

Elasticsearch Service

Data Application Guide

Product Documentation



Copyright Notice

©2013-2025 Tencent Cloud. All rights reserved.

Copyright in this document is exclusively owned by Tencent Cloud. You must not reproduce, modify, copy or distribute in any way, in whole or in part, the contents of this document without Tencent Cloud's the prior written consent.

Trademark Notice



All trademarks associated with Tencent Cloud and its services are owned by the Tencent corporate group, including its parent, subsidiaries and affiliated companies, as the case may be. Trademarks of third parties referred to in this document are owned by their respective proprietors.

Service Statement

This document is intended to provide users with general information about Tencent Cloud's products and services only and does not form part of Tencent Cloud's terms and conditions. Tencent Cloud's products or services are subject to change. Specific products and services and the standards applicable to them are exclusively provided for in Tencent Cloud's applicable terms and conditions.

Contents

Data Application Guide

Data Application Overview

Data Management

Autonomous Index Overview

Creating Autonomous Index

Index Search and Analysis

Basic Index Information

Index Monitoring

Index Configuration Management

Data Application Guide

Data Application Overview

Last updated : 2024-11-29 21:36:08

Data application is a one-stop data ingestion and management service built in ES. It helps you visually manage indices in the cloud and greatly reduce the usage and maintenance costs otherwise incurred by traditional manual index creation and management. **Data management** is supported currently, and **data ingestion** will be supported in the future.

Data management provides a wealth of services, including index creation, search and analysis, index monitoring, and configuration management, to help you manage data on easy-to-use GUIs. In addition, the [automated index](#) feature developed by Tencent Cloud supports **automated index lifecycle management and sharding fine-tuning, effectively increases the read and write efficiency, and reduces the storage costs** in scenarios involving time series data, such as log analysis and Ops monitoring.

Feature Overview

Data management provides the following features:

Index creation: It offers easy-to-use **GUIs** and the flexible **JSON mode** to effectively reduce the index creation costs. For more information, see [Creating Automated Index](#).

Index search and analysis: It allows you to enter the Kibana page from the index list for quick index search and analysis. For more information, see [Index Search and Analysis](#).

Basic index information: It allows you to view the index information and manage backing indices in the console. For more information, see [Basic Index Information](#).

Index monitoring: It provides a rich set of index monitoring metrics to help you view the real-time data of indices during use. For more information, see [Index Monitoring](#).

Index configuration management: It enables you to flexibly modify the index configuration information in the index management center in the console in response to business changes. For more information, see [Index Configuration Management](#).

Data Management

Autonomous Index Overview

Last updated : 2023-12-12 10:45:34

Background

You can usually roll over Elasticsearch indices to store time series data continuously generated by logging and monitoring components. This method implements basic data management features; however, to achieve complete index management, you still need to use it in combination with index template, index lifecycle management, and index alias features. In addition, it also incurs index maintenance costs. For example, to avoid the impact of insufficient shards on the write availability, the need to roll over a new index in case of a single-replica index node failure, and the impact of too many shards on the cluster stability, you must reasonably estimate the shard quantity before index creation. In order to solve these problems, the ES team has developed the **autonomous index** feature, a one-stop index management solution for **time series data** use cases such as log analysis and Ops monitoring. To use this feature, you only need to create an autonomous index in a few simple steps and specify a single autonomous index object for read and write requests. Automatic shard quantity fine-tuning and complete index lifecycle management are embedded to enhance the index usability and reduce the index maintenance costs. This document describes the use cases, strengths, and basic concepts of the autonomous index feature.

Use Cases

The autonomous index feature is suitable for log analysis, Ops monitoring, and other time series data use cases, such as log search and analysis, metric monitoring and analysis, as well as collection, monitoring, and analysis of smart IoT device data.

Strengths

Ease of use: An autonomous index can be created with a single command and can be used for read and write operations. It has many features, such as index rollover, cold/hot data migration, and deletion upon expiry, for you to configure, so you don't need to manage index lifecycle management (ILM) policies and index templates.

Ease of maintenance: The autonomous index feature can automatically adjust the number of index shards in response to the changes in the business write load and roll over a new index in case of a failure. This significantly reduces the index maintenance costs.

Prerequisites

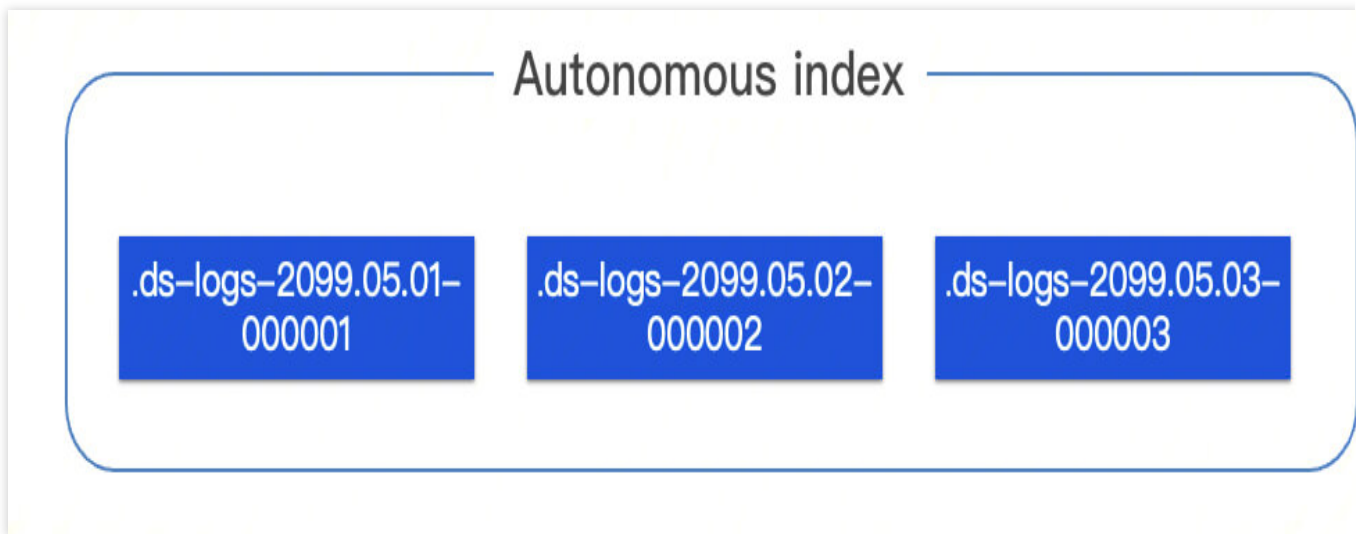
The autonomous index feature is naturally applicable to clusters on v7.14.2 created after June 1, 2022 and is supported for older clusters on this version after a rolling cluster restart. To use this feature in clusters on earlier versions, upgrade them to v7.14.2 first.

Each document written to an autonomous index must contain a time-type field with the same field name as defined in the autonomous index. If not specified during autonomous index creation, the field name is `@timestamp` by default.

Basic Concepts

Autonomous index and backing index

An autonomous index is implemented through enhancements to the Elasticsearch DataStream kernel. It is internally associated with one or more hidden backing indices (i.e., general Elasticsearch indices), so you only need to focus on and manipulate it.



