2. Download it at <https://www.elastic.co/downloads/past-releases#logstash-oss>.
2. Ensure that Logstash and the CSS cluster are connected. Run the **curl http://{ip}:{port}** command on the VM to test the connectivity between the VM and the Elasticsearch cluster. If 200 is returned, they are connected.
3. Configure the Logstash configuration file **logstash-sample.conf**.

The content of the **logstash-sample.conf** file is as follows:

```
input {
  beats {
    port => 5044
  }
}
# Split data.
filter {
  grok {
    match => {
      "message" => "[%{GREEDYDATA:timemaybe}]" "[%{WORD:level}]" "%{GREEDYDATA:content}"
    }
  }
  mutate {
    remove_field => ["@version","tags","source","input","prospector","beat"]
  }
}
# CSS cluster information
output {
  elasticsearch {
    hosts => ["http://192.168.0.4:9200"]
    index => "%{[@metadata][beat]}-%{+YYYY.MM.dd}"
    #user => "xxx"
    #password => "xxx"
  }
}
```

 NOTE

You can use Grok Debugger (<https://grokdebugger.com/>) to configure the **filter** mode of Logstash.

Step 3 Configure the index template of the Elasticsearch cluster.

1. Log in to the CSS management console.
2. In the navigation pane on the left, choose **Clusters > Elasticsearch**.
3. In the cluster list, locate the target cluster and click **Kibana** in the **Operation** column to log in to Kibana.
4. Click **Dev Tools** in the navigation tree on the left.
5. Create an index template.

For example, create an index template. Let the index use three shards and no replicas. Fields such as **@timestamp**, **content**, **host.name**, **level**, **log.file.path**, **message** and **timemaybe** are defined in the index.

```
PUT _template/filebeat
{
  "index_patterns": ["filebeat*"],
  "settings": {
    # Define the number of shards.
    "number_of_shards": 3,
    # Define the number of copies.
    "number_of_replicas": 0,
    "refresh_interval": "5s"
  },
  # Define a field.
  "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"
      }
    }
  }
}
```

```
}
}
}
```

Step 4 Prepare test data on ECS.

Run the following command to generate test data and write the data to **/root/tmp.log**:

```
bash -c 'while true; do echo [$(date)] [info] this is the test message; sleep 1; done;' >> /root/tmp.log &
```

The following is an example of the generated test data:

```
[Thu Feb 13 14:01:16 CST 2020] [info] this is the test message
```

Step 5 Run the following command to start Logstash:

```
nohup ./bin/logstash -f /opt/pht/logstash-6.8.6/logstash-sample.conf &
```

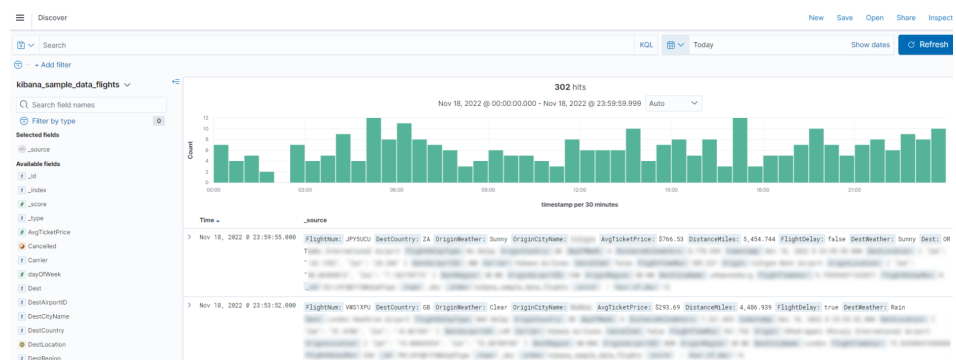
Step 6 Run the following command to start Filebeat:

```
./filebeat
```

Step 7 Use Kibana to query data and create reports.

1. Enter the Kibana page of the Elasticsearch cluster.
2. Click **Discover** in the navigation tree on the left, as shown in [Figure 5-2](#).

