BES Document

2020-11-17

Contents

Feature Release Records

Release time	Feature Overview
2019-06	• Added the HybridFS support. You can use the “mmap” for access to the nvd, dvd, tim, and cfs files, and “niofs” for other files, thus effectively improving the read performance. • The built-in segmentation plug-in can add “jieba” word segmentation. • Fixed the bugs that “ik” segmentation plug-in cannot dynamically update the thesaurus.
2019-02	• Connected the BES to the IAM system, and added Identity and Access Management (IAM) feature. Support setting of the read-only, OPS and management privileges for sub-users. • Support the cluster resource evaluation. The user can estimate the disk space according to index inflation, ES internal cost, emergency cost, and the number of replicas. Also, the user can estimate the number of nodes by combining the selected package. • When the thesaurus of IK does not meet the requirements, the user can define the thesaurus.
2019-01	• List of integrated plug-ins: repository-bos, baidu-pack, anylasis-ik, anylasis-pinyin, ingest-attachment, elasticsearch-sql, and so on.

	• http.cors.enabled: false http.cors.allow-origin: "*", some ES-related independently deployed Web services may need cross-region support.
2018-10	Ik analyzer supports the dynamic setup of thesaurus.
2018-08	• Modified the default “cfs ratio” from 0.1 to 0, i.e., the “cfs” is forbidden under “merge” by default. • Updated the scheduling cycle of the "index state" and "node state" to 5 min when modifing the "cluster info service". • Changed the size of thread pools, Index, Bulk, GET, and Search so that they can be subject to dynamic adjustment. • Added a plug-in, Pingyin Segmentation. • Support the backup and recovery of the BOS-based object storage system. • Support the “http basic” based authorization verification.

Product Description

Product Introduction

Baidu Elasticsearch(BES) is a hosting service that allows you to easily deploy, operate and expand in Baidu AI Cloud. Elasticsearch is a very popular open source distributed retrieval and analysis engine. You can create and configure the BES cluster within several minutes through the Baidu AI Cloud management console. The service automatically detects and replaces the failed Elasticsearch nodes, reducing the costs associated with self-managing infrastructure and Elasticsearch software. BES cloud service is fully compatible with the Elastosearch open source API, facilitating zero-cost migration of your existing Elastosearch business. Meanwhile, BES cloud service provides a privilege management mechanism, for you to freely configure the cluster privileges, so as to further ensure data security.

Basic Concept

Index

The Elasticsearch allows you to store data in one or more indexes. The index is a set of documents with similar characteristics. Compared with the traditional relational database domain, the index is equivalent to a database or a data storage schema in SQL. You can identify the index by its name (which must be all lowercase characters). You can also create, search, update, and delete the document by referencing this name. You can create any number of indexes as needed in an Elasticsearch cluster.

Type

Type is a logical partition within the index, but its meaning depends on the user requirements. So, the user can define one or more types within an index. Generally, the type is the predefinition for documents with the same domain. For example, in the index, you can define a type for storing user data, a type for storing log data, and a type for storing comment data. Compared with the traditional relational database domain, the type is equivalent to a table. But the ES weakens the concept of type. There is only one type in an index in the future.

Document

The document is the atomic unit of the Lucene index and search. It is a container containing one or more domains and is expressed based on JSON format. A document consists of one or more domains, each with a name and one or more values. Domains with multiple values are usually called multiple-value domains. Each document can store different domain sets, but documents of the same type shall have some similarities.

Mapping

In the Elasticsearch, all documents must be subject to analysis before being stored. The user can determine how to divide the text into tokens, which tokens should be filtered out, and which texts need additional processing, and so on, according to their requirements. Also, the Elasticsearch provides additional features, such as sorting the contents in the domain as needed. In fact, the Elasticsearch can automatically determine the type of domain based on its value.

Cluster

The Elasticsearch cluster is a set of one or more nodes that store the entire data set and provide federated indexes and search capabilities across all nodes. The cluster formed by multiple nodes has redundancy capability. It can ensure the overall availability of services when one or more nodes fail. The cluster is identified by its unique name, and the default name is elasticsearch. The node determines the Elasticsearch cluster to join by its cluster name, and one node can only be subordinate to one cluster. Even if you do not consider the redundancy and other characteristics, the Elasticsearch cluster with only one node can also realize all storage and search features.

Node

The Elasticsearch host running a single instance is called a Node. It is a member of the cluster and can store data, join in cluster index and search operations. Similar to a cluster, a node is identified by its name, which defaults to a random Marvel character name generated at startup. The user can customize any name they want as needed. But, the name should be as recognizable as possible for the management purpose. The node determines the cluster to join by the Elasticsearch cluster name configured for it.

Shard

The shard mechanism of the Elasticsearch can store the data within an index in multiple nodes in a distributed manner. It divides an index into multiple underlying physical Lucene indexes to complete the split storage function of index data. Each physical Lucene index is called a shard. Each shard is internally a fully functional and independent index. Thus, it can be stored by any host in the cluster. When creating an index, the user can specify the number of shards, which is 5 by default. There are two types of shards, i.e., primary shard and replica shard. The primary shard is available for document storage.

Under each new index, create 5 primary shards are automatically. You can define this number by configuration before index creation. However, with the index created, the number of the primary shards do not change any longer. The replica shard is a copy of the primary shard, which is used for redundant data and improving search performance. One replica shard is configured for each primary shard by default. However, you can also configure multiple replica shards. And the number of replica shards changes dynamically. The Elasticsearch automatically increases or decreases the number of replica shards as needed. The Elasticsearch cluster consists of multiple nodes. You can store all the shards on these nodes in a distributed manner.

Key Characteristics

Quick Deployment

By using Baidu Open Cloud Management Console, you can deploy the Elasticsearch cluster in several minutes without needing complicated configuration.

Dynamic Capacity Expansion

According to business requirements, the cluster scale can be dynamically expanded at any time through BCE console to meet business growth and make full use of cluster resources.

Security Isolation

Each user has an independent cluster, with free management, resource isolation and data security.

Access Control

Add the privilege management module for the Elasticsearch cluster for users to conveniently control the privileges of datasheets and to further improve data security.

Simplicity and Easy Use

Using RESTful interface, HTTP protocol and JSON format data, it is fully compatible with the official API of Elasticsearch. It is convenient for users to connect through any HTTP client.

High Availability

Real-time monitoring of the health status of the cluster and each node, automatic alarm and recovery when problems are found to ensure high availability of the cluster.

Powerful Data Retrieval and Analysis Capabilities

Relying on the open source distributed retrieval and analysis engine Elasticsearch, it provides users with powerful data retrieval and analysis functions.

High performance

Baidu expert team makes in-depth optimization to ensure the cluster performance.

Economy and Efficiency

Pay on demand, low in price.

Product Pricing

Product Pricing

This document introduces product pricing, billing rules, and expiration reminders.

On-demand billing

Baidu Elasticsearch products are charged by the service time (minute), according to the instance model configuration and number of the instances you select. The specific billing rules are as follows:

- In the case of being paid by the minute, less than 1 minute is counted as 1 minute.
- Start billing when the order is submitted, and the cluster instance has been assigned.
- Release all resources and stop billing as soon as the cluster gets deleted.
- When stopping the cluster, the fee is still charged because the resources have not been released. To stop charging, you must delete the cluster and release the resources.
- Currently, there is no billing for external network traffic.

Insufficient balance and arrears

- ***Reminder of insufficient balance*** : - Judge whether your account balance, including available vouchers, is sufficient to pay for the next 3 days in accordance with your bill amount in the last 3 days. If not, the system sends a renewal reminder to you. - Judge whether your account balance, including available vouchers, is sufficient to pay the bill for 1 day according to your billing amount of the latest 1 day. If insufficient, the system sends the renewal reminder to you.
- ***Handling with the arrears*** : - When your account balance is zero, and you cannot pay the Baidu Elasticsearch service bill, you are in arrears, and the system sends a renewal reminder. - The service is stopped immediately after arrears, then the system sends you a renewal reminder for stopping service. In order not to affect service, make sure that the amount in your account is sufficient when using the Baidu Elasticsearch service. - After 7 days of being in arrears, the system deletes the service data, and release the data if you do not recharge it.

Postpaid billing formula

Fee = unit price x number of nodes x service time

Postpaid price

Package name	Node type	CPU (Number of cores)	Memory (GB)	Disk	Unit price (CNY/minute/node)
balance type 1	Elasticsearch Node	2	10	150 GB SSD	0.02
balance type 2	Elasticsearch Node	6	30	500 GB SSD	0.06
storage type 1	Elasticsearch Node	4	16	500 GB SSD cloud disk	0.08
storage type 2	Elasticsearch Node	8	32	1 TB SSD cloud disk	0.15
storage type 3	Elasticsearch Node	16	64	2 TB SSD cloud disk	0.30
computing type 1	Elasticsearch Node	4	32	100 GB SSD cloud disk	0.06
computing type 2	Elasticsearch Node	16	64	200 GB SSD cloud disk	0.15
computing type 3	Elasticsearch Node	32	128	500 GB SSD cloud disk	0.30

Prepaid Price

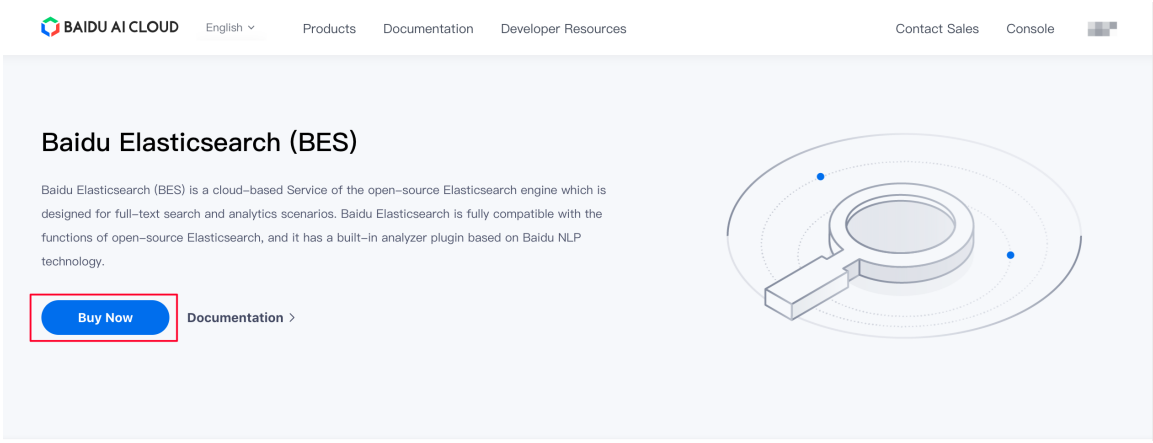
Package name	Node type	CPU (Number of cores)	memory (GB) /th>	Disk	Unit price (CNY/month/node)
balance type 1	Elasticsearch Node	2	10	150 GB SSD	520
balance type 2	Elasticsearch Node	6	30	500 GB SSD	1560
storage type 1	Elasticsearch Node	4	16	500 GB SSD cloud disk	1200
storage type 2	Elasticsearch Node	8	32	1 TB SSD cloud disk	2300
storage type 3	Elasticsearch Node	16	64	2 TB SSD cloud disk	4500
computing type 1	Elasticsearch Node	4	32	100 GB SSD cloud disk	1100
computing type 2	Elasticsearch Node	16	64	200 GB SSD cloud disk	2500
computing type 3	Elasticsearch Node	32	128	500 GB SSD cloud disk	5100

Getting Started

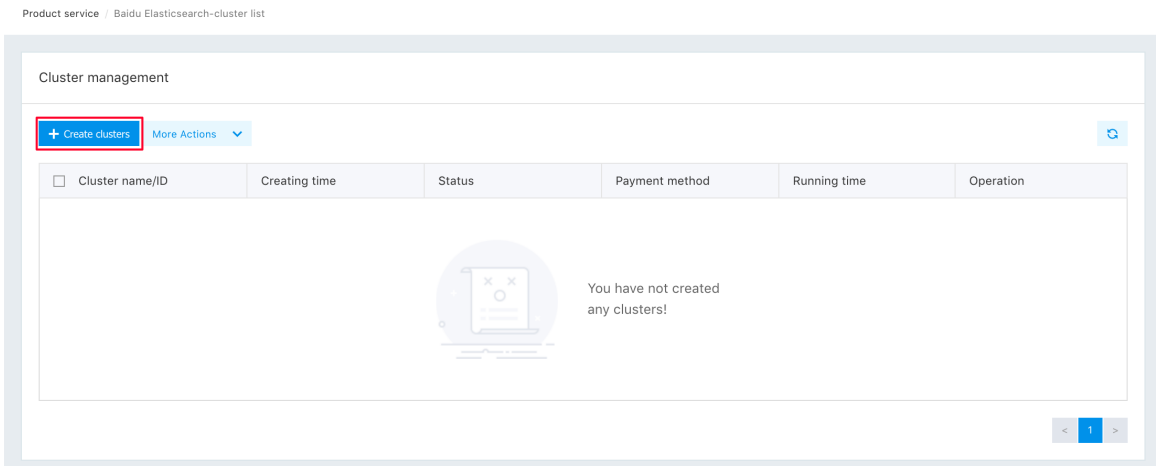
Create a Cluster

If you do not have a Baidu Open Cloud account, please register a Baidu account through this address https://cloud.baidu.com/index.html. Log in to Baidu Open Cloud with your account and enter the main menu of the console after registration.