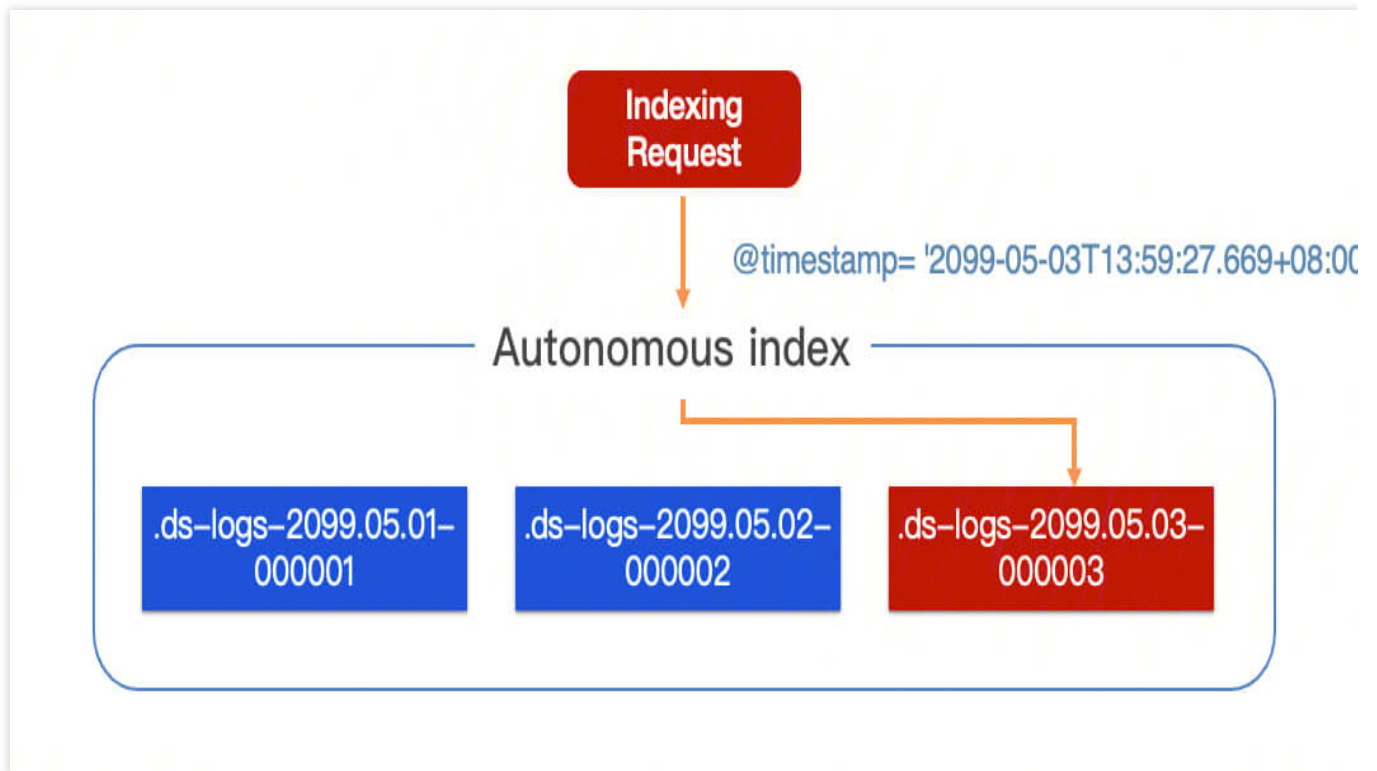
Write mode

Autonomous indexing supports two data writing modes: append writing and time-partitioned writing. In the append writing mode, data is written to the latest backup index, making it suitable for log scenarios. In the time-partitioned writing mode, data is written to the corresponding backup index based on the time field, making it ideal for metric scenarios.

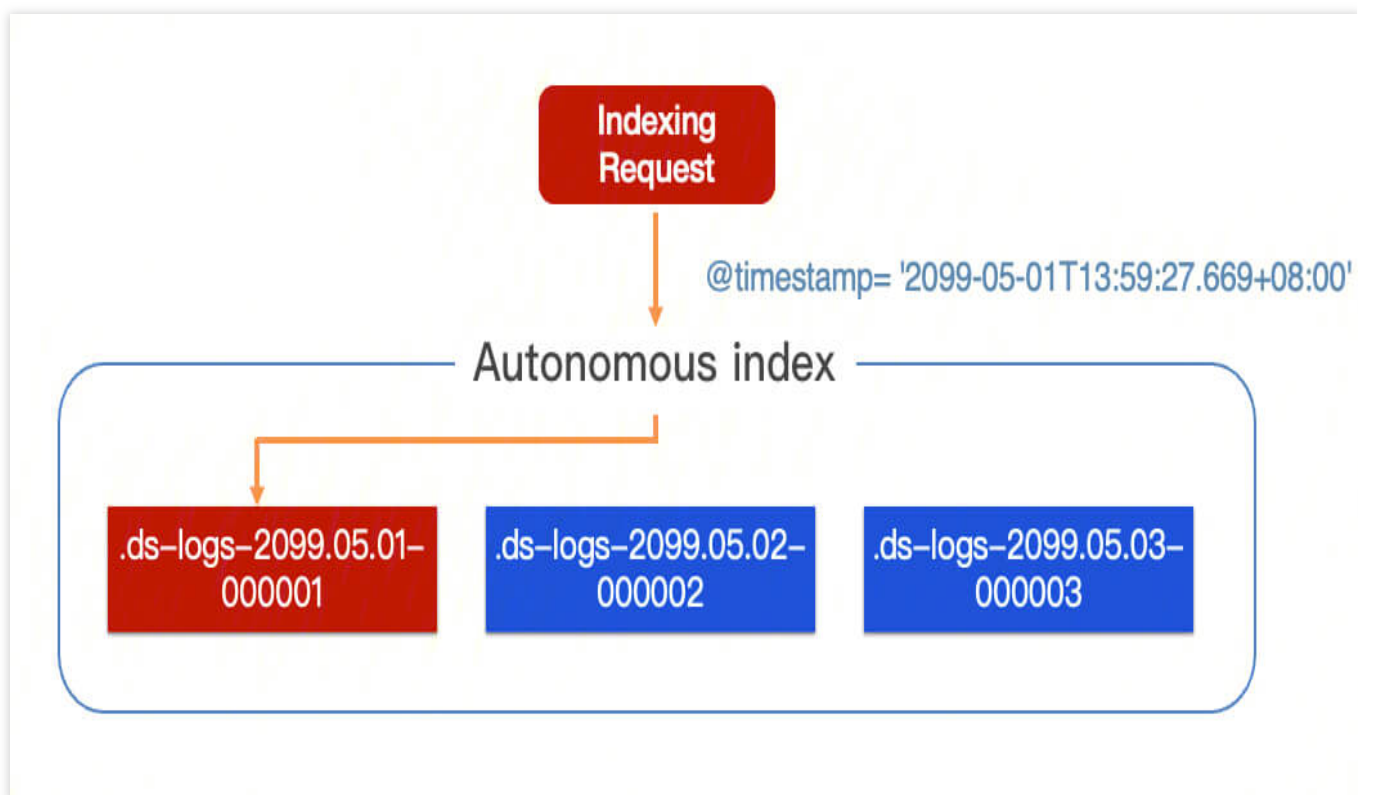
Write request

Write requests committed to an autonomous index will be routed to the latest backing index in append write mode or the backing index corresponding to the data time in shard-based write by time mode.