Figure 5-2 Discover page



----End

6 Ranking Search Results Using Elasticsearch Custom Rules

You can use the Elasticsearch cluster to sort the search results based on customized rules.

Scenario

Elasticsearch is a highly scalable open-source search and analysis engine. It allows users to sort search results based on customized rules. Custom sorting enables the definition of specific sorting rules based on service requirements, optimizing the relevance of search results and enhancing user experience. This solution is applicable in the following scenarios:

- E-commerce: Sort offerings based on factors such as sales volume, user comments, and prices.
- Content Management: Sort articles or blog entries based on the number of views and publishing time.
- Financial Services: Sort transaction records based on transaction amount, frequency, or risk score.
- Customer Support: Sort customer requests based on the urgency or opening time of service tickets.

Solution Architecture

The sorting API in Elasticsearch is used to sort search results according to customized rules. By calling the sorting API, you can query and sort data based on customized rules.

You can query with customized rules using either of the following methods:

- Calculate the final scores (**new_score**) of query results based on **vote** and sort the results in descending order.
$$\text{new_score} = \text{query_score} \times (\text{vote} \times \text{factor})$$
 - **query_score**: calculated based on the total number of search keywords found in a record. A record earns 1 point for each keyword it contains.
 - **vote**: vote of a record.
 - **factor**: user-defined weight of **vote**.

- Calculate the final scores (**new_score**) of query results based on **inline** and sort the results in descending order.
$$\text{new_score} = \text{query_score} \times \text{inline}$$
 - **query_score**: calculated based on the total number of search keywords found in a record. A record earns 1 point for each keyword it contains.
 - **vote**: vote of a record.
 - **inline**: Configure two value options for this parameter and a threshold for **vote**. One option is used if **vote** exceeds the threshold, and the other is used if **vote** is smaller than or equal to the threshold. In this way, the query accuracy will not be affected by abnormal **vote** values.

Advantages

- **Flexibility**: Customized sorting rules can meet various complex service requirements.
- **Scalability**: The distributed nature of Elasticsearch supports horizontal expansion to accommodate increasing data volumes.
- **Performance**: Elasticsearch's optimization mechanisms ensure efficient sorting operations, maintaining good performance even with large-scale datasets.
- **Real-time**: The near real-time search capability of Elasticsearch ensures the timeliness of sorting results.

Prerequisites

An Elasticsearch cluster is available.

Procedure

NOTE

The code examples in this section can only be used for clusters Elasticsearch 7.x or later.

1. Log in to the CSS management console.
2. In the navigation pane on the left, click **Clusters** to go to the Elasticsearch cluster list.
3. Click **Access Kibana** in the **Operation** column of a cluster.
4. In the navigation tree on the left of Kibana, choose **Dev Tools**. The command execution page is displayed.
5. Create an index and specify a custom mapping to define the data type.

For example, the content of the **tv.json** file is as follows:

```
{
  "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 }
  ]
}
```

Run the following command to create the **mall** index and specify the user-defined mapping to define the data type:

```
PUT /mall?pretty
{
  "mappings": {
```

```
"properties": {
  "name": {
    "type": "text",
    "fields": {
      "keyword": {
        "type": "keyword"
      }
    }
  },
  "description": {
    "type": "text",
    "fields": {
      "keyword": {
        "type": "keyword"
      }
    }
  },
  "vote": {
    "type": "float"
  }
}
```

6. Import data.

Run the following command to import data in the **tv.json** file to the **mall** index:

```
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. Query data based on customized rules. The query results can be scored based on **vote** or **inline**.

Assume a user wants to query TVs with USB, HDMI, and/or DisplayPort ports. The final query score can be calculated in the following ways and used for sorting:

– Scoring based on **vote**

The score is calculated using the formula **new_score = query_score x (vote x factor)**. Run the following command:

```
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
    }
  }
}
```

```
}  
}
```

The query results are displayed in descending order of the score. The command output is as follows:

```
{  
  "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  
        }  
      }  
    ]  
  }  
}
```

- Scoring based on **inline**.

The score is calculated using the formula **new_score = query_score x inline**. In this example, if **vote** > 0.8, the value of **inline** is 1. If **vote** ≤ 0.8, the value of **inline** is 0.5. Run the following command:

```
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
    }
  }
}
```

The query results are displayed in descending order of the score. The command output is as follows:

```
{
  "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 Synchronizing Data from RDS for MySQL to Elasticsearch Through Logstash

Scenarios

The logstash-input-jdbc plugin is installed by default in CSS's Logstash clusters. This plugin allows Logstash to fetch data from the relational database RDS for MySQL and push the data to Elasticsearch for deeper analysis. All you need to do is to configure the Logstash configuration file to define the JDBC input and Elasticsearch output. This solution can be used for the following purposes:

- **Real-time data update and synchronization:** Synchronizes an RDS for MySQL database with Elasticsearch in real time to leverage the latter's powerful search and analytics capabilities.
- **Log analytics and search:** Synchronizes log data from RDS for MySQL to Elasticsearch for quick search and analytics.
- **Application performance monitoring:** Synchronizes application performance data stored in RDS for MySQL to Elasticsearch through Logstash for real-time performance monitoring and performance analysis.
- **Data backup and recovery:** Uses Logstash to back up the data in RDS for MySQL to Elasticsearch, allowing for quick recovery in case of data loss or corruption.

Solution Architecture

Figure 7-1 Synchronizing RDS for MySQL with Elasticsearch

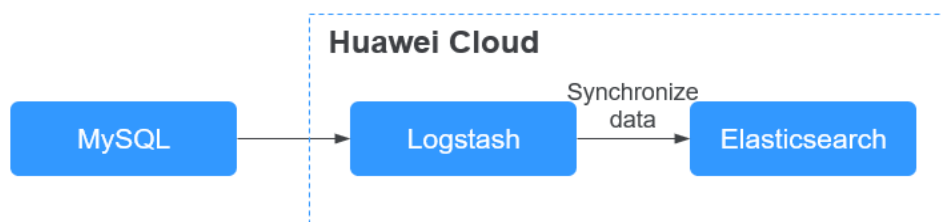


Figure 7-1 illustrates the process of synchronizing RDS for MySQL with Elasticsearch through Logstash.

The logstash-input-jdbc plugin enables both full data migrations and incremental data updates from RDS for MySQL to Elasticsearch.

Advantages

- **Flexible:** Logstash offers versatile data collection, transformation, optimization, and output capabilities, meeting diverse data synchronization needs.
- **Real-time:** Logstash supports near-real time data synchronization, accommodating the needs of most scenarios.
- **Easy-to-use:** The Logstash configuration file can be configured to meet various synchronization needs. The configuration is straightforward and requires no coding.

Constraints

- The same ID fields must be used in both Elasticsearch and RDS for MySQL. This is to establish a direct mapping between MySQL records and Elasticsearch indexes. For instance, when a record is updated in RDS for MySQL, the synchronization task overwrites the corresponding index in Elasticsearch with the same ID.
- New or updated records in RDS for MySQL must include a field indicating the insertion or update time.
Logstash records the latest timestamp for each record during each polling cycle. During the next poll, it only processes records with timestamps later than the previously recorded ones.
- The RDS for MySQL database, Logstash cluster, and Elasticsearch cluster must be in the same time zone; otherwise, there may be time zone-related errors after the synchronization.

Prerequisites

- An RDS for MySQL database is available. This topic uses an RDS for MySQL instance on Huawei Cloud as an example. For details, see [Buying an RDS for MySQL DB Instance](#).
- A Logstash cluster used for data synchronization has been created. For details, see [Creating a Logstash Cluster](#). This topic uses a Logstash 7.10.0 cluster as an example.
- An Elasticsearch cluster has been created. For details, see [Creating an Elasticsearch Cluster](#). This topic uses an Elasticsearch 7.10.2 cluster as an example.

The RDS for MySQL DB, Logstash cluster, and Elasticsearch cluster are in the same VPC.

If an in-house built or third-party MySQL database is used, check whether the database driver is MariaDB.