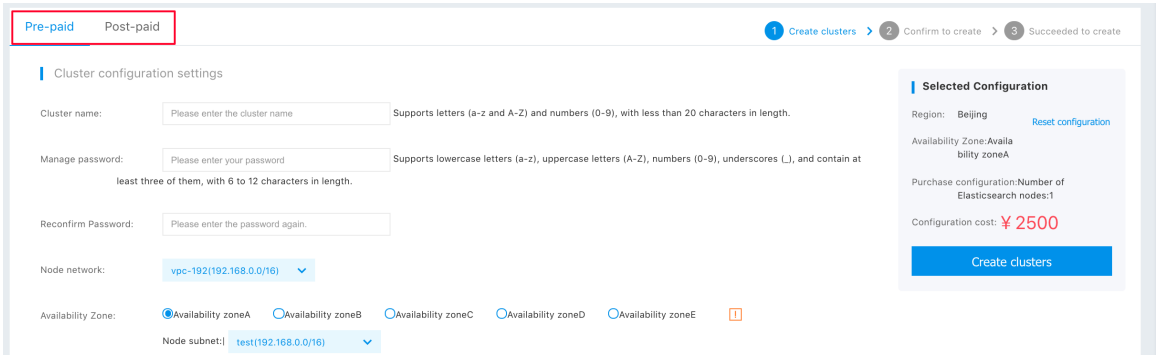
After entering, click “Shop Now”.



Enter the cluster list page:

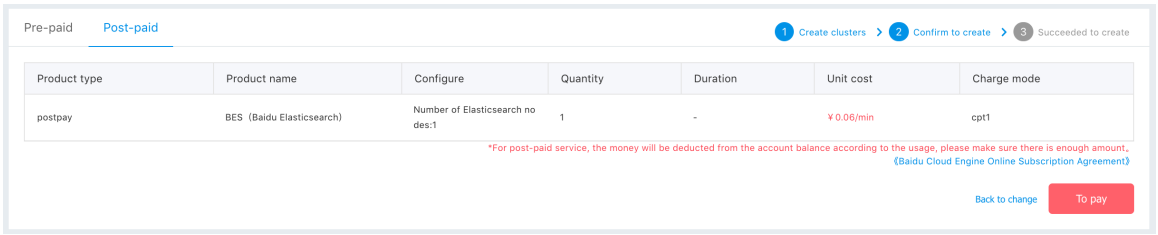


Click “Create Clusters” to enter the cluster configuration page:

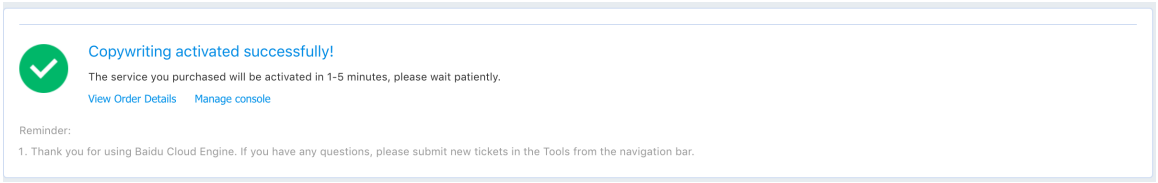


Note: Here, we provide two billing modes for users to select: Prepaid and postpaid. Prepaid cannot be deleted before expiration, and the cluster can be deleted at any time for postpaid mode. So we suggest that if to test the ES service, users shall select the postpaid mode. For long-term use, select the prepaid mode for its much lower price.

Enter the cluster name and the password of the administrator account “superuser”, select the number of Elasticsearch Nodes as required, and then click “Create Cluster” to enter the order confirmation page, and you need to confirm whether the price and payment method meet the expectation:

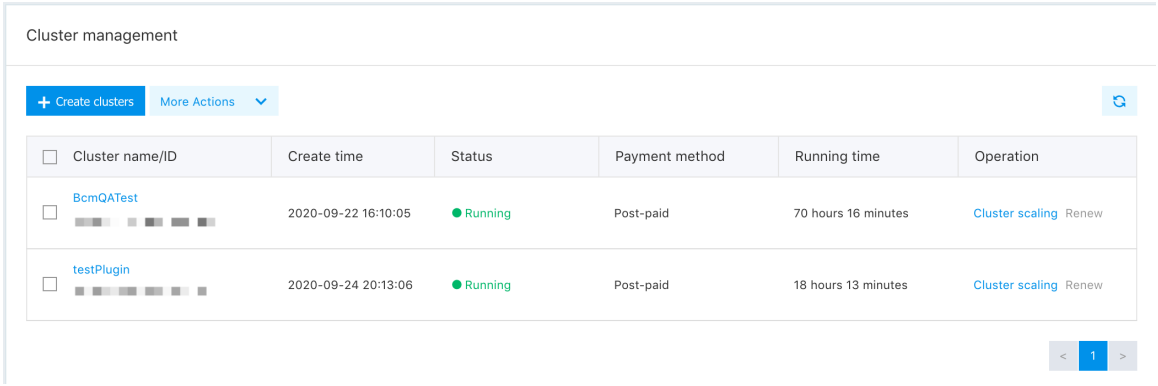


Click “Pay” to complete the construction of the Elasticsearch cluster:

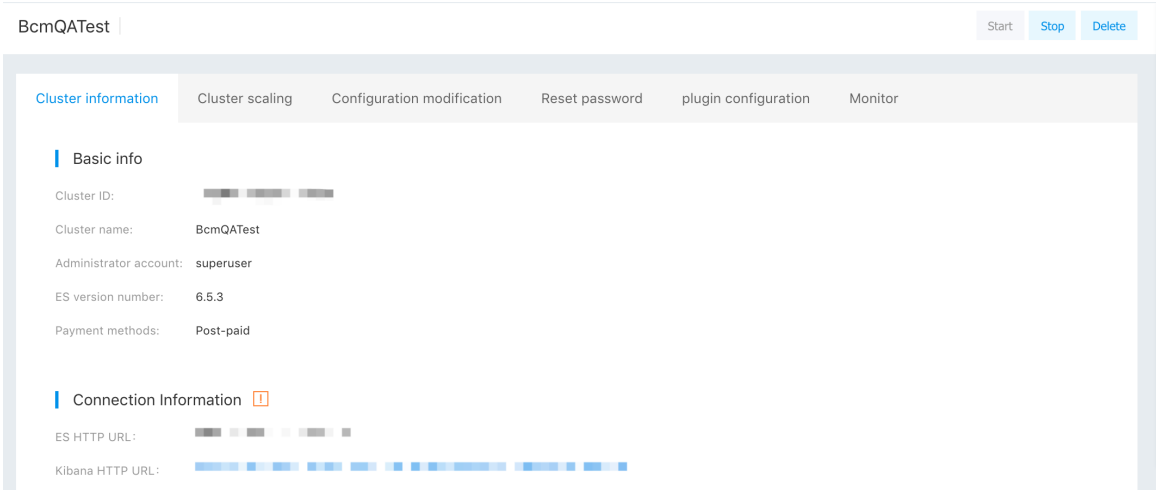


For a convenient use, we hide the complicated parameter configuration of Elasticsearch so that users can easily build and use Elasticsearch. The cluster name we create here is “quickstart” and is configured with 1 Elasticsearch Node.

About 1 minute later, the cluster creation is completed. Click “Management Console” to enter the cluster management interface:



Click the cluster name “quickstart” to view the details of cluster:



Here, we can perform various management operations on the cluster, including stopping and deleting the cluster, scaling the cluster, modifying the password, etc.; you can also view the running status of each node and information such as URL of HTTP connection etc. It is easy to find that the cluster creation and management of Elasticsearch is very simple, basically, you only need to select basic information such as the number of nodes, packages, etc.

The “http” access address of “es” is in the red mark, and the access mode based on “tcp transport” is not provided on cloud, and only the “restful api” access using “http” is provided.

Access the Elasticsearch Service

After building the Elasticsearch cluster, connect the cluster through any HTTP client to import or query data. Note that the Elasticsearch is within the user-defined vpc during cluster creation, and only the bccs in the same vpc can access ES services. By default, there's only one superuser, and its password is the administrator's password specified when the user creates the cluster. By taking the curl command under Linux as an example, introduce how to access ES services.

For example, access the cluster by Http Basic authentication:

```
[root@instance-wkvu7kzn-08 ~]# curl -XGET 172.16.0.19:8200 --user superuser:admin123
{
  "name" : "Ojlelj0",
  "cluster_name" : "quickstart",
  "cluster_uuid" : "hmUugbRlRueD_8RgoV-0LQ",
  "version" : {
    "number" : "5.5.0",
    "build_hash" : "82e5562cc17da61d96ad2501746e0e0b4fbf567f",
    "build_date" : "05/23/2018 15:28:40:040 CST",
    "build_snapshot" : false,
    "lucene_version" : "6.6.0"
  },
  "tagline" : "You Know, for Search"
}
```

By parameter authentication:

```
[root@instance-wkvu7kzn-08 ~]# curl -XGET '172.16.0.19:8200?username=superuser&password=admin123'
{
  "name" : "Ojlelj0",
  "cluster_name" : "quickstart",
  "cluster_uuid" : "hmUugbRlRueD_8RgoV-0LQ",
  "version" : {
    "number" : "5.5.0",
    "build_hash" : "82e5562cc17da61d96ad2501746e0e0b4fbf567f",
    "build_date" : "05/23/2018 15:28:40:040 CST",
    "build_snapshot" : false,
    "lucene_version" : "6.6.0"
  },
  "tagline" : "You Know, for Search"
}
[root@instance-wkvu7kzn-08 ~]#
```

Create an index

Create an index through the index API (with user name and password) created through Elasticsearch:

```
curl -u username:password -XPUT host:port/index_name
```

```
[root@instance-wkvu7kzn-08 ~]# curl -XPUT "http://172.16.0.19:8200/testindex?username=superuser&password=admin123" -d'
> {
>     "settings": {
>       "number_of_replicas": 0,
>       "number_of_shards": 1
>    }
> }'
```

Import data

Real-time import:

```
curl -u username:password -XPOST 'host:port/index_name/type/id' -d '
{

   "field": "value"
}
```

The id is the unique id of the data. If you do not specify an id, the system generates one randomly.

```
[root@instance-wkvu7kzn-08 ~]# curl -XPOST "http://172.16.0.19:8200/testindex/typename/docid?username=superuser&password=admin123" -d'
> {
>     "student_name":"xiao ming",
>     "student_age": 12
> }'
```

Batch import:

```
curl -u username:password -XPOST 'host:port/index_name/type/_bulk?pretty' -d '
{"index":{"_id":"1"}}
{"name": "John Doe" }
{"index":{"_id":"2"}}
{"name": "Jane Doe" }
'
```

Every two rows form one data and must get separated by a newline character. Import the file using the following command:

```
curl -u username:password -XPOST 'host:port/index_name/type/_bulk' --data-binary @import.json
```

Import.json is the file name, and a single file cannot exceed 100M. A large file can be divided into small files and import in parallel.

Query

List all indexes in the cluster:

```
curl -u username:password -XGET 'host:port/_cat/indices?v'
```

```
[root@instance-wkvu7kzn-08 ~]# curl -XGET "http://172.16.0.19:8200/_cat/indices?v&username=superuser&password=admin123"
health status index      uuid                   pri rep docs.count docs.deleted store.size pri.store.size
green  open   testindex  eiDsZOI2Sdme6rSGLDsEmQ   1   0          1            0      3.7kb          3.7kb
green  open   simpleauth u6Z5oQXMRYqvrjMY_TD5BA   1   0          1            0      4.6kb          4.6kb
[root@instance-wkvu7kzn-08 ~]#
```

Query data in the index:

```
curl -u username:password -XGET 'host:port/testindex/_search?pretty' -d '
{

   "query": { "match_all": {} }
}'
```

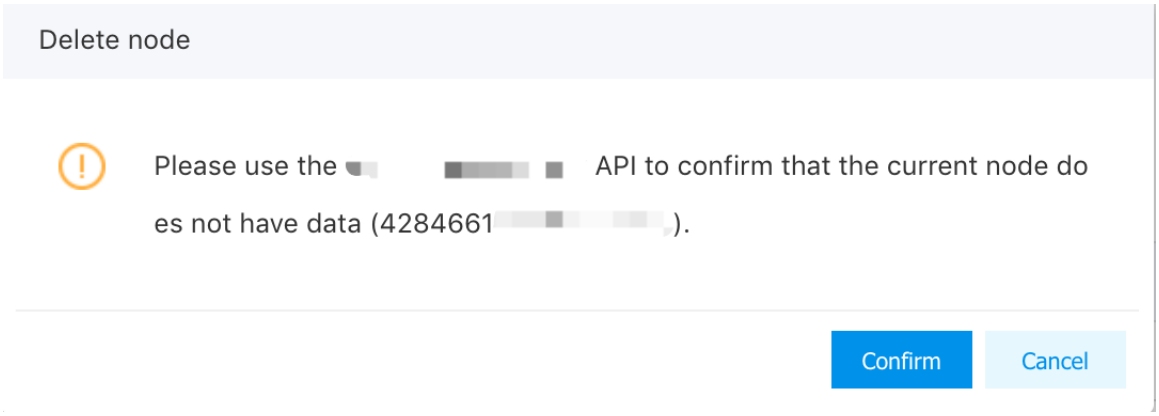
```
[root@instance-wkvu7kzn-08 ~]# curl -XPOST "http://172.16.0.19:8200/testindex/_search?pretty&username=superuser&password=admin123" -d'
> {
>     "query": { "match_all": {} }
> }'
{
  "took" : 0,
  "timed_out" : false,
  "_shards" : {
    "total" : 1,
    "successful" : 1,
    "failed" : 0
  },
  "hits" : {
    "total" : 1,
    "max_score" : 1.0,
    "hits" : [
      {
        "_index" : "testindex",
        "_type" : "typename",
        "_id" : "docid",
        "_score" : 1.0,
        "_source" : {
          "student_name" : "xiao ming",
          "student_age" : 12
        }
      }
    ]
  }
}
```