1.1 Append write

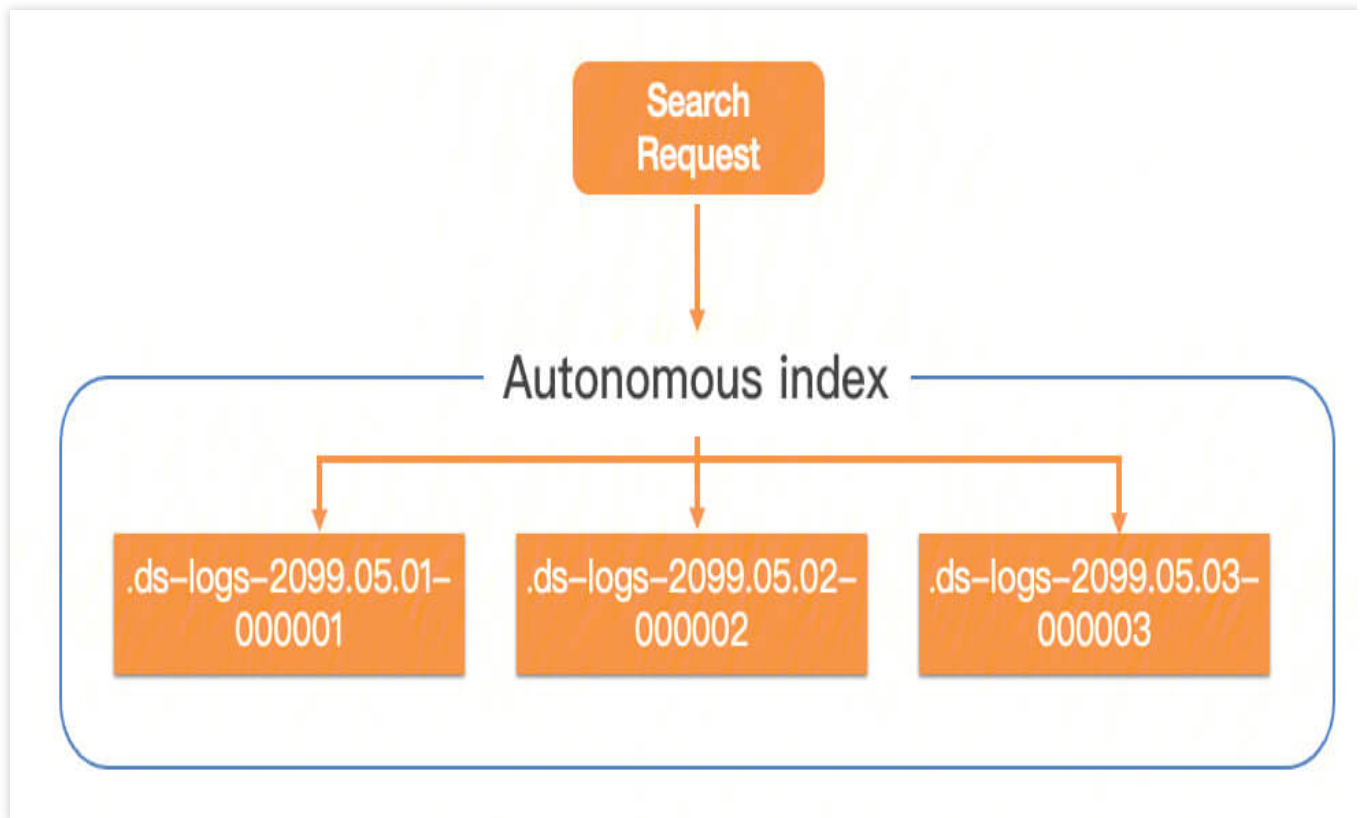


1.2 Shard-based write by time



Query request

Query requests committed to an autonomous index will be forwarded to all backing indices.



Rolling update

A rolling update will create a new backing index for the autonomous index. Currently, two rolling update methods are supported:

1.1 Automatic rolling update: It is implemented through the built-in feature of the autonomous index. When the rollover cycle condition configured for the autonomous index is met, or when the node of the backing index currently providing the write service fails, the new backing index will be rolled over automatically.

1.2 Manual rolling update: It is implemented through the [rollover API](#).

Index lifecycle management

This is implemented through Elasticsearch's ILM feature. You can directly configure ILM policies for an autonomous index with no need to manage policies and associated index templates. Elasticsearch's all ILM policies are supported.

Index shard quantity management

This is implemented through the built-in feature of the autonomous index. It promptly and stably adjusts the number of index shards in response to the changes in the real-time write load. You don't need to worry about the write availability issue caused by insufficient index shards as well as the issue caused by too many cluster shards.

Directions

ES allows you to use and manage autonomous indices on easy-to-use GUIs. For more information, see [Creating Autonomous Index](#).

Common APIs

1. Autonomous index creation:

```
PUT /_data_stream/indexname
{
  "mappings": {
    "properties": {
    }
  },
  "settings": {
  },
  "policy": {
    "warm.actions.migrate": {},
    "warm.min_age": "3d"
  },
  "options": {
    "timestamp_field": "@timestamp",
    "expire.max_age": "100d",
    "expire.max_size": "1TB",
    "pre_create.enable": true,
    "rollover.max_age": "1h",
    "rollover.dynamic": true,
    "shard_num.dynamic": true,
    "write_mode": "time_partition"
  }
}
```

mappings

It is optional and used to set the ES index mapping like the mapping in an Elasticsearch index.

settings

It is optional and used to set the ES index settings like the settings in an Elasticsearch index.

policy

It is optional and used to set the ILM policy like the ILM in an Elasticsearch index, but in a simplified way.

options

Autonomous index attributes, including:

timestamp_field: Time field, which is customizable and optional and will be `@timestamp` by default if not specified.

expire.max_age: Retention period of time range shards. The unit can be set to `h` (hour) or `d` (day). The value can be one hour at the minimum and will be 0 (i.e., not to delete) by default if not specified.

expire.max_size: Maximum size of time range shards. When this value is exceeded, historical time range shards will be eliminated. The unit can be set to `b`, `kb`, `mb`, `gb`, `tb`, or `pb`. The value will be 0 (i.e., not to eliminate) by default if not specified.

precreate.enable: Whether to enable time range shard precreation. The value is `true` (yes) by default.

`rollover.max_age`: Rollover cycle of time range shards. The unit can be set to `h` (hour) or `d` (day). This value can be `1h` at the minimum and is `1d` by default. `-1` indicates not to roll over time range shards.

`rollover.dynamic`: Whether to enable the dynamic adjustment of the time range shard rollover cycle. The value is `true` (i.e., yes) by default.

`shard_num.dynamic`: Whether to enable the dynamic adjustment of the time range shard quantity. The value is `true` (i.e., yes) by default.

`write_mode`: Write mode of time range shards. Valid values: `append_only` (default): append write, where data will be written to the latest time range shard; `time_partition`: shard-based write by time, where data will be written to the time range shard corresponding to the data time.

2. Autonomous Index Delete:

```
DELETE /_data_stream/index_name
```

3. Autonomous Index Modification:

```
POST _data_stream/indexname/_update
{
  "options": {
    "expire.max_age": "30d"
  }
}
```

Apart from `options.timestamp_field` and `options.write_mode`, all other properties can be modified.

After the configuration is successfully modified, the lifecycle-related configurations will take effect in all backing indices. Other configurations, such as the number of shards, number of replicas, and field mappings, will take effect only in subsequent rolled-over backing indices and will not update existing ones.