- Yes: Proceed to data synchronization configuration.
- No: Upload a SQL JDBC driver that is compatible with the RDS database version in use to the Logstash cluster. For details, see [FAQ: What Do I Do If the MySQL Driver Is Incompatible?](#).

Procedure

Step 1 Test connectivity between the Logstash cluster and data sources.

1. Log in to the CSS management console.
2. In the navigation pane on the left, choose **Clusters > Logstash**. The cluster list is displayed.
3. In the displayed cluster list, locate the row that contains the target cluster, and click **Configuration Center** in the **Operation** column. The **Configuration Center** page is displayed.

Alternatively, in the cluster list, click the cluster name to go to the cluster information page. In the navigation pane on the left, choose **Configuration Center**.

4. On the **Configuration Center** page, click **Test Connectivity**.
5. In the **Test Connectivity** dialog box, enter the IP address and port number of the data source and destination, and click **Test**.

You can test a maximum of 10 IP addresses at a time. You can click **Add** to add more IP addresses and click **Test** at the bottom to test connectivity to multiple IP addresses at a time.

Figure 7-2 Test Connectivity



If **Available** is displayed, the network is connected. If the network is disconnected, configure routes for the Logstash cluster to connect the clusters. For details, see [Configuring Routes for a Logstash Cluster](#).

Step 2 Create a Logstash configuration file for data synchronization.

1. On the Configuration Center page of the Logstash cluster, click **Create** in the upper right corner. On the **Create Configuration File** page, edit the configuration file.

Table 7-1 Creating a Logstash configuration file

Parameter	Description
Name	User-defined configuration file name. It can contain only letters, digits, hyphens (-), and underscores (_), and must start with a letter. The minimum length is 4 characters.
Configuration File Content	Configure the configuration file by referring to the code example below. NOTE The size of each configuration file cannot exceed 100 KB.

Parameter	Description
Hidden Content	For items that you enter in this box, the corresponding strings will be replaced with *** in the configurations. This configuration is not required in our example here.

```

input {
  jdbc{
    # Configure the JDBC driver.
    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"
    # Retain the default values.
    jdbc_paging_enabled => "true"
    jdbc_page_size => "50000"
    # SQL statement for data migration.
    statement => "select a.user_code AS doctor_id,a.record_status from cluster "
    # Scheduled task, which repeats itself every 5 minutes. This interval can be customized.
    schedule => "*/5 * * * *"
  }
}
filter {
}
output {
  elasticsearch {
    hosts => ["xxx.xxx.xxx.xxx:port","xxx.xxx.xxx.xxx:port","xxx.xxx.xxx.xxx:port"]
    # Set the index name.
    index => "rds_doctor_index"
    user => "admin"
    password => "xx"
    # Document ID in the index. Keep it consistent with a primary key of the target table in RDS for
MySQL.
    document_id => "%{primary_id}"
    # Configure a certificate only if the destination Elasticsearch cluster uses HTTPS.
    ssl => true
    ssl_certificate_verification => false
    cacert => "/rds/datastore/logstash/v7.10.0/package/logstash-7.10.0/extend/cert/
CloudSearchService.cer"
    # Retain the default values.
    manage_template => false
    ilm_enabled => false
  }
}

```

Table 7-2 Configuration items

Configuration Item		Mandatory	Description
input	jdbc_driver_library	Yes	<p>Path of the JDBC driver library.</p> <ul style="list-style-type: none"> - If the database driver is MariaDB, set the value to /rds/datastore/logstash/v7.10.0/package/logstash-7.10.0/extend/jars/mariadb-java-client-2.7.0.jar. - If the database driver is a SQL JDBC driver compatible with the RDS database version in use, contact technical support to set the value. <p>For how to set JDBC driver parameters, see Jdbc input plugin.</p>
	jdbc_driver_class	Yes	<p>Class path of the driver library.</p> <ul style="list-style-type: none"> - If the database driver is MariaDB, set the value to org.mariadb.jdbc.Driver. - If the database driver is a SQL JDBC driver compatible with the RDS database version in use, set the value to com.mysql.jdbc.Driver.
	jdbc_connection_string	Yes	<p>Address for accessing MySQL JDBC.</p> <ul style="list-style-type: none"> - If the database driver is MariaDB, set the value to jdbc:mariadb://xxx.xxx.xxx.xxx:port/cms?useUnicode=true&characterEncoding=utf8mb4&autoReconnect=true&allowMultiQueries=true. - If the database driver is a SQL JDBC driver compatible with the RDS database version in use, set the value to jdbc:mysql://xxx.xxx.xxx.xxx:port/cms. <p>Replace xxx.xxx.xxx.xxx:port with the database address plus port number.</p>