For other detailed API introduction of the Elasticsearch, see the official documentations of the Elasticsearch:

https://www.elastic.co/guide/en/elasticsearch/reference/current/index.html

Delete a cluster

If you do not want to use the cluster after a simple test, you need to delete it. For this purpose, click the "Delete" button at the upper right corner on the "Cluster Information" page, and then click "Confirm":



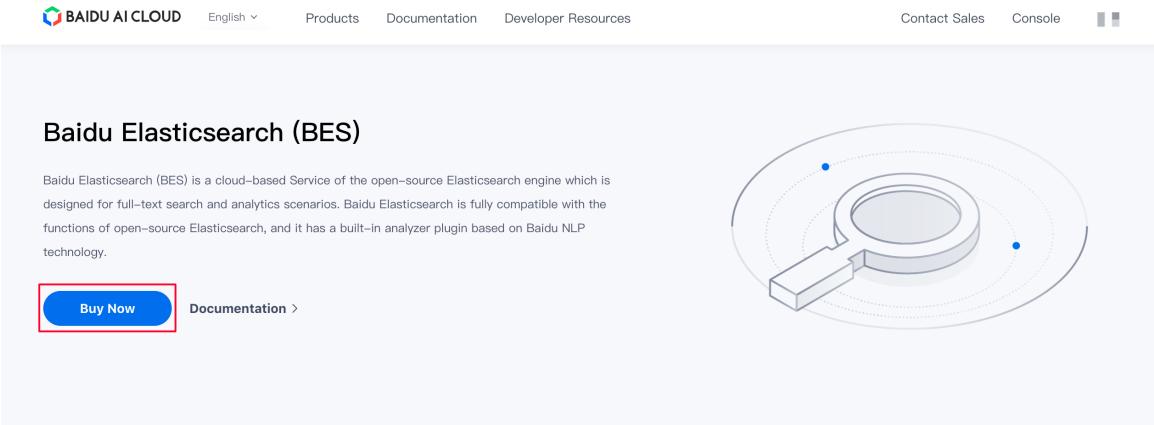
After deleting the cluster, all the data get cleared and cannot be recovered. Meanwhile, you do not need to pay for it any longer.

User Manual

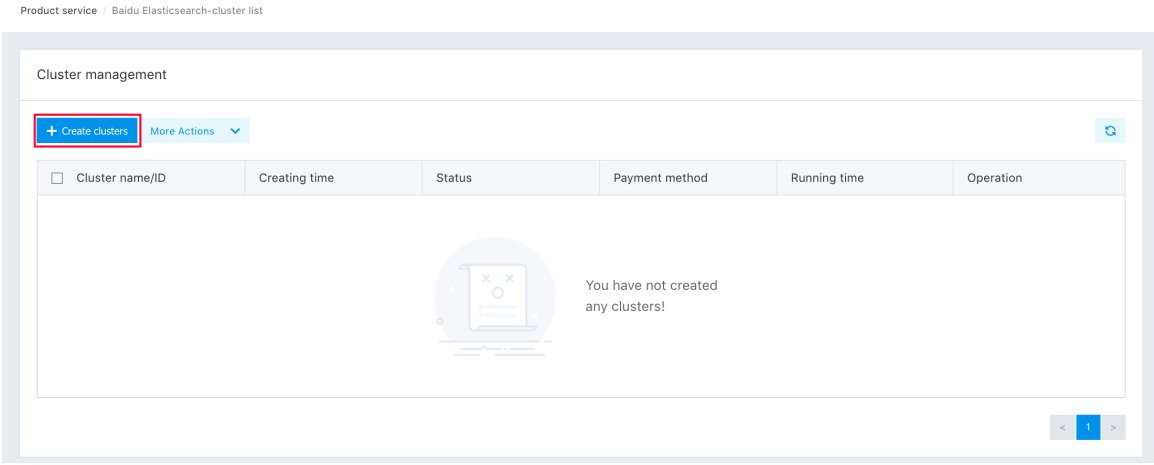
Create a Cluster

1. Log in to Baidu AI Cloud and select Baidu Elasticsearch service. If you do not have a Baidu AI Cloud account, please register one via http://bce.baidu.com/index.html and then select Baidu Elasticsearch service.

2. Click “Buy now”.



3. Enter Baidu Elasticsearch console and click “Create Clusters”.



4. Configuration

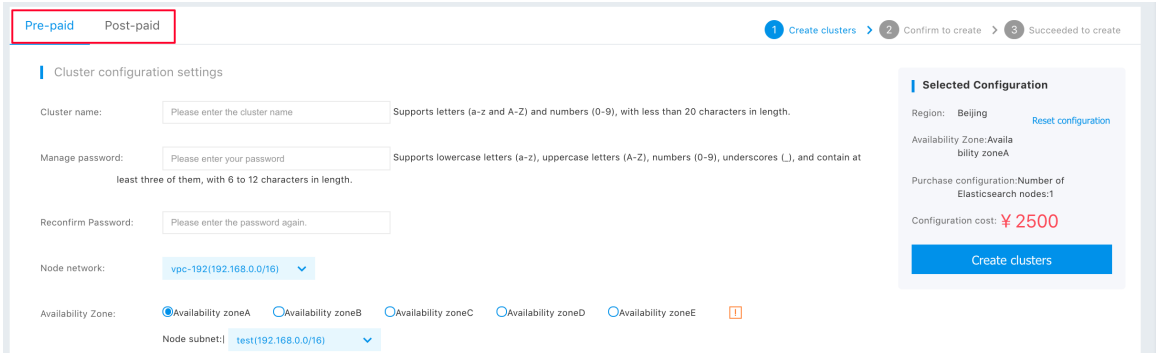
The following gives descriptions for some options:

- Payment modes.
  Prepaid mode. You cannot delete the cluster before it expires. But, it is available for you at a preferential price. You can select this mode for long-term use. Postpaid mode. Unlike the prepaid mode, you can delete the cluster at any time. Recommend you to select the post-payment for your first use. During formal use, you can change the postpaid mode to the prepaid mode.

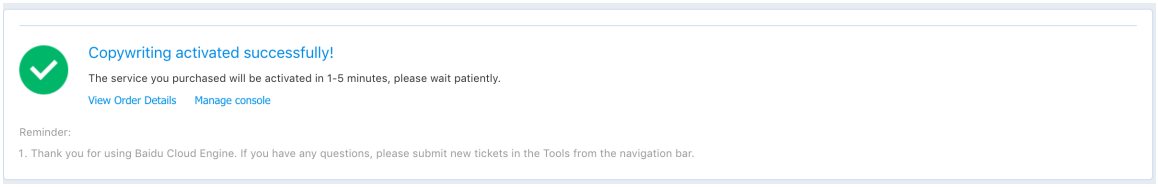
- Enable the “Kibana” service.
  “Kibana” is a platform for “ES” data analysis and visualization, and is enabled by default. Generally, you can access it through a browser only after binding a public IP (For more information, see “Public Network Binding”).

- Number of “Elasticsearch Nodes”
  It is better to create a cluster with more than 3 nodes. The recommended computing method is as follows: Considering the 1.5 times index expansion of the ES, and the maximum disk usage rate of 75%, the number of nodes = (data x number of replicas x 1.5)/(disk capacity of each node x 75%).

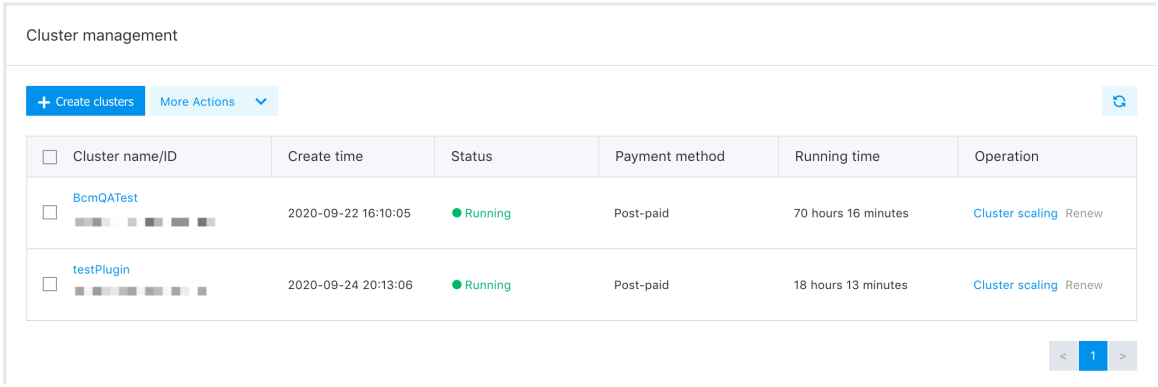
- Security group
  Note that, if it is a custom security group, you should guarantee the communication of 9200 ports between nodes.



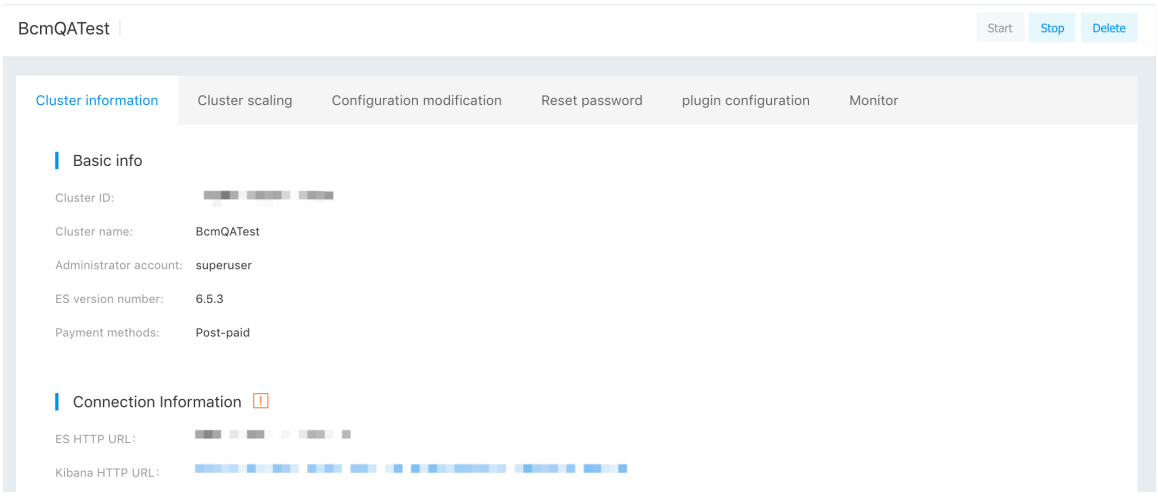
5. Click “Create Clusters” and enter the order confirmation page. After confirmation, building the Elasticsearch cluster is complete.



6. Click “Management Console” to return the cluster management interface.



7. After a while, the creation of the cluster gets done. Then, click the cluster name to enter the page of details.



The following describes some information.

- ES HTTP URL

  You can use it for access to the Elasticsearch. The user name is “superuser”, and the password is the administrator’s password set during cluster creation. Access to this URL is possible only in the same VPC network.

- Kibana HTTP URL

  You can use it for access to the Kibana. The authenticated user is the same as the one accessing the Elasticsearch.

Code of Es Service Regions

Region	Code
Beijing	bj
Baoding	bd
Suzhou	su
Guangzhou	gz
Hong Kong	hkg
Financial Could Wuhan Special Zone	fwh

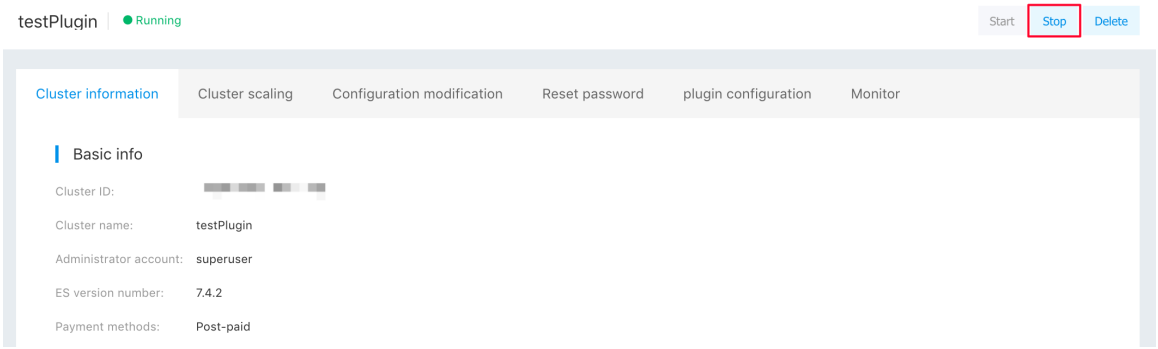
Cluster Scaling

After entering the cluster information page, click the “Cluster Scaling” tab, modify the number of cluster nodes. At this moment, the adjusted cost appears on the right. Then, click “Cluster Scaling” for confirmation.

Notice: The capacity expansion may take some time. In addition, the capacity shrink feature is under development and is unavailable temporarily.