4. Autonomous Index Inquiry, consistent with the usage of ordinary index inquiries:

```
GET indexname/_search
```

5. Write, consistent with the usage of ordinary index writes, supports both bulk and doc methods:

```
PUT /indexname/_bulk
{"create":{ }}
{"@timestamp": "2022-03-13T03:07:34.348+08:00","field1": "a"}
{"create":{ }}
{"@timestamp": "2022-03-24T10:51:34.348+08:00","field1": "a"}
```

6. Autonomous Index Scrolling:

```
POST indexname/_rollover
```

7. Query Autonomous Index Definition:

```
1.Query by specified name
GET _data_stream/indexname?include_define
2.Query all Autonomous Indexes
GET _data_stream?include_define
----
```

Among them, the `include_define` option means that the result includes the content of self-governing index attributes. If not specified, it will be consistent with the content returned by the community DataStream.

Creating Autonomous Index

Last updated : 2024-11-29 21:56:04

Prerequisites

You have a Tencent Cloud account. For more information on how to create an account, see [Signing Up](#).

You have created an ES cluster on v7.14.2. For more information on how to create a cluster, see [Creating Clusters](#).

Note:

Only [autonomous indices](#) can be created.

Developed by Tencent Cloud, the autonomous index feature is suitable for time series data use cases such as log analysis and Ops monitoring, and can achieve index lifecycle management and automatic sharding optimization, with improved read and write efficiency. This feature is naturally applicable to clusters on v7.14.2 created after June 1, 2022 and is supported for older clusters on this version after a cluster restart. To use this feature in clusters on earlier versions, upgrade them to v7.14.2 first.

Directions

Step 1. Go to the "Create" page

1. Log in to the [ES console](#) and click **Data Management** to enter the index list.
2. Click **Create index** to enter the **Create** page.

Step 2. Enter the basic information

Index name: It must contain 1–255 characters, excluding Chinese characters, uppercase letters, space, and some symbols (, / , * , ? , " , < , > , | , # , : , and ,), and it cannot start with -, _, +, or .

Cluster: Cluster of the index.

Step 3. Enter the index configuration information

Data source configuration

1. Field mapping

Dynamic creation: It is enabled by default. After it is enabled, the collected source data will be automatically parsed to generate the field mappings of the index.

Enter sample for automatic configuration: After disabling **Dynamic creation**, you can generate the field mappings of the index through the **Enter sample for automatic configuration** input box. After you enter a JSON-formatted data sample and click **Confirm**, the platform will automatically verify the data. If there is no problem, fields will be mapped to the field mapping table. The deduction rule and sample are as follows:

Deduction rule: When the field value is `true` or `false` , the mapping type is `boolean` ; when the field value is an integer, the mapping type is `long` ; when the field value is a floating point, the mapping type is `double` ; when the field value is a string with 36 or fewer characters, the mapping type is `keyword` ; when the field value is a string with more than 36 characters, the mapping type is `text` ; when the field value is a string in date format, the mapping type is `date` ; when the field value supports nesting, the mapping type is `object` .Enter the following JSON-formatted data sample in the **Enter sample for automatic configuration** input box:

```
{
  "bool_field": true,
  "date_field": "2022/01/26 00:00:00",
  "double_field": 3.14,
  "keyword_field": "This is a line of text that does not require word
segmentation",
  "long_field": 126,
  "object_field": {
    "sub_field": 2022
  },
  "text_field": "This is a line of text that requires word segmentation. Text
with more than 36 characters will be identified as requiring word segmentation
and defined as the text type"
}
```

The parsing result is as shown below:

Index configurationChange to JSOI

Data source configuration

Field mapping ☐ **Dynamic creation** Automatically parse the collected source data and create the field mappings of the index

Enter sample for automatic configuration

Field name	Field type	Include Chinese characters <small>i</small>	Enable index <small>i</small>	Enable statistics <small>i</small>	
<input type="text" value="bool_field"/>	<input type="text" value="boolean"/>	--	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="button" value="x"/>
<input type="text" value="date_field"/>	<input type="text" value="date"/>	--	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="button" value="x"/>
<input type="text" value="double_field"/>	<input type="text" value="double"/>	--	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="button" value="x"/>
<input type="text" value="keyword_field"/>	<input type="text" value="keyword"/>	--	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="button" value="x"/>
<input type="text" value="long_field"/>	<input type="text" value="long"/>	--	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="button" value="x"/>
<input type="text" value="object_field"/>	<input type="text" value="object"/>	--	--	--	<input type="button" value="x"/>
<input type="text" value="text_field"/>	<input type="text" value="text"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	--	<input type="button" value="x"/>

+ Add field

Field mapping splits the original data into multiple segments by field (i.e., key:value) for indexing and search as follows:

Parameter	Description
Field name	Name of the field in the written data
Field type	Field data type. Valid values: boolean, keyword, long, double, date, text. More field types are supported in the JSON mode. For more information, see Field data types .
Include Chinese characters	This feature can be enabled if the field contains Chinese characters and you want to search for Chinese characters, but it will increase the index size. After this feature is enabled, the <code>ik_max_word</code> analyzer will be applied to the text field by default.
Enable index	Enabling this feature allows for creating an index for this field for search.
Enable statistics	Enabling this option allows for statistical analysis of field values, but it increases the index size.

2. Time field

The time field refers to the date field in the data. This field records the data creation time and cannot be modified after the index is created.

Enable index and **Enable statistics** are enabled by default for the time field and cannot be disabled.

After dynamic creation is disabled, the time field will map the date fields in the table, and you can select one from the drop-down list as the time field.

3. Write mode

Data is written to the index. This mode cannot be changed after the index is created successfully. Currently, two write modes are supported:

Append write (suitable for the log case): Data will be written to the latest backing index.

Shard-based write by time (suitable for the monitoring case): Data will be written in the backing index of the corresponding time period based on the time field.

Lifecycle configuration

1. Storage by tier

Storage by tier indicates that you can store indices on nodes with different attributes based on their access frequency. For example, you can store less-frequently queried and updated data on a warm tier for cost savings. The time it takes to migrate an index to a tier is calculated from the index rolling update start.

2. Deletion upon expiry

The **Delete upon expiry** option supports deleting historical data based on the **Max age** and **Max size**.

Max age: If the write mode is **Append write**, the backing index will be deleted when the specified value is reached after the index creation time. If the write mode is **Shard-based write by time**, the backing index will be deleted when the specified value is reached after the time when no more data is written to the index.

Max size: When the size of the autonomous index reaches the specified value, a historical backing index will be deleted based on the applicable condition. If the write mode is **Append write**, the oldest backing index will be deleted, starting from the index creation time. If the write mode is **Shard-based write by time**, the oldest backing index will be deleted, starting from the time when no more data is written to the index.

Basic info

Index name

Cluster

[Create ES cluster](#)

Only v.7.14.2 clusters created on and after June 1, 2022 support the autonomous index feature. Those created before this date can support this feature upon restart.

Index configuration

[Change to JSON](#)

Data source configuration

Field mapping

☒ **Dynamic creation** Automatically parse the collected source data and create the field mappings of the index

Time field

The time field refers to the date field in the data. This field records the data creation time and cannot be modified after the index is created.