Configuration Item		Mandatory	Description
	jdbc_user	Yes	Username for accessing MySQL JDBC.
	jdbc_password	Yes	Password for accessing MySQL JDBC.
	statement	Yes	SQL statement for data migration.
	schedule	Yes	Scheduled task. The data synchronization interval is customizable.
output	hosts	Yes	Address for accessing the Elasticsearch cluster.
	index	Yes	Index to which data is loaded.
	user	No	Username for accessing the Elasticsearch cluster, which is required only for security-mode clusters.
	password	No	Password for accessing the Elasticsearch cluster, which is required only for security-mode clusters.
	document_id	Yes	Document ID in the index. Keep it consistent with a primary key used in the target table in RDS for MySQL.
	ssl	No	Whether to enable HTTPS communication. If HTTPS access is enabled for the Elasticsearch cluster, set this parameter to true . Otherwise, do not set this parameter.
	ssl_certificate_verification	No	Whether to verify the server-end Elasticsearch certificate. Set this parameter only when ssl is set to true . - true : Verify the certificate. - false : Ignore the certificate.
	cacert	No	HTTPS access certificate. Retain the default for a CSS cluster.

2. Click **Next** to configure Logstash pipeline parameters. In this example, retain the default settings.
3. After the configuration is complete, click **Create**.
On the **Configuration Center** page, you can check the created configuration file. If its status changes to **Available**, it has been successfully created.

Step 3 Start the Logstash configuration file.

1. In the configuration file list, select the configuration file you want to start, and click **Start** in the upper left corner.
2. In the **Start Logstash** dialog box, select **Keepalive** if necessary.
3. Click **OK** to start the configuration file and hence the Logstash migration task. You can check the started configuration file in the pipeline list.

Step 4 Verify that the database is now synchronized with the Elasticsearch cluster.

1. On the CSS management console, choose **Clusters > Elasticsearch**.
2. In the Elasticsearch cluster list, locate the destination cluster, and click **Access Kibana** in the **Operation** column to log in to the Kibana console.
3. In the navigation tree on the left, choose **Dev Tools**.
4. Run the following command to query index data:

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

If the value of **count** is not 0 in the command output, data synchronization is successful.

----End

FAQ: What Do I Do If the MySQL Driver Is Incompatible?

After the Logstash configuration file is started in the Logstash cluster, the Logstash pipeline is not working properly. Click **Run Log**. If the log contains error information similar to the following, the MySQL driver is incompatible.

```
[2024-05-21T11:31:00,196][ERROR][logstash.inputs.jdbc ] Java::JavaSql::SQLException:
(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
```

Solution:

1. Stop the Logstash configuration file.
2. Download a SQL JDBC driver compatible with the RDS database version in use, for example, **mysql-connector-java-8.0.11.tar.gz**. Extract **mysql-connector-java-8.0.11.jar** from it.
Download address: <https://downloads.mysql.com/archives/c-j/>
3. Contact technical support to upload the .jar file to the Logstash cluster used for data synchronization.
4. Modify the Logstash configuration file.
Change the values of **jdbc_driver_class** and **jdbc_connection_string**. Replace **xxx.xxx.xxx.xxx:port** with the database address plus port number.

```
jdbc_driver_class => "com.mysql.jdbc.Driver"  
# Enter the MySQL JDBC URL.  
jdbc_connection_string => "jdbc:mysql://xxx.xxx.xxx.xxx:port/cms"
```

5. Restart the configuration file.