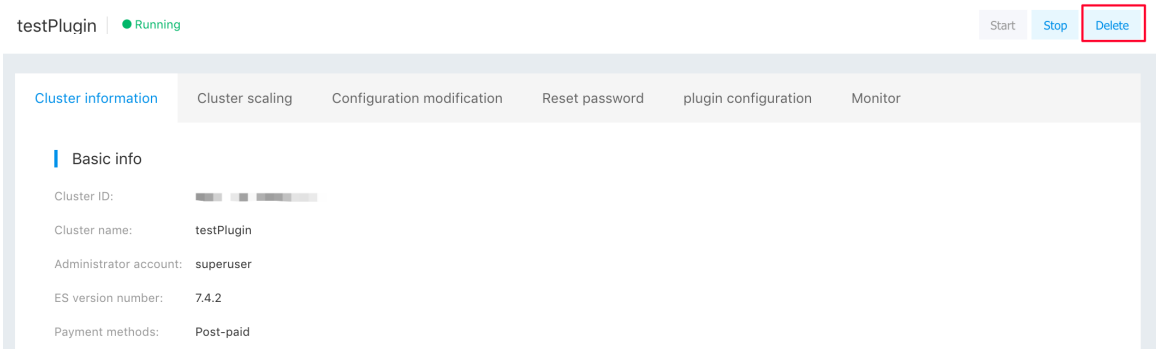
Stop a Cluster

After entering the cluster information page, click the “Stop” button at the upper right corner and then “OK”. Notice: When the cluster is in a stopped state, the resources are not released. So, it is charged in this case. When the cluster stops, no external services are available. But, you can restart it at any time.



Delete a Cluster

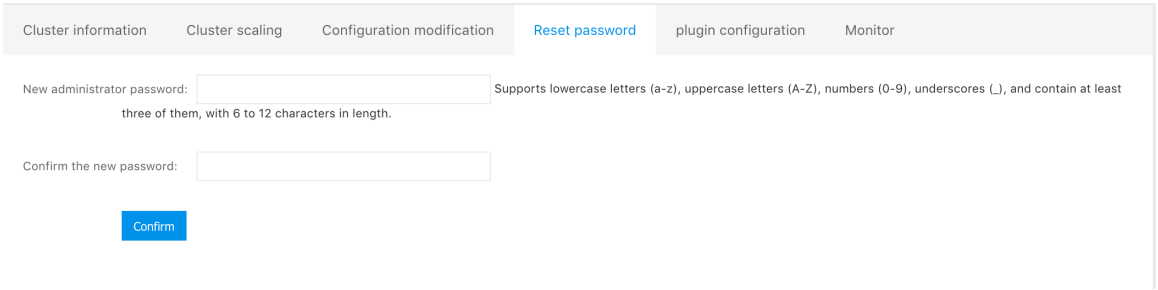
After entering the cluster information page, click the “Delete” button at the upper right corner, and then “OK”. Notice: After deleting the cluster, all the data get cleared and cannot be recovered. Meanwhile, do not collect a charge for it.



Reset the Password

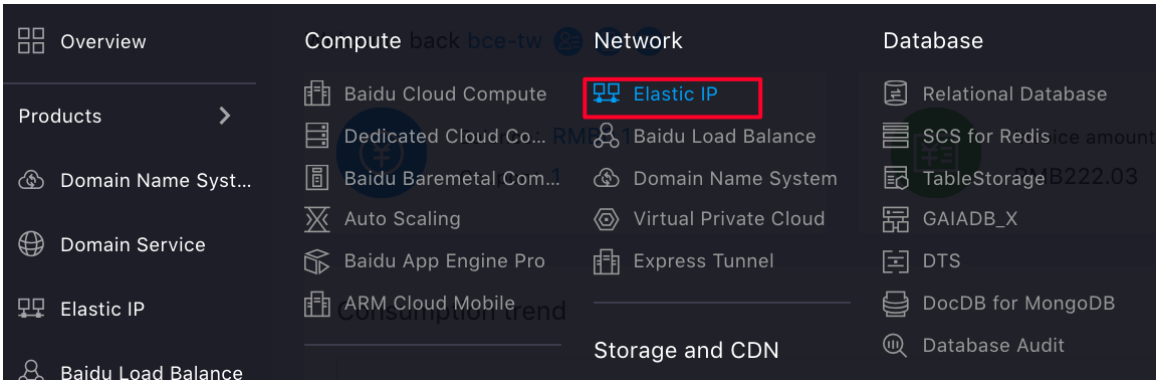
If desired to reset the password of the “superuser” (admin), enter the cluster information page and click the “Reset Password” tab. Then, type the new admin password and enter again for confirmation. After that, enter the short message verification code to confirm. Notice: You can also reset the password through the privilege management API of Baidu Elasticsearch, and

can view the “Privilege Management” module in the user manual.



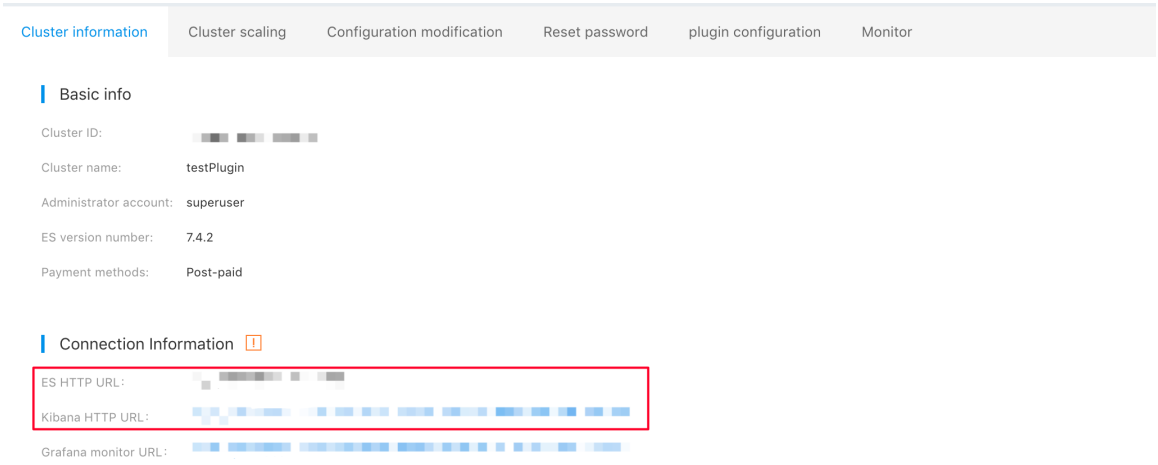
Bind the Public Network

To enable the "Elasticsearch" or "Kibana" to access external networks, enter the EIP page to create a public IP.



After creating the EIP is successful, return to the BES's cluster information page to bind this EIP. Then, you can select to bind it to the "Elasticsearch" or "Kibana address".

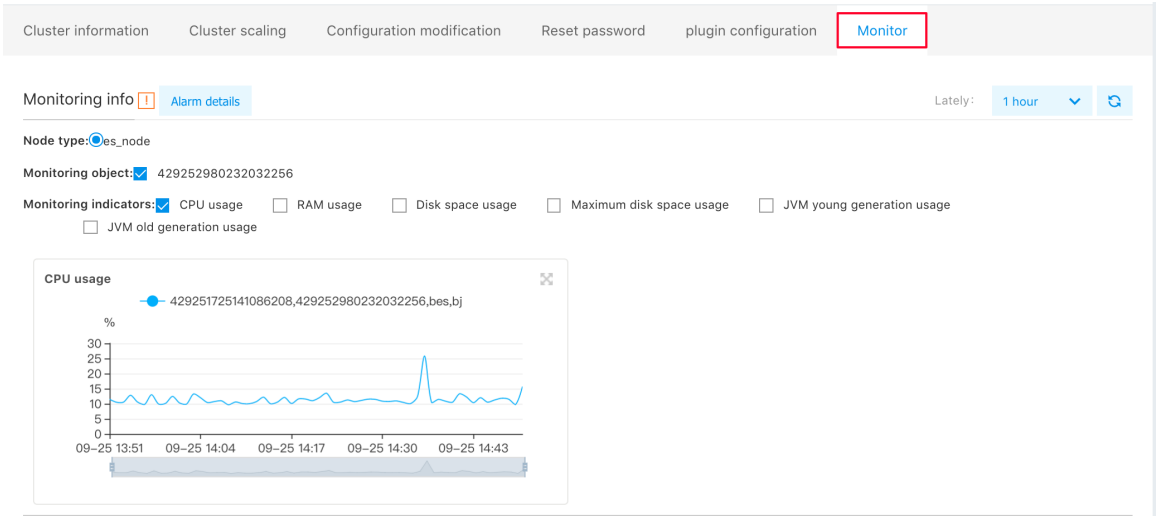
After a period, refresh the page, and check the results.



Notice: Enabling public network access reduces security significantly. In the case of leakage of the user name and password of the "Elasticsearch", anyone can use your "Elasticsearch" service through the public network address.

Monitoring and Alarm

1. By entering the cluster information page, click the “Monitor” tab. Then, you can see the corresponding BCM monitoring chart of each node.



2. If you need to configure the alarm strategy, click “Alarm Details” in the upper right corner to enter the alarm strategy page of “BCM”.

3. Select the monitoring items for alarming, and configure information such as thresholds, and then click “Submit”.

Access the Cluster Resources

Evaluate the disk space storage capacity

For the disk space storage capacity, mainly consider the following aspects:

- Index expansion: When writing data in “ES”, the “ES” stores the original data by default and establishes indexes on each field. Thus, the data volume in the “ES” expands, compared to the original data. However, the data stored in the “ES” is compressed, which can be offset by the index expansion from our observation. Generally, the index expansion does not affect capacity. This expansion rate varies with the configuration parameters of the “ES”. The most relevant configuration parameters are “_all” and “_source”. At present, the “_all” is disabled by default, and the “_source” is enabled. If required to query the original document content, the user can also disable the “_source” to reduce the disk used size.

- “ES” internal consumption: You need to perform regular merge between segments in the “ES”, which reduces the number of segments and releases the space occupied by the deleted documents. The original data remains reserved during the merge, and the “ES” writes a piece of new data. In this case, the data volume doubles, but the “ES” does not fuse all segments, but only selects part of segments for the merge.

- Emergency consumption: Sometimes, the “ES” redistributes data to other nodes after a node is down. In this case, each node needs a particular space to receive data. The space reserved for the merge is also available for emergency consumption, so you can calculate it by combining points 2 and 3 above. The actual disk usage rate is 70%, which is relatively high. If above 70%, an alarm take place.

- Number of replicas.

Considering the four points above, obtain the disk space = original data volume*1.0* number of replicas/0.7

Except for testing, suggest that the number of replicas is at least 2. In this case, disk space = original data volume * 2.8. In most cases, the user can estimate the number of nodes by using the disk space as an index to combine the selected package.

Shard-related assessment

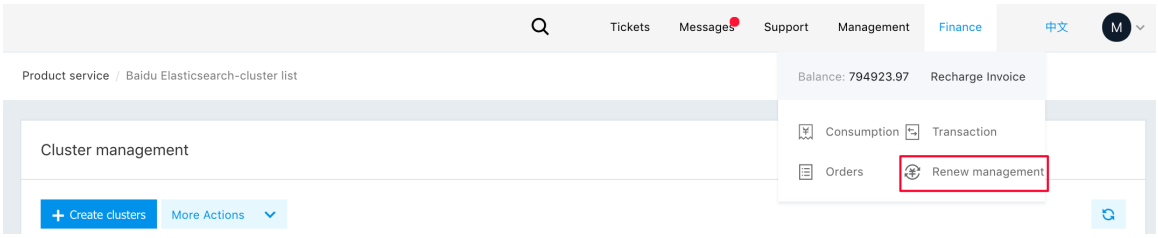
- Each “shard” maintains a separate “lucene” index. The index between “shards” does not merge. Lots of information on the “lucene” index gets stored in the “jvm” memory, such as “FST”, “DocValue” index metadata information, and “FieldInfos” information. This information occupies a huge “JVM” memory. Considering the pointer compression feature of the “JVM”, the “ES” suggests that the “JVM” memory should not exceed 30 GB. Subject to this restriction, suggest that the number of

shards on a node should be below 500 and no more than 1,000.

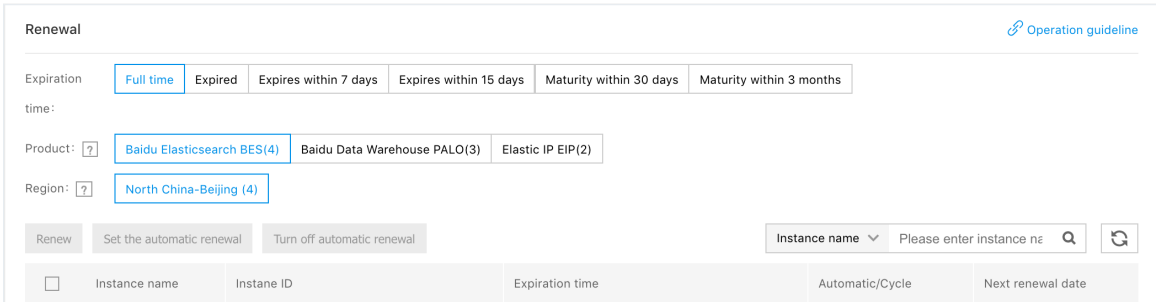
- Big “shards” cause very slow migration of the “shard” between nodes in abnormal conditions. In case of a shutdown, the recovery lasts for a long time. When a “shard” is big, the query delay is much longer. The size of a “shard” should be controlled between 10 GB and 30 GB to achieve a sound query performance. Do not exceed 100 GB.

Automatic Renewal

Two payment methods are available for the prepaid clusters, i.e., manual renewal (operate it on the cluster list page) and automatic renewal. To perform the automatic renewal, enter this page first.



After selecting the Elasticsearch, you can see your prepaid clusters.



Select a cluster and click “Renew”.

Select the renewal duration and click “Next” to confirm. Then, you can receive a short message for the renewal.

After setting, the cluster renewal gets validated automatically within 7 days before the expiration of the last order. The user needs to ensure a sufficient amount in the account.

Privilege Management

The main features of access control are as follows:

- Limit the user's access to the data according to the URL. For example, the administrator wants to give a user the privileges of managing the cluster configuration parameters. The administrator can authorize the "POST" privilege of the URL "/_cluster/setting" to the user.
- Limit the user's data read/write operation according to the Index.