Write mode ⓘ

Lifecycle configuration

Storage by tier

Warm tier

☒ Store less-frequently queried and updated data on a warm tier for cost savings

Delete upon expiry

Max age

☒

Max size

☒

[Advanced settings](#)

Advanced settings

1. Creation parameters

Shard number: A shard is a partition of data stored in the index. If **Dynamic adjustment** is enabled, the shard quantity is only used as the initial value for the backing index and will be automatically adjusted to the optimal value by algorithms.

Shard quantity dynamic adjustment: The platform will automatically adjust the shard quantity based on the business load to keep the index under the best condition. The adjusted shard quantity will apply to backing indices that are rolled over subsequently but not to existing backing indices.

Replicas: The number of replica shards of each primary shard.

Refresh interval: The interval required for data to be searchable after it is written in the index.

2. Index rollover

Rollover cycle: From the creation date, roll the index over once in each specified cycle.

Rollover cycle dynamic adjustment: The platform will automatically adjust the rollover cycle based on the business load to keep the index under the best condition.

▲ Advanced settings

Creation parameters

Shard number ⓘ

Dynamic adjustment ☒

The platform will automatically adjust the shard number based on the business load to keep the index under the best condition

If the dynamic adjustment feature is enabled, the shard number here will be the initial trial value only, and the algorithms will automatically get the optimal one.

Replicas ⓘ

Refresh interval ⓘ

sec ▼

Index rollover

Rollover cycle ⓘ

Dynamic adjustment ☒

The platform will automatically adjust the rollover cycle based on the business load to keep the index under the best condition

JSON mode

1. Feature description

Currently, you can switch to the **JSON mode** by clicking **Change to JSON mode** in the top-right corner of **Index configuration** to create an index. After a successful switch, the configuration information will be automatically synced to the UI in the corresponding mode.

2. Description of common parameters

The autonomous index feature provides options and policies to help you quickly configure rolling update and lifecycle management. It is compatible with the native syntax of Elasticsearch settings and mappings. Common parameters are as described below:

Type	Parameter	Description	Instructions
settings	index.number_of_shards	Number of primary shards	It is of the numeric type and must be an integer greater than or equal to 1.
	index.number_of_replicas	Number of replicas	It is of the numeric type and must be an integer greater than or equal to 0.
	index.refresh_interval	Refresh interval	It is of the string type. The unit can be set to `d` (day), `h` (hour), `m` (minute), `s` (second), `ms` (millisecond), `micros` (microsecond), or `nanos` (nanosecond). For example, `30s` indicates to set the refresh interval to 30 seconds.

mappings	field	Field name	It is of the string type, cannot contain Chinese characters, and must be unique.
	type	Field type	It is of the string type and can be set to `boolean`, `keyword`, `long`, `double`, `date`, or `text`. For more information, see Field data types
	analyzer	Analyzer	It is of the string type and can be an analyzer in the ES cluster, such as `ik_max_word`.
	index	Index status	It is of the boolean type and can be set to `true` or `false`.
	doc_values	Statistics status	It is of the boolean type and can be set to `true` or `false`.
options	pre_create.enable	Index precreation	It is of the boolean type and can be set to `true` (default) or `false`. Note that if it is disabled when the write mode is Append write , backing indices will not be rolled over based on the specified rollover cycle.
	rollover.dynamic	Rollover cycle dynamic adjustment	It is of the boolean type and can be set to `true` or `false`.
	rollover.max_age	Rollover cycle	It is of the string type. The unit can be set to `d` (day) or `h` (hour). For example, `1d` indicates to set the rollover cycle to one day.
	shard_num.dynamic	Shard quantity dynamic adjustment	It is of the boolean type and can be set to `true` or `false`.
	write_mode	Write mode	It is of the string type. `append_only` indicates `Append write`, and `time_partition` indicates `Shard-based write by time`.
	expire.max_age	Max age for deletion upon expiry	It is of the string type. The unit can be set to `d` (day) or `h` (hour). For example, `1d` indicates to set the max age to one day.
	expire.max_size	Max size for deletion	It is of the string type. The unit can be set to `PB`, `TB`, `GB`, `MB`, `KB` or `B`. For example, `1TB`

		upon expiry	indicates to set the max size to 1 TB.
policy	warm.actions.migrate	Index migration settings for the warm phase	See Migrate
	warm.min_age	Storage period before migration to the warm tier	It is of the string type. The unit can be set to `d` (day) or `h` (hour). For example, `1d` indicates to migrate the index to the warm tier one day after the start of the index rolling update.
	cold.actions.migrate	Index migration settings for the cold phase	See Migrate
	cold.min_age	Storage period before migration to the cold tier	It is of the string type. The unit can be set to `d` (day) or `h` (hour). For example, `1d` indicates to migrate the index to the cold tier one day after the start of the index rolling update.

Index configuration

[Change to the form](#)

```
1 {
2   "mappings": {
3     "properties": {}
4   },
5   "options": {
6     "expire.max_age": "120d",
7     "expire.max_size": "1tb",
8     "rollover.dynamic": true,
9     "shard_num.dynamic": true,
10    "timestamp_field": "@timestamp",
11    "write_mode": "append_only"
12  },
13  "policy": {
14    "warm.actions.migrate": {},
15    "warm.min_age": "7d"
16  },
17  "settings": {
18    "index.number_of_replicas": 1,
19    "index.number_of_shards": 1,
20    "index.refresh_interval": "30s"
21  }
22 }
```

Creation completion

Click **Confirm**. After the index is created successfully, you will be redirected to the **index list** in which the index is included.

Subsequent Operations

Index search and analysis

Data management allows you to redirect to the index search and analysis page. For more information, see [Index Search and Analysis](#).

Basic index information

Data management allows you to view the index information and manage backing indices in the console. For more information, see [Basic Index Information](#).

Index monitoring

Data management provides a rich set of index monitoring metrics to help you view the real-time data of indices during use. For more information, see [Index Monitoring](#).

Index configuration management

Data management enables you to flexibly modify the index configuration information in the console in response to business changes. After successful modification, lifecycle configurations will apply to all backing indices, and configurations of other items will take effect only in those rolled over later and will not update existing ones. For more information, see [Index Configuration Management](#).

Index Search and Analysis

Last updated : 2022-06-29 12:15:52

ES comes with the Kibana module. You can search for and analyze index data in Kibana. You can also quickly access Kibana in the index list for data search and analysis.

Search and Analysis

The data management feature provides entries for quick access to the search and analysis pages. In the index list, click an entry to open the Kibana login page. After successful login, you can enter the corresponding page.

Create

Analyze

Cluster

Cluster

Enter an index nam

Index name	Index status	Index size	Storage by tier	Max size	Time field	Creation time	Operation
test10	Green	416 B	/	Permanently stored	d10	2022-06-23 11:40:33	Search Monitoring More
test9	Green	416 B	/	Permanently stored	d9	2022-06-23 11:40:22	Search Monitoring More
test8	Green	416 B	/	Permanently stored	d8	2022-06-23 11:40:11	Search Monitoring More
test7	Green	416 B	/	Permanently stored	d7	2022-06-23 11:39:58	Search Monitoring More
test6	Green	416 B	/	Permanently stored	d6	2022-06-23 11:39:47	Search Monitoring More
test5	Green	416 B	/	1TB	d5	2022-06-23 11:39:28	Search Monitoring More
test4	Green	416 B	/	120 day(s)	d4	2022-06-23 11:39:14	Search Monitoring More
test3	Green	416 B	/	Permanently stored	d3	2022-06-23 11:38:58	Search Monitoring More
test2	Green	416 B	Hot / Warm	120 day(s)	d2	2022-06-23 11:38:41	Search Monitoring More
test1	Green	416 B	Hot / Warm	120 day(s) / 1TB	d1	2022-06-23 11:08:17	Search Monitoring More

Total items: 12

10 / page

1

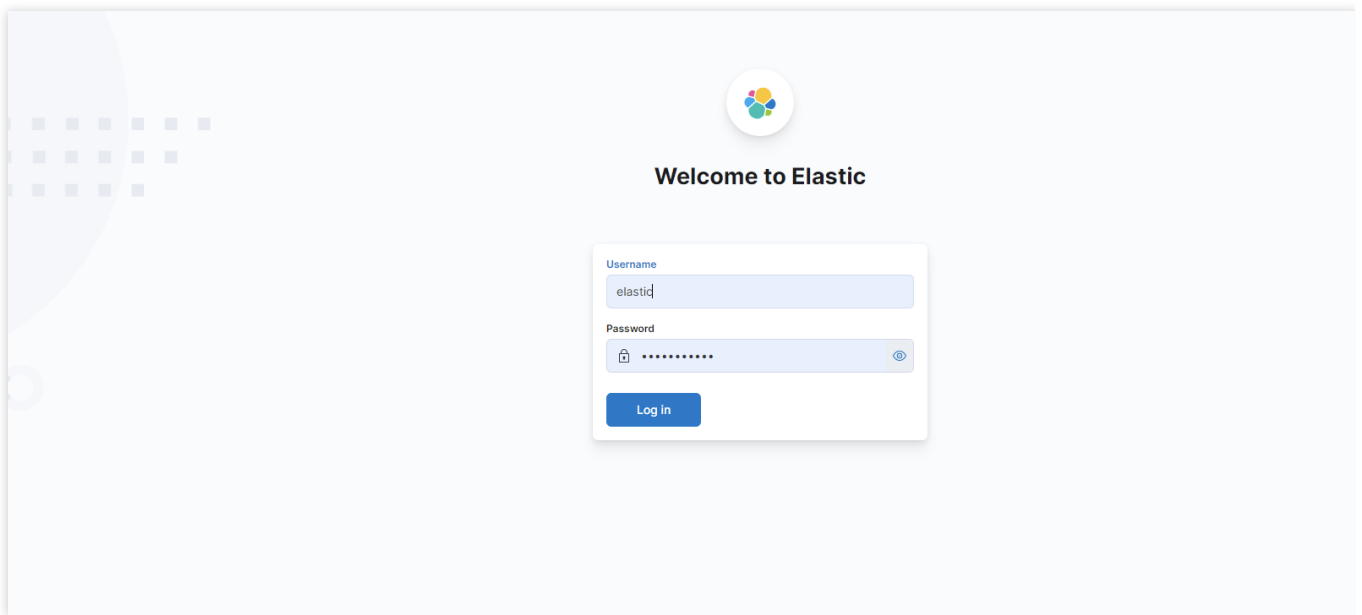
/ 2 pages

Note:

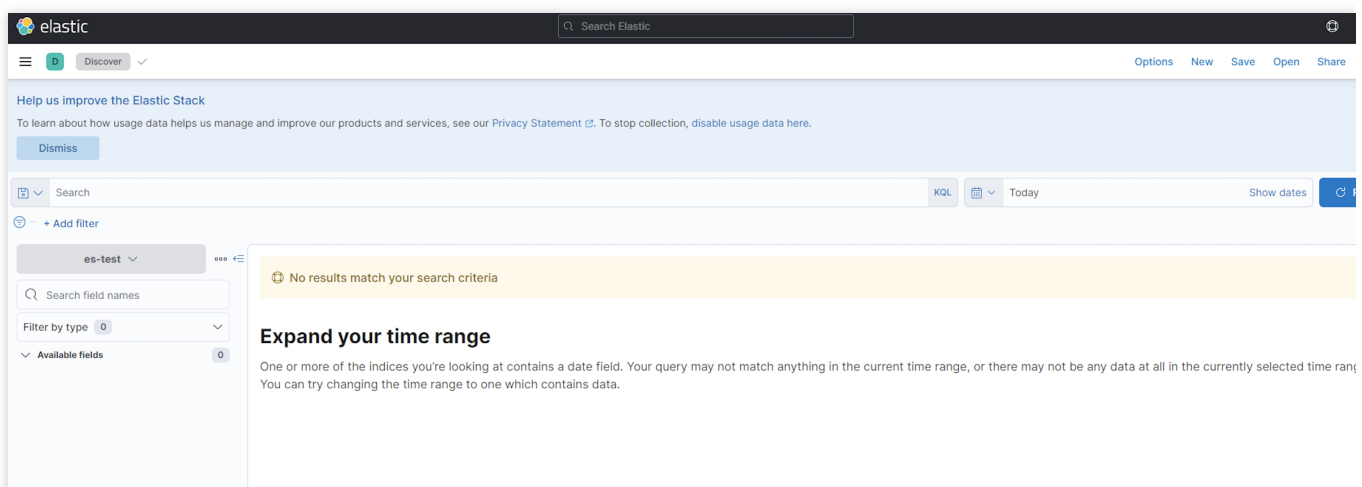
Click **Search** to enter the **Discover** page of the index. Click **Analyze** to enter the **Dashboard** page.

Login

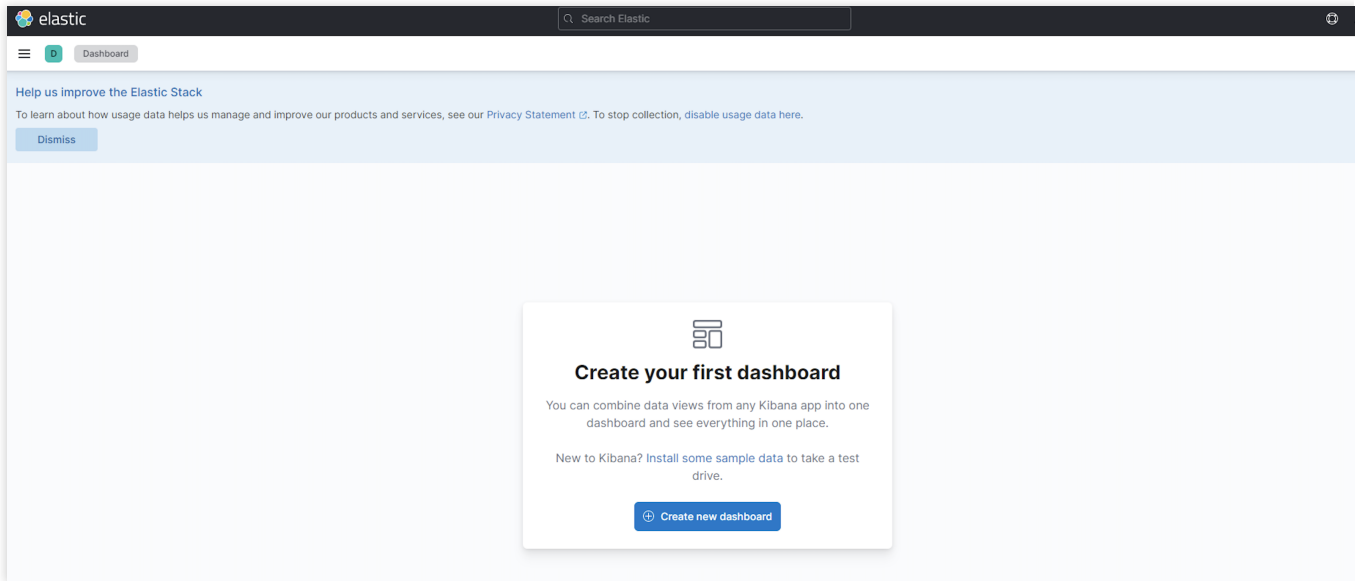
To access the Kibana page, you need to log in with the username "elastic" and the Kibana password you set when you created your cluster. If you forgot your password, you can reset it on the cluster details page. For security reasons, you can configure an access blocklist/allowlist for the public address of the Kibana page. For more information, see [ES Cluster](#).