Create a User

```
 curl -u username:password --header "Content-Type: application/json" -XPOST 'host:port/_user/create' -d'
{
    "username" : "test",
    "password" : "test123",
    "get_path" : ["/test-", "/online-", "/_cat/health"],
    "put_path" : ["/test-"],
    "post_path" : ["/test-"],
    "del_path" : ["/test-"],
    "read_index": ["test-", "online-"],
    "write_index": ["test-"]
}'
```

The command mentioned above is to create a "test" user with a login password of "test123". The user has "put", "post", "get" and "delete" privileges for the indexes starting with "test-" and "get" privilege for the indexes starting with "online-", corresponding to the four operations of Restful API.

1. The access path with prefix matching is in "path", corresponding to 4 request methods, i.e., "PUT", "POST", "GET", and "DELETE" in the "HTTP".
2. "Read_index" and "write_index" indicates a list of "indexes" for the user to perform read/write operations. According to prefix matching, the privilege plug-in automatically determines whether to perform read or write operation during the access according to the API accessed by the user.
3. Only the "superuser" can execute the "Create User" command, and other users cannot do so in any case.

Modify a User

```
 curl -u username:password --header "Content-Type: application/json" -XPOST 'host:port/_user/alter' -d'
{
    "username" : "test",
    "password" : "test321",
    "get_path" : ["/test-", "/_cat/health"],
    "read_index": ["test-"]
}'
```

1. This API can modify the user's access privilege as well as the user's password.
2. The fields appeared in the message body overwrite the contents of the original fields. In the hidden field, reserve the original contents. For example, if "write_index" does not appear, the user's "write" privilege for the index can not change, but the privilege of the "read index" is only for "test-xxx".
3. Only "root" and "superuser" can execute this API.

Delete a User

```
 curl -u username:password --header "Content-Type: application/json" -XPOST 'host:port/_user/delete' -d'
{
    "username" : "test"
}'
```

Display the User Information

```
 curl -u username:password --header "Content-Type: application/json" -XPOST 'host:port/_user/show' -d'
{
  "size" : 20
}'
```

1. The API needs to add its privilege path when creating ordinary users so that they can view their user information.
2. If "root" or "superuser" is executing the API, display all users.
3. "Size" means the number of users returned. If not written, it is "10" by default.

Reset the Password

```
 curl -u username:password --header "Content-Type: application/json" -XPOST 'host:port/_user/resetpasswd' -d'
{
    "password" : "test123"
}'
```

1. The API needs to add its privilege path when creating ordinary users so that they can modify their passwords.
2. The administrator should use the API of the updated user when modifying the password of an ordinary user.

Identity and Access Management

Introduction

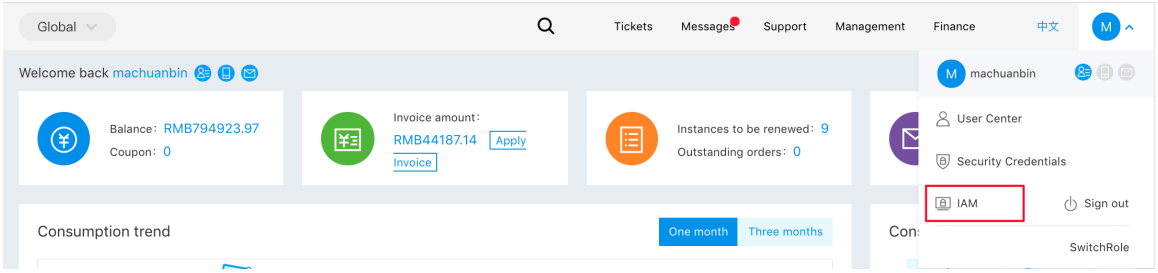
Identity and Access Management (IAM) is mainly used to help users manage access rights to resources under cloud accounts. It is applicable to different roles in the enterprise and can give different rights to different employees to use products. It is recommended that you use Identity and Access Management (IAM) when your enterprise has multi-user collaborative operation resources.

Applicable to the following scenes:

- Customers of medium and large enterprises: Perform the authorized management of multiple employees in the company;
- Technology-based vendor or SAAS platform providers: Manage the resources and rights of the proxy clients;
- Small and medium developers or small enterprises: Add the project members or collaborators to manage the resources.

Create Users

1. After the main account user logs in, selects "Identity and Access Management (IAM)" in the console to enter the user management page.



2. Click "User Administration" in the left navigation bar, and click "New User" on the "Sub-User Management List" page.

3. In the popped up "New User" dialog box, fill in and confirm "User Name", and return to "Sub-user Administration List" area to view the sub-user created.

Configuration Policies

BES supports both system policies and users custom policies to realize product-level and instance-level privilege control of BEC respectively.

- System policy: A privilege set predefined by Baidu AI Cloud system to manage resources. Such policies can authorize sub-users directly and users can only use rather than modify.
- Custom policies: Created by users themselves, a more detailed privilege set for managing resources, which can configure privilege for a single instance, and can meet differentiated permission management of accounts to different users more flexibly.

System Policies

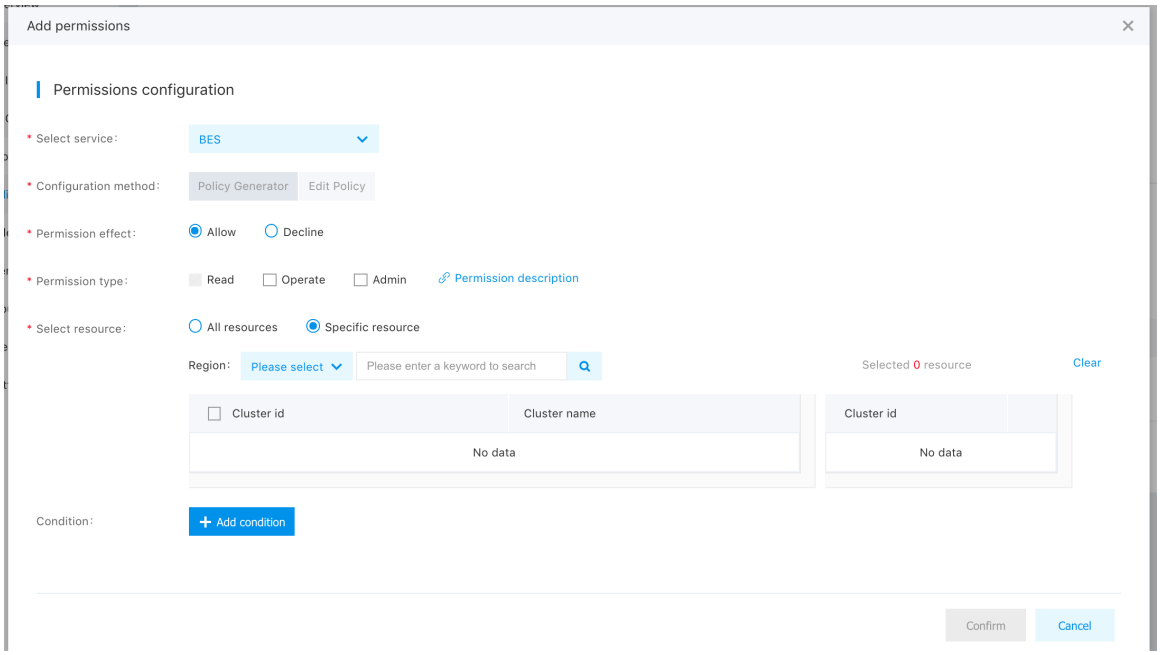
The system policies includes read-only, OPS and managing privileges. The scope of privileges is as follows:

Policy name	Permission instruction	Privilege scope
BESReadAccessPolicy	BES read-only privilege	The read-only privilege includes the right to view the cluster details
BESOperateAccessPolicy	The privileges of OPS and operation of BES	The OPS privileges include the “read-only” privilege, and the functions of starting, stopping, modifying password, binding and unbinding EIP and other privileges
BESFullControlPolicy	The privileges of full control and management of BES	The managing privileges include the “OPS” privilege, and the functions of creating, deleting and cluster scaling

Custom Policies

The custom policies are authorized from a single cluster dimension. Unlike system policies, they are only valid for the selected cluster.

The sub-user first enters【Policy Management】through the left navigation bar, and then clicks “Create Policy”. The user fills in the policy name and selects the service type as “Baidu Elasticsearch BES”. By default, the policy generation method is the policy generator and no modification is required.



User Authorization

Select "Add Privilege" in "Operation" column of the corresponding sub-user in "User Administration -> Sub-user Management List Page", and select and authorize the system privileges or custom policies for users.

> **Note:** If you modify the privileges of a sub-user without modifying the existing policy rules, you can only delete the existing policies and add new policies, but you cannot unselect the added policy privileges.

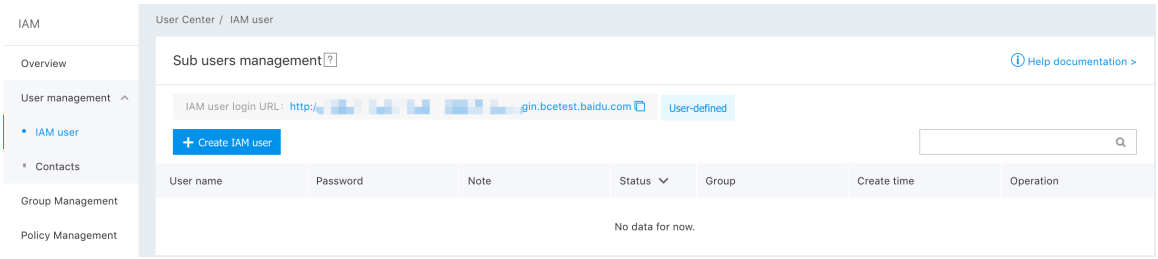
Relevant Service Authorization

In order to enable sub-users to use BES services normally, they also need to have the privileges for other relevant services in Baidu AI Cloud. The privilege policies and the influence scope of relevant services are as follows:

Policy names of relevant services	Description	Influence scope
FCOrderAccessPolicy	The privileges of full control of order services	BES managing, and OPS privileges
VpcFullControlPolicy	The privileges of full control of VPC services	BES managing privileges
SubnetFullControlPolicy	The privileges of full control of subnet services	BES managing privileges
EIPFullControlPolicy	Privileges of full control and management of EIP	BES managing privileges
SecurityGroupFullControlPolicy	The privileges of full control of security group	BES managing privileges

Sub-user Login

After the main account has authorized the sub-user, it can send the link to the sub-user; sub-user can log in to the management console of the main account through IAM user login link, and operate and view the main account resources based on authorized policies.



For other detailed operations, please see: Identity and Access Management (IAM).

Backup Recovery

This document introduces how to use Baidu Elasticsearch (BES) for snapshot creation and restoration, and how to snapshot your data into Baidu Object Storage (BOS). The snapshot and backup are substantially identical in their concept. However, the snapshot gives more focus on the point-in-time, i.e., a certain moment.

Create a snapshot

Creating a snapshot mainly includes two steps below:

- Create a BOS-based warehouse.
- Create a data snapshot.

Create a BOS-based warehouse

- Before creating the warehouse, you need to create the corresponding "bucket" in your BOS and ensure that the user has

the corresponding privileges. Here, you can identify the user by the "access_key" and "secrect_key" of Baidu AI Cloud. You can select the storage type of the "bucket" as required. A recommended choice for you is the standard storage.

- You should ensure that the bucket corresponding to the BOS is in the same region as your Elasticsearch cluster
- The "Es_repo" sets up the warehouse name for you. You can select another name according to your business requirements.

```
 PUT /_snapshot/es_repo
{
    "type": "bos",
    "settings": {
        "access_key": "your access_key",
        "secret_key": "your secret_key",
        "endpoint": "s3.bj.bcebos.com",
        "bucket": "es-repo",
        "base_path": ""
    }
}
```

Meaning of relevant parameters:

Parameters	Features
types	Means the type of warehouse. Enter `bos` here
access_key	Means the "access_key" of Baidu AI Cloud. You can see it in the Baidu AI Cloud "console"
secret_key	Means the "secret_key" of Baidu AI Cloud. You can see it in the "console" of Baidu AI Cloud
endpoint	Means the service domain of each "region" to which the BOS corresponds
bucket	BOS bucket must ensure that the corresponding user identity has "bucket" read and write privileges
base_path	Means the start position of the warehouse, which is the root directory by default
chunk_size	Means that a large file is chunked into multiple parts. The file size is 1 GB by default, the minimum size is 5 MB, and the maximum 5 TB
max_snapshot_bytes_per_sec	Means the maximum snapshotting speed per node, which is `40mb/s` by default
max_restore_bytes_per_sec	Means the maximum restoration speed per node,which is `40mb/s` by default

BOS corresponds to the service domain name of each region.

Region	Access Endpoint
BJ	s3.bj.bcebos.com
GZ	s3.gz.bcebos.com
SU	s3.su.bcebos.com

If you need to modify the corresponding parameters after creating the warehouse, use the "POST" method. If the data to be uploaded is very big, limit the size of the chunks in the "snapshot" process. If it exceeds this size, the data gets uploaded to BOS in chunks.

```
POST /_snapshot/es_repo
{
    "type": "bos",
    "settings": {
        "access_key": "your access_key",
        "secret_key": "your secret_key",
        "endpoint": "s3.bj.bcebos.com",
        "bucket": "es-repo",
        "chunk_size": "1g",
        "base_path": ""
    }
}
```

List All Warehouse Information

```
GET /_snapshot
```

View the specific warehouse information

```
GET /_snapshot/{warehouse name you set}
```

Snapshot

One warehouse can contain multiple snapshots. Each snapshot is a set of a series of indexes, or maybe a single index, a part of indexes, and all indexes. You can specify the index of the snapshot required when creating a snapshot. If you snapshot all open indexes in the cluster without specifying it, give the snapshot a unique name. The name should have a certain meaning. For example, snapshot_2018_07_01 means a snapshot created on July 1, 2018, so that you can recover it according to your data requirements during restoration.