After successful login, you will be redirected to the **Discover** page. Index patterns have been created for autonomous indices by index name by default, so you can directly perform data search:



After successful login, you will be redirected to the **Dashboard** page.



Basic Index Information

Last updated : 2024-11-29 21:58:05

The data management feature allows you to view basic information such as index status, cluster, and Elasticsearch version and manage [backing indices](#).

Directions


1. Log in to the [ES console](#).
2. Select the target cluster in **Data Management**. Then, click an **Index name** in the index list to enter the **Basic Information** page of the index.

Basic Information

Basic information

The **basic information** module displays information such as index name, index status, index type, cluster, Elasticsearch version, index size, and creation time. The index size is obtained by adding the size of all backing indices in the index.

Basic info

Index name	test1
Index type	Autonomous index
Index status	Green
Cluster	
Elasticsearch version	7.14.2
Index size	416 B
Creation time	2022-06-23 11:08:17

Configuration information

The **configuration information** module displays information such as write mode, time field, and index lifecycle settings.

Configuration info

Write mode ⓘ	Append write (suitable for the log case)
Time field	d1
Storage on Warm tier	From rolling update, migrate the data to Warm tier 7 day(s) later
Max size	120 day(s) / 1TB

Backing index management

You generally don't need to care about backing index, as it is the underlying implementation logic of autonomous index. If you want to pay more attention to and manipulate backup index, see the following:

The **backing index management** module allows you to view the information of backing indices, including index name, index status, index size, current lifecycle phase, and creation time, as well as delete backing indices.

Note:

The latest backing index in an autonomous index and the backing index being written cannot be deleted.

▼ Backing index (advanced)						About backing i
Index name ⚙	Index status	Index size ⚙	Current lifecycle phase ▾	Creation time ⚙	Operation	
.ds-test1-2022.06.23-000001	Green	416 B	● Hot phase	2022-06-23 11:08:17	Delete	
Total items: 1				10 ▾ / page	⏮ ⏪ 1 ⏩ ⏭ / 1 page	

A rolling update will create a new backing index for the autonomous index. Currently, two rolling update methods are supported:

Automatic rolling update: It is implemented through the built-in feature of the autonomous index. When the rollover cycle condition configured for the autonomous index is met, or when the node of the backing index currently providing the write service fails, the new backing index will be rolled over automatically.

Manual rolling update: It is implemented through the [rollover API](#).

Index Monitoring

Last updated : 2024-11-29 21:59:21

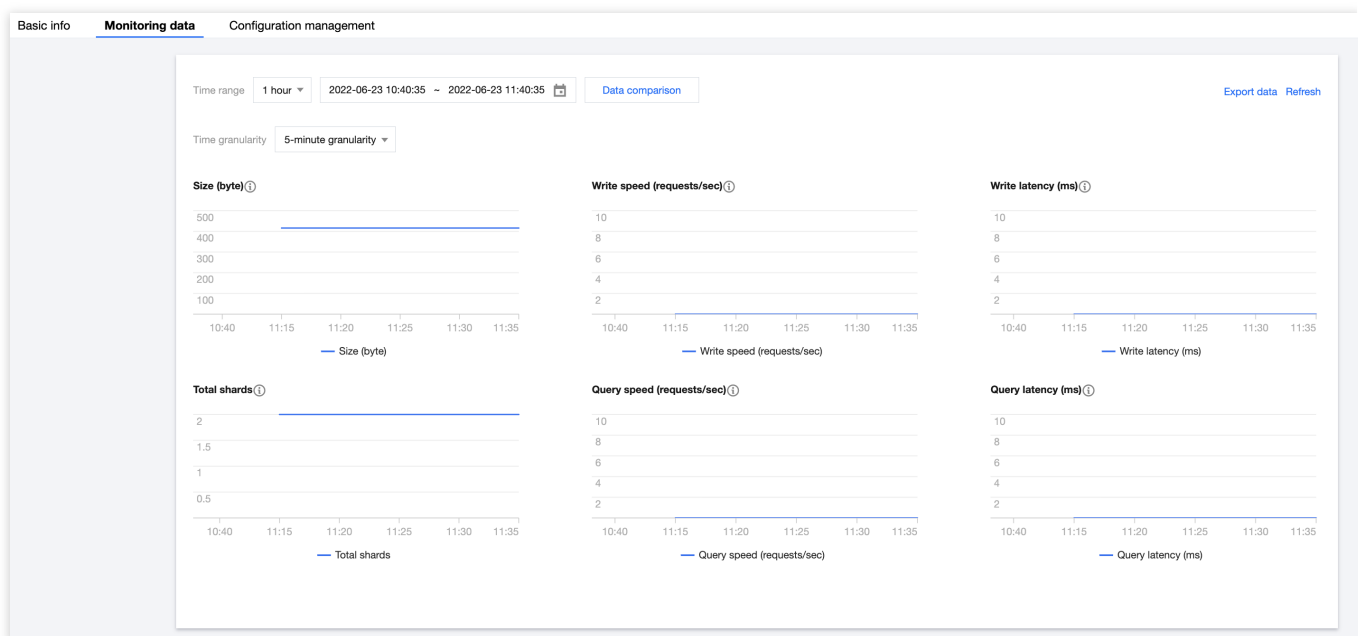
The data management feature provides many monitoring metrics for indices in an ES cluster to monitor index conditions, such as storage, write, and query. Based on these metrics, you can understand the index usage in real time and promptly handle possible risks to ensure stable index operations. This document describes how to view monitoring data through data management.

Directions

1. Log in to the [ES console](#).
2. Select the target cluster in **Data Management**. Then, click an **Index name** in the index list to enter the **Basic Information** page of the index.
3. Select the **Monitoring Data** page to view the overall index usage.

Monitoring data

Monitoring metrics cover six dimensions: size, total shards, write speed, write latency, query speed, and query latency.



Descriptions of metrics

The statistical period of each metric is five minutes; that is, the cluster's metrics are collected once every five minutes.