Initiate a snapshot:

```
PUT /_snapshot/es_repo/snapshot_2018_07_01?wait_for_completion=true
```

This request snapshots all open index snapshots in the cluster to the `es_repo` warehouse and name this snapshot to `snapshot_2018_07_01`. This request gets returned immediately after the snapshot initialization, and the snapshot process runs at the backend of your cluster.

The `wait_for_completion` parameter is used to tell whether the request gets returned after the snapshot initialization or after the snapshot is complete. It is `false` by default and returned after snapshot initialization.

When the "snapshot" gets initialized, information about all previous "snapshots" is loaded into the memory. It indicates that even if "wait_for_completion" is set to "false", it may take a few seconds or even a few minutes when there is a big warehouse.

By default, all `open` and `started` indexes in the cluster have "snapshot" created. In the snapshot request, you can specify those indexes to be snapshotted:

```
PUT /_snapshot/es_repo/snapshot_2018_07_01
{
  "indices": "index1,index2",
  "ignore_unavailable": true,
  "include_global_state": false
}
```

Parameters	Features
indices	The "index" list to be included in "snapshot", `multi index syntax` supported.
ignore_unavailable	When it is set to "true", ignore the "index" that does not exist in "indices". It is not set by default. If the "index" does not exist, an error occurs.
include_global_state	When setting it "false", avoid snapshotting the `cluster global state`.

`cluster global state` means the cluster global metadata information maintained by BES. For more information, please see: https://www.elastic.co/guide/en/elasticsearch/reference/current/cluster-state.html

"Snapshot" has the following characteristics:

- "Snapshot" is incremental, and one "snapshot" indicates the `point-in-time` view of the index ("Records" added after "snapshot" creation is not visible in this "snapshot").
- Except for creating the snapshot of all primary shards of this index, you can snapshot the `global cluster metadata` (including persistent cluster setting and templates.)
- For a cluster, only one "snapshot" can run at any time. Executing all snapshot requests is impossible until the last snapshot is complete. Otherwise, rejecting all the snapshots may result.

Query a snapshot

With a "snapshot" created, you can obtain the information about the "snapshot" by initiating a `GET` request for the warehouse and the snapshot name.

Basic format: GET /_snapshot/{your_repo_name}/{your_snapshot_name}, as follows

```
GET /_snapshot/es_repo/snapshot_2018_07_01
```

The response returned contains all information related to the snapshot:

```
{
 "snapshots": [
   {
     "snapshot": "snapshot_2018_07_19",
     "uuid": "TWKo55e7TSy1Sq4WLxMVrQ",
     "version_id": 5050099,
     "version": "5.5.0",
     "indices": [
       "snapindex"
     ],
     "state": "SUCCESS",
     "start_time": "2018-07-19T10:53:17.543Z",
     "start_time_in_millis": 1531997597543,
     "end_time": "2018-07-19T10:53:21.795Z",
     "end_time_in_millis": 1531997601795,
     "duration_in_millis": 4252,
     "failures": [],
     "shards": {
       "total": 1,
       "failed": 0,
       "successful": 1
     }
   }
 ]
}
```

You can obtain the information of multiple snapshots, and the `GET` request supports the wildcard matching of multiple snapshot information. In this case, the meaning of the snapshot name has the following effect:

```
GET /_snapshot/es_repo/snapshot_order_*
```

You can use the parameter `_all` to obtain complete list information of all snapshots in a warehouse:

```
GET /_snapshot/es_repo/_all
```

Stop or delete a snapshot

The BES service provided by Baidu AI Cloud does not have a separate API for stopping the snapshot. Stopping or deleting a snapshot is one semantics. If found that the "snapshot" execution is wrong or takes a very long time, you can stop the "snapshot" running at the backend by deleting it:

```
DELETE /_snapshot/es_repo/snapshot_2018_07_01
```

Delete the snapshot from the warehouse:

```
DELETE /_snapshot/es_repo/snapshot_2018_07_01
```

You can also delete a warehouse directly:

```
DELETE /_snapshot/es_repo
```

Notice:`When deleting a snapshot or warehouse, the ES simply removes the cluster's location reference to the warehouse or snapshot. The user needs to process real physical files and others by themselves. When confirming that all snapshots are no longer applicable, you can delete the warehouse metadata in the Elasticsearch, and then log in to Baidu AI Cloud BOS console to manually delete the warehouse'. Additionally,' Don't manually delete any snapshot file in the BOS warehouses, manually deleting a snapshot file results in the snapshot being unavailable and irrecoverable loss when recovered.`

View snapshot progress

You can view the progress information of a snapshot through the `status` interface.

```
GET /_snapshot/es_repo/snapshot_2018_07_19/_status
```

The following is the detailed statistical information returned by the `status` interface:

```
{
 "snapshots": [
   {
     "snapshot": "snapshot_2018_07_19",
     "repository": "es_repo",
     "uuid": "TWKo55e7TSy1Sq4WLxMVrQ",
     "state": "SUCCESS",  .....  [A]
     "shards_stats": {
       "initializing": 0,
       "started": 0,
       "finalizing": 0,
       "done": 1,
       "failed": 0,
       "total": 1
     },
     "stats": {
       "number_of_files": 16,
       "processed_files": 16,
       "total_size_in_bytes": 18639,
       "processed_size_in_bytes": 18639,
       "start_time_in_millis": 1531997598051,
       "time_in_millis": 2782
     },
     "indices": {
       "snapindex": {
         "shards_stats": {
           "initializing": 0,
           "started": 0,
           "finalizing": 0,
           "done": 1,  ..... [B]
           "failed": 0,
           "total": 1
         },
         "stats": {
           "number_of_files": 16,
           "processed_files": 16,
           "total_size_in_bytes": 18639,
           "processed_size_in_bytes": 18639,
           "start_time_in_millis": 1531997598051,
           "time_in_millis": 2782
         },
         "shards": {
           "0": {
             "stage": "DONE",..... [C]
             "stats": {
               "number_of_files": 16,
               "processed_files": 16,
               "total_size_in_bytes": 18639,
               "processed_size_in_bytes": 18639,
               "start_time_in_millis": 1531997598051,
               "time_in_millis": 2782
             }
           }
         }
       }
     }
   }
 ]
}
```


It contains all information of the snapshot, such as the start time, total size, total number of files, and total number of files

that have been processed. The current state of all "indexes" snapshotted and state of all "shards" under the "index" are also recorded detailedly.

- [A] indicates that this snapshot has been complete, displaying the `SUCCESS` state. And, the running snapshot displays `IN_PROGRESS`.
- [B] indicates that the snapshots of all shards of this index have been complete.
- [C] indicates that the snapshots of shards corresponding to this index have been complete.

Different state values indicate different meanings:

Status Value	Meaning
INIT	The snapshot does not get started but is initializing
STARTED	The snapshot is copying the "index" file
FINALIZE	The metadata of the snapshot is being written into a remote warehouse
DONE	The snapshot is complete successfully
FAILURE	The snapshot gets failed. For the possible causes, you can see them in the `status` API

Restorer a snapshot

You can restore a snapshot through the following command:

```
POST /_snapshot/es_repo/snapshot_2018_07_19/_restore
```

By default, all indexes in the specified snapshot get restored. You can specify the `index` and `global cluster state` by adding `indices` and `include_global_state` to the request body:

```
POST /_snapshot/es_repo/snapshot_2018_07_19/_restore
{
  "indices": "snapindex",
  "ignore_unavailable": true,
  "include_global_state": true,
  "rename_pattern": "snap(.+)",
  "rename_replacement": "restore$1"
}
```

You can use the `rename_pattern` and `rename_replacement` to rename the “index”. Most “indexes” can be reset as follows:

```
POST /_snapshot/repo/snapshot_wyf_2018_01_29/_restore
{
  "indices": "wyf",
  "index_settings": {
    “index.number_of_replicas": 0
  },
  "ignore_index_settings": [
    "index.refresh_interval"
  ]
}
```

You should pay attention to that:`You cannot perform some settings during restoration, such as index.number_of_shards`. However, you can restore it to another cluster. The version of the new cluster must be the same as or larger than the cluster snapshotted (only 1 "major version" larger is allowed). For example, you can restore a snapshot of 1.x to 2.x, but not to 5.x.

Like the snapshot request, the `restore` request gets returned immediately after checking the snapshot information and verifying the index information in the snapshot. The restoration gets done at the backend of the cluster. You can add the parameter `wait_for_completion` at the end of the request to complete the restoration before the request is blocked:

```
POST /_snapshot/es_repo/snapshot_2018_07_19/_restore?wait_for_completion=true
```

Monitor the snapshot restoration

Restoring the data from the BOS warehouse utilizes the internal recovery mechanism of the Elasticsearch. From the principle of internal implementation, restoring the data from the warehouse is entirely equivalent to restoring the data from one node to another. The internal restoration of the Elasticsearch includes existing_store restoration, peer restoration, and snapshot restoration.

You can view the restoration progress through the `recovery` API:

```
GET /{index}/_recovery

GET snapindex/_recovery
```

This interface returns the following responses:

```
{
  "snapindex": {
    "shards": [
      {
        "id": 0,
        "type": "SNAPSHOT", ..... [A]
        "stage": "DONE", ..... [B]
        "primary": true,
        "start_time_in_millis": 1532065843418,
        "stop_time_in_millis": 1532065845773,
        "total_time_in_millis": 2354,
        "source": { ..... [C]
          "repository": "es_repo",
          "snapshot": "snapshot_2018_07_19",
          "version": "5.5.0",
          "index": "snapindex"
        },
        "target": {
          "id": "8wR8Z38USImEeSO0SZ1_hA",
          "host": "192.168.16.5",
          "transport_address": "192.168.16.5:9300",
          "ip": "192.168.16.5",
          "name": "8wR8Z38"
        },
        "index": {
          "size": {
            "total_in_bytes": 18668,
            "reused_in_bytes": 0,
            "recovered_in_bytes": 18668,
            "percent": "100.0%" ..... [D]
          },
          "files": {
            "total": 16,
            "reused": 0,
            "recovered": 16,
            "percent": "100.0%"
          },
          "total_time_in_millis": 2148,
          "source_throttle_time_in_millis": 0,
          "target_throttle_time_in_millis": 0
        },
        "translog": {
          "recovered": 0,
          "total": 0,
          "percent": "100.0%",
          "total_on_start": 0,
          "total_time_in_millis": 158
        },
        "verify_index": {
          "check_index_time_in_millis": 0,
          "total_time_in_millis": 0
        }
      }
    ]
  }
}
```


- [A] `type` indicates restoration from the remote warehouse snapshot.
- [B] `stage` field indicates this restoration is complete.

- [C] `source` field indicates this restoration is complete.
- [D] `percent` field indicates the completion percentage of the restoration.

Cancel the running restoration

You can cancel the "index" restoration by deleting the "index" being restored:

```
DELETE /snapindex
```

Notes for snapshot

- One cluster can only have one snapshot running at a time.
- You cannot create a snapshot at the same time of deleting a snapshot.

Notes for restoration

- The index of a `restore` may be nonexistent. Otherwise, it must be in the `closed` state.
- The index of a `restore` overwrites the previous file. Even if the file is identical, delete the `old` file, and then create a `new` file.
- The `restore` process skips the `translog` recovery process to create a new translog.
- If the restoration target is not the snapshotted ES cluster but a new cluster, you need to create a "repo" in the new cluster and set the "read_only" parameter to "true".

Configure the IK-Analyzer-Thesaurus

Configure the thesaurus

The user can define the thesaurus when the thesaurus of IK does not meet the requirements. The configuration steps are as follows:

- The user puts the edited thesaurus on an http server.
- The user configures the http address of the "ik" thesaurus in "es". For example, Baidu AI Cloud's word configuration file is "baidu.dict". When the configuration file of the stop word is "baidu_stop.dic", send a command to "es" as follows:

```
 PUT /_cluster/settings
{
    "persistent": {
        "bpack.ik_analyzer.remote_ext_dict":"http://ip:port/baidu.dic",
        "bpack.ik_analyzer.remote_ext_stopwords":"http://ip:port/baidu_stop.dic"
    }
}
```

- "Es" checks whether the thesaurus file directed by the "http url" in the setting changes every 60s. If so, "es" automatically downloads the file and load it into the "ik".

Verify whether the thesaurus is valid

After configuration, the user can check through "API POST /_analyze" whether the thesaurus is valid. For example:

- Before configuring the thesaurus, send the command

```
 POST /_analyze
{
  "analyzer" : "ik_smart",
  "text" : ["Zhao Xiaomingming is so handsome"]
}
```

The returned result of the es is as follows:

```
{
  "tokens": [
    {
      "token": "Zhao",
      "start_offset": 0,
      "end_offset": 1,
      "type": "CN_WORD",
      "position": 0
    },
    {
      "token": "Xiaoming",
      "start_offset": 1,
      "end_offset": 3,
      "type": "CN_WORD",
      "position": 1
    },
    {
      "token": "Ming",
      "start_offset": 3,
      "end_offset": 4,
      "type": "CN_WORD",
      "position": 2
    },
    {
      "token": "So handsome"",
      "start_offset": 4,
      "end_offset": 6,
      "type": "CN_WORD",
      "position": 3
    }
  ]
}
```

- Then, configure the thesaurus. The normal thesaurus only contains "Zhao Xiaomingming" and the stop word thesaurus contains "So handsome". After configuration, call "/_analyze api" again. The results are as follows:

```
{
  "tokens": [
    {
      "token": "Zhao Xiaomingming",
      "start_offset": 0,
      "end_offset": 4,
      "type": "CN_WORD",
      "position": 0
    }
  ]
}
```

The result shows that "Zhao Xiaomingming" is a separate word, while "so handsome" is removed as stop words.