The metrics are as described below:

Monitoring Metric	Statistical Method	Details
Size	The average index size during a statistical period.	This value is the total size of a single index or all backing indices for an autonomous index. You can view the index storage capacity change based on this value.
Total Shards	The average number of shards in the index during a statistical period.	This value is the total number of all shards of a single index or all backing indices for an autonomous index, including primary and replica shards.
Write Speed	The average number of index requests received by indices or autonomous indices per second during a statistical period. Calculation rule for the number of index requests per second of an index: the total number of historical index requests of all index shards (<code>_cat/shard?h=indexing.index_total</code>) is recorded once every statistical period (five minutes), and the difference between two adjacent records (i.e., the absolute value in one statistical period) is taken for calculation (number of index requests / 300 seconds) to get the average number of index requests per second in one statistical period.	-
Query Speed	The average number of search (query) requests received by indices or autonomous indices per second during a statistical period. Calculation rule for the number of search requests per second of an index: the total number of historical search requests of all index shards (<code>_cat/shard?h=search.query_total</code>) is recorded once every statistical period (five minutes), and the difference between two adjacent records (i.e., the absolute value in one statistical period) is taken for calculation (number of search requests / 300 seconds) to get the average number of search requests per second in one statistical period.	-
Write Latency	Write latency (<code>index_latency</code>) refers to the average time taken by a single index request (ms/request). Calculation rule for the average index request time: two metrics are recorded once every statistical	Write latency is the average time it takes to write a single

	<p>period (five minutes), i.e., total time taken by historical index requests of all index shards (<code>_cat/shard?h=indexing.index_time</code>) and total number of historical index requests of all index shards (<code>_cat/shard?h=indexing.index_total</code>), and the difference between two adjacent records (i.e., the absolute value in one statistical period) is taken for calculation (index request time / number of index requests) to get the average index request time in one statistical period.</p>	<p>document. If the write latency is too high, we recommend you increase the number of index shards or nodes or upgrade the cluster node specification.</p>
Query Latency	<p>Query latency (<code>search_latency</code>) refers to the average time taken by a single search request (ms/request). Calculation rule for the average search request time: two metrics are recorded once every statistical period (five minutes), i.e., total time taken by historical search requests of all index shards (<code>_cat/shard?h=search.query_time</code>) and total number of historical search requests of all index shards (<code>_cat/shard?h=search.query_total</code>), and the difference between two adjacent records (i.e., the absolute value in one statistical period) is taken for calculation (search request time / number of search requests) to get the average search request time in one statistical period.</p>	<p>Query latency is the average time it takes to perform a single query. If the query latency is too high, we recommend you perform query optimization based on query profile, upgrade the cluster node specification, or increase the number of nodes.</p>

Index Configuration Management

Last updated : 2024-11-29 21:59:50

Overview

The data management feature allows you to view and manage indices. You can view the configuration of an index or modify the index configuration on the **Configuration management** page to quickly adapt to business changes.

Directions

1. Log in to the [ES console](#).
2. Select the target cluster in **Data Management**, click **More** in the index list, and select **Configuration management** in the drop-down list to enter the index configuration management page.

View mode

After entering this page, the view mode is selected by default, where the information of data source configuration, lifecycle configuration, and advanced settings is displayed. You can click **Change to JSON mode** in the top-right corner to view the current index configuration in JSON format.

Index configuration[Change to JSON](#)

Data source configuration

Field mapping	Field name	Field type	Include Chinese characters ⓘ	Enable index ⓘ	Enable statistics ⓘ
The dynamic creation feature is enabled, and source data will be automatically parsed to create the field mappings of the index.					

Time field: d1

Write mode ⓘ: Append write (suitable for the log case)

Lifecycle configuration

Storage by tier

Warm tier

☒ Store less-frequently queried and updated data on a warm tier for cost savings

From rolling update: Migrate the data to Warm tier 7 day(s) later

Delete upon expiry

Max age

☒

From rolling update: 120 day(s)

Max size

☒

From rolling update: 1 TB

[Advanced settings](#)[Modify configuration](#)

Index configuration

Configurations synchronized [Change to the form](#)

Current configuration

```
1 {
2   "mappings": {
3     "properties": {}
4   },
5   "options": {
6     "expire.max_age": "120d",
7     "expire.max_size": "1tb",
8     "rollover.dynamic": "true",
9     "rollover.max_age": "30d",
10    "shard_num.dynamic": "true",
11    "timestamp_field": "d1",
12    "write_mode": "append_only"
13  },
14  "policy": {
15    "warm.actions.migrate": {},
16    "warm.min_age": "7d"
17  },
18  "settings": {
19    "index.number_of_replicas": "1",
20    "index.number_of_shards": "1",
21    "index.refresh_interval": "30s"
22  }
23 }
```

[Modify configuration](#)

UI edit mode

In the view mode, click **Modify Configuration** in the bottom-left corner to enter the edit mode, where you can modify the information of the index configuration. After successful modification, lifecycle configurations will apply to all backing indices, and configurations of other items such as shard number, replica shards, and field mappings will take effect only in those rolled over later and will not update existing ones.

Note:

The time field and write mode cannot be modified.

Index configuration

Change to JSON

Data source configuration

Field mapping

Field name	Field type	Include Chinese characters	Enable index	Enable statistics
The dynamic creation feature is enabled, and source data will be automatically parsed to create the field mappings of the index.				
<div>+ Add field</div>				

Time field

d1

Write mode

Append write (suitable for the log case)

Lifecycle configuration

Storage by tier

Warm tier

Store less-frequently queried and updated data on a warm tier for cost savings

From rolling update:

Migrate the data to Warm tier

7

day(s)

later

Delete upon expiry

Max age

From rolling update:

120

day(s)

Max size

From rolling update:

1

TB

Advanced settings

Confirm

Cancel

JSON mode

After switching to the JSON edit mode, the left side is the current configuration of the running index, and the right side is the **Modify Configuration** input box where you can enter the modified configuration information. Corresponding index configuration items will be updated after the modification is saved successfully.

Index configuration

[Change to the form](#)

Current configuration

```
1 {
2   "mappings": {
3     "properties": {}
4   },
5   "options": {
6     "expire.max_age": "120d",
7     "expire.max_size": "1tb",
8     "rollover.dynamic": "true",
9     "rollover.max_age": "30d",
10    "shard_num.dynamic": "true",
11    "timestamp_field": "d1",
12    "write_mode": "append_only"
13  },
14  "policy": {
15    "warm.actions.migrate": {},
16    "warm.min_age": "7d"
17  },
18  "settings": {
19    "index.number_of_replicas": "1",
20    "index.number_of_shards": "1",
21    "index.refresh_interval": "30s"
22  }
}
```

Modify configuration

```
1 {
2   "mappings": {
3     "properties": {}
4   },
5   "options": {},
6   "policy": {},
7   "settings": {}
8 }
```

Confirm

Cancel

As shown in the figure, you can change the storage time before data is moved to the warm tier from 2 hours to 2 days. Click **Confirm**, and the configuration will be updated.

Index configuration

[Change to the form](#)

Current configuration

```
1 {
2   "mappings": {
3     "properties": {}
4   },
5   "options": {
6     "expire.max_age": "120d",
7     "expire.max_size": "1tb",
8     "rollover.dynamic": "true",
9     "rollover.max_age": "30d",
10    "shard_num.dynamic": "true",
11    "timestamp_field": "d1",
12    "write_mode": "append_only"
13  },
14  "policy": {
15    "warm.actions.migrate": {},
16    "warm.min_age": "2h"
17  },
18  "settings": {
19    "index.number_of_replicas": "1",
20    "index.number_of_shards": "1",
21    "index.refresh_interval": "30s"
22  }
}
```

Modify configuration

[Format J](#)

```
1 {
2   "mappings": {
3     "properties": {}
4   },
5   "options": {},
6   "policy": {
7     "warm.actions.migrate": {},
8     "warm.min_age": "2d"
9   },
10  "settings": {}
11 }
```

Confirm

Cancel