Upgrade Between Individual Version of the Es

There are three ways to upgrade between different versions of the "Es":

- Replace the binary package. Download the new version of the "es" binary package, replace the existing package, and start. There is a risk for this method, that if an error occurs during upgrade, it cannot roll back.
- "Backup restore". Back up the "es" cluster data to shared storage such as BOS, HDFS, and NFS and then build a new cluster with the new version binary package. Then backed up "restore" data on the new cluster. In most cases, recommend you to use this method.
- reindex. You can perform the upgrade between any versions. But because of the original cluster data read out one by one and then inserted into the new cluster, the performance is not good enough and takes a long time. The reindex function starts from version 2.x.

The main versions of "Es" used in production include versions 1.7, 2.4, 5.5, and 6.5. The following gives descriptions of the upgrade methods between different versions.

- In the case of version 1.7 -> 2.4, you can replace the "jar" package by "backup restore" in normal conditions. However, it is found that in some cases it cannot be upgraded, including the following situations: 1. The field name contains special characters. 2. The "doc value" setting of the same field in different types is different. In this case, you can upgrade only after deleting these abnormal "indexes".
- In the case of version 1.7 --> 5.5, the upgrade cannot get done directly. Version 1.7 must be upgraded to 2.4 and then from 2.4 reindex to 5.5.
- In the case of version 1.7 --> 6.5, the upgrade cannot get done directly. Version 1.7 must be upgraded to 2.4 and then from 2.4 reindex to 6.5.
- In the case of version 2.4 --> 5.5, the upgrade cannot get done by "backup restore" or replacing the binary system.
- In the case of version 2.4 --> 6.5, the upgrade gets done by the reindex.
- In the case of version 5.5 --> 6.5, the upgrade can be performed by "backup restore" or replacing the binary system.

Sum up one principle from the above descriptions:

- The upgrade with only 1 version number difference between main versions from X to X+1 (e.g., from 1.x to 2.x, or 5.x to 6.x) is possible by "backup restore" or replacing the binary "jar" package.
- The upgrade with version difference above 1 (e.g., from 1.x to 5.x) between main versions cannot be realized by backup restore or replacing the binary system, but by "reindex".
- After upgrading from version X to version X+1 by backup restore or replacing the binary "jar" package, you cannot upgrade to version X+2 in the same way, but by "reindex".

For detailed constraint information of upgrade, please see Es Official Website.

NLP Chinese Word Segmentation Plugin

The `analysis-baidu-nlp` is the Chinese word segmentation plug-in developed independently by the Baidu AI Cloud Elasticsearch (ES) team. The performance and accuracy rate of the plug-in in the Chinese word segmentation are at the advanced level in the industry.

Background

The `analysis-baidu-nlp` is based on the DeepCRF model developed independently by Baidu NLP. The model condenses Baidu's technology accumulation of over ten years in the Chinese search field. The performance and accuracy rate of the model are at

the leading position in the industry.

Provide basic granularity and phrase granularity word segmentation results for different application requirements. The phrase granularity is the result of intelligent combination of basic granularity segmentations.

> Note: The dictionary model is loaded to the out-of-core memory of JVM when it is used at the first time. The package node memory that we recommend to use is 8G above.

Word Segmentation Granularity

analysis-baidu-nlp Mainly provide `Analyzer` of two granularities:

1. Basic granularity model (bd-nlp-basic)
2. Phrase granularity model (bd-nlp-phrase)

Two `Analyzer` internally integrated case sensitivity filter, stopwords filter, out-of-the-box

Provide two kinds of `Tokenizers` with the same name:

1. Basic model granularity (bd-nlp-basic)
2. Phrase large granularity model (bd-nlp-phrase)

The two granularities `Tokenizers` only provide the most original word segmentation results. The user can add the custom stopwords filter and some complex filters according to self-own application requirements.

Comparison with ik in the Basic Granularity and Phrase Granularity Word Segmentation

Comparison in basic granularity

Conduct comparison in the basic maximum granularity word segmentation effect for "Maintenance Fund"

- bd-nlp-basic Word segmentation

```
 POST /_analyze
{
   "text": " Maintenance fund ",
   "analyzer": "bd-nlp-basic"
}
```

Word segmentation result:

```
 {
  "tokens": [
    {
      "token": " Maintenance ",
      "start_offset": 0,
      "end_offset": 2,
      "type": "WORD",
      "position": 0
    },
    {
      "token": " Fund ",
      "start_offset": 2,
      "end_offset": 4,
      "type": "WORD",
      "position": 1
    }
  ]
}
```

- ik_max_word Word segmentation

```
 POST _analyze
{
   "analyzer": "ik_max_word",
   "text": " Maintenance fund "
}
```

Word segmentation result:

```
{
 "tokens": [
   {
     "token": " Maintenance fund ",
     "start_offset": 0,
     "end_offset": 4,
     "type": "CN_WORD",
     "position": 0
   },
   {
     "token": " Maintenance ",
     "start_offset": 0,
     "end_offset": 2,
     "type": "CN_WORD",
     "position": 1
   },
   {
     "token": " Maintenance ",
     "start_offset": 0,
     "end_offset": 1,
     "type": "CN_WORD",
     "position": 2
   },
   {
     "token": " Maintenance ",
     "start_offset": 1,
     "end_offset": 2,
     "type": "CN_CHAR",
     "position": 3
   },
   {
     "token": " Fund ",
     "start_offset": 2,
     "end_offset": 4,
     "type": "CN_WORD",
     "position": 4
   },
   {
     "token": " Fund ",
     "start_offset": 2,
     "end_offset": 3,
     "type": "CN_WORD",
     "position": 5
   },
   {
     "token": " Fund ",
     "start_offset": 3,
     "end_offset": 4,
     "type": "CN_CHAR",
     "position": 6
   }
 ]
}
```


Conduct phrase word segmentation effect comparison for "Qingming Festival, also known as Outing Festival, Xingqing Festival, March Festival, Ancestor Worship Festival, etc.".

- bd-nlp-basic Phrase word segmentation

```
 POST /_analyze
{
    "text": " Qingming Festival is also known as Outing Festival, Xingqing Festival, March Festival, Ancestor Worship Festival, etc. ",
    "analyzer": "bd-nlp-phrase"
}
```

Phrase segmentation result:

```
 {
  "tokens": [
    {
      "token": " Qingming Festival ",
      "start_offset": 0,
      "end_offset": 3,
      "type": "WORD",
      "position": 0
    },
    {
      "token": " Also known as ",
      "start_offset": 4,
      "end_offset": 6,
      "type": "WORD",
      "position": 2
    },
    {
      "token": " Outing Festival ",
      "start_offset": 6,
      "end_offset": 9,
      "type": "WORD",
      "position": 3
    },
    {
      "token": " Xingqing Festival ",
      "start_offset": 10,
      "end_offset": 13,
      "type": "WORD",
      "position": 5
    },
    {
      "token": " March Festival ",
      "start_offset": 14,
      "end_offset": 17,
      "type": "WORD",
      "position": 7
    },
    {
      "token": " Ancestor Worship ",
      "start_offset": 18,
      "end_offset": 20,
      "type": "WORD",
      "position": 9
    },
    {
      "token": " Byte ",
      "start_offset": 20,
      "end_offset": 21,
      "type": "WORD",
      "position": 10
    }
  ]
}
```


- ik_smart Intelligent word segmentation

```
 POST _analyze
{
    "analyzer": "ik_smart",
    "text": " Qingming Festival is also known as Outing Festival, Xingqing Festival, March Festival, Ancestor Worship Festival, etc. "
}
```

Word segmentation result:

```
 {
  "tokens": [
    {
      "token": " Qingming Festival ",
      "start_offset": 0,
      "end_offset": 3,
      "type": "CN_WORD",
      "position": 0
    },
    {
      "token": " Also known as ",
      "start_offset": 4,
      "end_offset": 6,
      "type": "CN_WORD",
      "position": 1
    },
    {
      "token": " Outing ",
      "start_offset": 6,
      "end_offset": 8,
      "type": "CN_WORD",
      "position": 2
    },
    {
      "token": " Byte ",
      "start_offset": 8,
      "end_offset": 9,
      "type": "CN_WORD",
      "position": 3
    },
    {
      "token": " Row ",
      "start_offset": 10,
      "end_offset": 11,
      "type": "CN_WORD",
      "position": 4
    },
    {
      "token": " Qing ",
      "start_offset": 11,
      "end_offset": 12,
      "type": "CN_CHAR",
      "position": 5
    },
    {
      "token": " Byte ",
      "start_offset": 12,
      "end_offset": 13,
      "type": "CN_WORD",
      "position": 6
    },
    {
```

```
    {
      "token": " March ",
      "start_offset": 14,
      "end_offset": 16,
      "type": "CN_WORD",
      "position": 7
    },
    {
      "token": " Byte ",
      "start_offset": 16,
      "end_offset": 17,
      "type": "COUNT",
      "position": 8
    },
    {
      "token": " Ancestor Worship ",
      "start_offset": 18,
      "end_offset": 20,
      "type": "CN_WORD",
      "position": 9
    },
    {
      "token": " Byte ",
      "start_offset": 20,
      "end_offset": 21,
      "type": "CN_WORD",
      "position": 10
    }
  ]
}
```

Analyze API Use

Basic model granularity word segmentation

```
 POST /_analyze
{
  "analyzer": "bd-nlp-basic",
  "text": " Last year, we had a Fireside Competition with them. We won in the first round but was defeated in the second round and the third round. "
}
```

Word segmentation result:

```
 {
  "tokens": [
    {
      "token": " Last year ",
      "start_offset": 0,
      "end_offset": 2,
      "type": "WORD",
      "position": 0
    },
    {
      "token": " We ",
      "start_offset": 2,
      "end_offset": 4,
      "type": "WORD",
      "position": 1
    },
    {
```

```
    "token": " and ",
    "start_offset": 4,
    "end_offset": 5,
    "type": "WORD",
    "position": 2
  },
  {
    "token": " Them ",
    "start_offset": 5,
    "end_offset": 7,
    "type": "WORD",
    "position": 3
  },
  {
    "token": " Had ",
    "start_offset": 7,
    "end_offset": 9,
    "type": "WORD",
    "position": 4
  },
  {
    "token": " Fireside ",
    "start_offset": 10,
    "end_offset": 12,
    "type": "WORD",
    "position": 6
  },
  {
    "token": " Competition ",
    "start_offset": 12,
    "end_offset": 14,
    "type": "WORD",
    "position": 7
  },
  {
    "token": " The first ",
    "start_offset": 15,
    "end_offset": 17,
    "type": "WORD",
    "position": 9
  },
  {
    "token": " Round ",
    "start_offset": 17,
    "end_offset": 19,
    "type": "WORD",
    "position": 10
  },
  {
    "token": " Won ",
    "start_offset": 19,
    "end_offset": 20,
    "type": "WORD",
    "position": 11
  },
  {
    "token": " The second ",
    "start_offset": 22,
    "end_offset": 24,
    "type": "WORD",
    "position": 14
  },
  {
```

```
    {
      "token": " Round ",
      "start_offset": 24,
      "end_offset": 26,
      "type": "WORD",
      "position": 15
    },
    {
      "token": " and ",
      "start_offset": 26,
      "end_offset": 27,
      "type": "WORD",
      "position": 16
    },
    {
      "token": " The third ",
      "start_offset": 27,
      "end_offset": 29,
      "type": "WORD",
      "position": 17
    },
    {
      "token": " Round ",
      "start_offset": 29,
      "end_offset": 31,
      "type": "WORD",
      "position": 18
    },
    {
      "token": " Failure ",
      "start_offset": 32,
      "end_offset": 33,
      "type": "WORD",
      "position": 20
    },
    {
      "token": " Failure ",
      "start_offset": 33,
      "end_offset": 34,
      "type": "WORD",
      "position": 21
    },
    {
      "token": " Failure ",
      "start_offset": 34,
      "end_offset": 35,
      "type": "WORD",
      "position": 22
    },
    {
      "token": " Failure",
      "start_offset": 35,
      "end_offset": 36,
      "type": "WORD",
      "position": 23
    }
  ]
}
```


Phrase model large granularity word segmentation

```
 POST /_analyze
{
  "analyzer": "bd-nlp-phrase",
  "text": " Last year, we had a Fireside Competition with them. We won in the first round but was defeated in the second round and the third round. 。"
}
```

Word segmentation result:


```
 {
  "tokens": [
    {
      "token": " Last year ",
      "start_offset": 0,
      "end_offset": 2,
      "type": "WORD",
      "position": 0
    },
    {
      "token": " We ",
      "start_offset": 2,
      "end_offset": 4,
      "type": "WORD",
      "position": 1
    },
    {
      "token": " and ",
      "start_offset": 4,
      "end_offset": 5,
      "type": "WORD",
      "position": 2
    },
    {
      "token": " Them ",
      "start_offset": 5,
      "end_offset": 7,
      "type": "WORD",
      "position": 3
    },
    {
      "token": " Had ",
      "start_offset": 7,
      "end_offset": 9,
      "type": "WORD",
      "position": 4
    },
    {
      "token": " Fireside Competition ",
      "start_offset": 10,
      "end_offset": 14,
      "type": "WORD",
      "position": 6
    },
    {
      "token": " The first round ",
      "start_offset": 15,
      "end_offset": 19,
      "type": "WORD",
      "position": 8
    },
    {
```

```
    {
      "token": " Won ",
      "start_offset": 19,
      "end_offset": 20,
      "type": "WORD",
      "position": 9
    },
    {
      "token": " The second round ",
      "start_offset": 22,
      "end_offset": 26,
      "type": "WORD",
      "position": 12
    },
    {
      "token": " and ",
      "start_offset": 26,
      "end_offset": 27,
      "type": "WORD",
      "position": 13
    },
    {
      "token": " The third ",
      "start_offset": 27,
      "end_offset": 29,
      "type": "WORD",
      "position": 14
    },
    {
      "token": " Round ",
      "start_offset": 29,
      "end_offset": 31,
      "type": "WORD",
      "position": 15
    },
    {
      "token": " Failure ",
      "start_offset": 32,
      "end_offset": 33,
      "type": "WORD",
      "position": 17
    },
    {
      "token": " Failure ",
      "start_offset": 33,
      "end_offset": 34,
      "type": "WORD",
      "position": 18
    },
    {
      "token": " Failure ",
      "start_offset": 34,
      "end_offset": 35,
      "type": "WORD",
      "position": 19
    },
    {
      "token": " Failure ",
      "start_offset": 35,
      "end_offset": 36,
      "type": "WORD",
      "position": 20
    }
```

```
  ]
}
```

Index Assignment Analyzer

```
 PUT test
{
  "mappings": {
    "doc": {
      "properties": {
        "k1": {
          "type": "text",
          "analyzer": "bd-nlp-basic" // Use the basic granularity model
        },
        "k2": {
          "type": "text",
          "analyzer": "bd-nlp-phrase" // Use the phrase granularity model
        }
      }
    }
  },
  "settings": {
    "index": {
      "number_of_shards": "1",
      "number_of_replicas": "0"
    }
  }
}
```

Index Assignment Tokenizer

```
 PUT /test
{
   "settings":{
      "analysis":{
         "analyzer":{
            "my_analyzer":{
               "tokenizer":"bd-nlp-basic",   // Customize an analyzer
               "filter":[
                  "lowercase"              // Add filters required by the application
               ]
            }
         }
      }
   },
   "mappings":{
      "properties":{
         "k2":{
            "type":"text",
            "analyzer":"my_analyzer"         // Apply the custom analyzer to the corresponding field
         }
      }
   }
} }
}
```

Accuracy Rate and Recall Rate

Big data set test results in Baidu:

Model	Accuracy rate	Recall rate	F value
analysis-baidu-nlp	98.8%	98.9%	98.8%

Vector Search Plug-in User Guide

The Elasticsearch vector retrieval plug-in is developed by Baidu Elasticsearch team, and can quickly realize vector retrieval, vector computation and other requirements.

Background

In recent years, the vector retrievals based on Text (Document) Embedding, eigenvector, etc. are widely applied in the similarity retrieval of recommended systems and pictures. The user can use Word2vec and other tools to map the image, audio, natural language and other complex data information to eigenvectors, then retrieve the eigenvectors by the vector retrieval algorithm to realize the handling of complex data information. To process the vector data, Baidu Elasticsearch vector retrieval plug-in provides two vector retrieval algorithms: Linear algorithm and hnsw algorithm.

Algorithm	Meaning	Applicable Scenario	Disadvantages	Support distance algorithm
linear	Linear computation of all vector data	The recall rate is 100%. The query time is proportional to the data volume. It is usually used for effect contrast.	The efficiency is lower under the large data volume. Consume cpu All-In-Memory	Cosine distance (cosine) Euclidean distance (l2) Dot product (dot_prod)
hnsw	Conduct approximate calculation of data based on the hnsw algorithm	The single machine data volume is small. High requirements on the recall rate High requirements on the query speed.	The data expansion ratio is higher. An index should be built after the data are written. All-In-Memory	Cosine distance (cosine) Euclidean distance (l2)

Cluster Preparation

	Recommendation	Description
Cluster selection	At least memory of 16G above	The vector retrieval has a higher requirement on the cluster memory. If the data volume exceeds 10G, it is recommended to select the 16-core package of 64G above, such as computation 2 type, computation 3 type and storage 3 type.
Single machine data volume	It is recommended not to exceed one third of the total node memory.	The vector retrieval has a higher requirement on the cluster memory.
Writing traffic limit	Take the computation 2 type (16-core and 64G) node for example. It is recommended that the single node writing traffic limit is controlled within 4000tps.	The building of vector index is a CPU intensity task. It is recommended not to write data with large traffic. Because all the data are loaded into the system memory in the query process, do not write data with large traffic at the same time during the query process.

Method of Application

Before writing data, the user should configure the knn parameters according to the vector dimension information and performance requirements of business, select the distance computation algorithm, and create the required knn index. You can write data after building the index. After building the index, you can conduct the vector retrieval query by the query mode provided below.

Create the knn index

We should create the knn index in advance by the following method:

As shown in the following example below, we create an index with the name of `test-index`, including `field1` and `field2`. You can also customize the index name and field name according to your own requirements.

```
 PUT /test-index
{
    "settings": {
      "index": {
        "codec": "bpack_knn_hnsw",
        "bpack.knn.hnsw.space": "cosine",
        "bpack.knn.hnsw.m": 16,
        "bpack.knn.hnsw.ef_construction": 512
      }
    },
    "mappings": {
      "properties": {
        "field1": {
          "type": "bpack_vector",
          "dims": 2
        },
        "field2": {
          "type": "bpack_knn_vector",
          "dims": 2
        }
      }
    }
}
```

Parameter	Description
index.codec	The `bpack_knn_hnsw` supports *hnsw algorithm* and *linear algorithm*. Or it only supports *linear algorithm*.
type	The vector retrieval plug-in provides two new vector field types, `bpack_vector` and `bpack_knn_vector`. `bpack_vector` represents a common vector field and supports *linear algorithm.* ; `bpack_knn_vector` represents a vector search field and supports *linear algorithm* and *hnsw algorithm*.
dims	Vector dimension, supporting 2~2048 dimensions.

The *bpack.knn.hnsw parameter* meaning in settings is as shown in the *index level parameter optimization* below.

Write and query data

Write data

We write data in the *_doc* of index *test-index* we created just now, and the example of writing data is as below:

```
 POST /test-index/_doc/
{
   "field1" : [6.5, 2.5],
   "field2" : [6.5, 2.5],
   "price" : 10
}
```

And `field1` is the field of `bpack_vector` type we just set. `field2` is the field of `bpack_knn_vector` type we just set. `price` stands for other common fields.

After building an index, we can query the data as below:

linear query

The linear algorithm can query the field of `bpack_knn_vector` as well as that of `bpack_vector` type. In the following example, we query the field `field1` of `bpack_ vector`.

```
 POST /test-index/_search
{
  "query": {
    "script_score": {
      "query": {
        "match_all": {}
      },
      "script": {
        "source": "bpack_knn_script",
        "lang": "knn",
        "params": {
          "space": "cosine",
          "field": "field1",
          "vector": [3.5, 2.5]
        }
      }
    }
  },
  "size": 100
}
Or
POST /test-index/_search
{
 "query": {
  "function_score": {
    "boost_mode": "replace",
    "script_score": {
      "script": {
        "source": "bpack_knn_script",
        "lang": "knn",
        "params": {
          "space": "cosine",
          "field": "field1",
          "vector": [3.5, 2.5]
        }
      }
    }
  }
 },
 "size": 100
}
```

And the query parameter means:

Parameter	Description	Default value
source	Select the computing method, and set it as `bpack_knn_script` here.	Required parameters
space	Distance algorithm parameter. The linear algorithm supports three distance algorithms: Cosine distance （cosine）, Dot product （dot_prod）, Euclidean distance （l2）.	cosine
field	Vector field name.	Required parameters
vector	The format is float array. The array length must be consistent with the dims specified by the field mapping when the index is created.	Required parameters

hnsw query

When we use hnsw for query. The index must specify `index.codec` as `bpack_knn_hnsw`. Meanwhile, the type specified by the vector field mapping to be queried must be `bpack_ knn_ hnsw`. In the following example, we query the field `field2` of `bpack_knn_ vector`.

```
 POST /test-index/_search
{
    "size" : 10,
    "query": {
        "knn": {
            "field2": {
                "vector": [3, 4],
                "k": 2,
                "ef": 512
            }
        }
    }
}
```

And the query parameter means:

Parameter	Description	Default value
vector	The format is float array. The array length must be consistent with the dims specified by the field mapping when the index is created. Or the results may have errors.	Required parameters
k	The value taken for the nearest number queried in the hnsw algorithm is a positive integer.	Required parameters
ef	This parameter represents the size of the nearest dynamic scanning zone during the search period. The higher the value is, the higher the query accuracy rate is, and the slower the query speed is. The value range is [2,1024].	512

Parameter Optimization

Index level parameter

The index *settings* parameter must be provided when an index is created. The default value is used when these settings are not provided. These settings are static. This means that you can't modify the created indexes. The specific parameter analysis is as below:

Parameter	Description	Default value
bpack.knn.hnsw.m	This parameter represents the number of tow-way links created for each new element during its building period. The reasonable range of m is 2-100. It mainly impacts the memory, storage consumption and accuracy. The higher m value means a higher-consumption memory and storage, slower index building time and a higher accuracy rate. It is recommended to take values according to (vector dimension *1.5) to guarantee the performance. The values 12-48 can satisfy the requirements of most scenarios.	16
bpack.knn.hnsw.space	Distance algorithm of vector retrieval computation Distance algorithm parameter. Hnsw Support two distance algorithms: Cosine distance （cosine）、 Euclidean distance （l2）。	cosine
bpack.knn.hnsw.ef_construction	This parameter represents the size of the nearest dynamic scanning zone during the index building period. The higher the value is, the higher the query accuracy rate is, but the slower the index building is. The value range is [2,1024].	512

Cluster level parameters

Common parameters

Parameter	Description	Default value
bpack.knn.hnsw.index_thread_qty	This parameter represents the number of threads allowed for building graphs by HNSW. (by default, nmslib sets this value as the number of cores n. However, Elasticsearch can create n threads for generating indexes. If each index thread can call nmslib to build graphs, that is to say, each thread generates n threads, this may cause simultaneous running of n^2 threads, and 100% CPU utilization ratio. Therefore, this value is set as 1 by default). The value range is [1,32].	1

Cache settings Settings of cache parameters of linear algorithm

Parameter	Description	Default value
bpack.knn.memory.cache.limit	This parameter indicates the maximum capacity of cache. When the cache attempts to load the data and the data exceed the maximum capacity limit of cache, the eviction operation is trigged. This value can be set as a percentage, and represents the percentage of jvm memory. It can be also set as a value with the storage capacity unit, such as 『10kb』, 『10mb』and 『3g』, It is recommended not to set a fractional value, such as『1.5g』.	10%
bpack.knn.memory.cache.expiry.time	This parameter indicates that the data are cleared from the cache when the data are not accessed in the duration. It is expressed in TimeUnit format, such as 『10s』, 『10m』and 『3h』. It is can't be set as a fractional value, such as『1.5h』. Generally speaking, we set this value for over 30 minutes to ensure the cache result can be effectively hit by the following queries; if a too small value is set, it is cleared quickly.	30m

Settings of cache parameters of hnsw algorithm

Parameter	Description	Default value
bpack.knn.cache.item.expiry.time	This parameter indicates that the data are cleared from the cache when the data are not accessed in the duration. It is expressed in TimeUnit format, such as 『10s』,『10m』and 『3h』. It is can't be set as a fractional value, such as『1.5h』. Generally speaking, we set this value for over 30 minutes to ensure the cache result can be effectively hit by the following queries; if a too small value is set, it is cleared quickly.	180m

Settings of Circuit Breaker

The hnsw algorithm consumes a lot of out-of-core memory. If the consumed memory is too much, the pagecache which can be used by Elasticsearch/Lucene is insufficient, and the cluster performance declines. To avoid this situation, we can configure *Circuit Breaker* to limit the excessive consumption of out-of-core memory. Currently, when the memory reaches the breaker limit we configure, the eviction mechanism is triggered to trigger the cache items which are uncommonly used.

Parameter	Description	Default value
bpack.knn.memory.circuit_breaker.limit	This parameter indicates the maximum capacity of cache. When the cache of hnsw exceeds the maximum capacity limit of the cache, the eviction operation is trigged and the *circuit_breaker_triggered* status is set as *true* (can be queried by the query of statistical information api). This value can be set as a percentage, and represents the percentage of the remaining memory of the server excluding the jvm of Elasticsearch. It can be also set as a value with the storage capacity unit, such as 『10kb』,『10mb』and『3g』, It is recommended not to set a fractional value, such as『1.5g』. For example, one machine has 100GB of memory and the jvm of Elasticsearch uses 32GB. The default value of *bpack.knn.memory.circuit_breaker.limit* is (60% * (100 -32) = 40.8GB).	60%
bpack.knn.circuit_breaker.unset.percentage	This parameter represents the removal percentage of *Circuit Breaker*. When the cache capacity is smaller than *bpack.knn.circuit_breaker.unset.percentage*, *Circuit Breaker* removes the triggering. The *circuit_breaker_triggered* status is set as false (can be queried by the query of statistical information api).	*75*

Example

```
 PUT /_cluster/settings
{
   "persistent" : {
      "bpack.knn.hnsw.index_thread_qty" : 1,
      "bpack.knn.cache.item.expiry.time": "15m",
      "bpack.knn.memory.cache.limit": "1g",
      "bpack.knn.memory.cache.expiry.time":"10m",
      "bpack.knn.memory.circuit_breaker.limit" : "55%",
      "bpack.knn.circuit_breaker.unset.percentage": 23
   }
}
```

View the related statistical information of hnsw algorithm

The method to query the status is as below:

```
 GET /_bpack/_knn/stats
GET /_bpack/_knn/nodeId1,nodeId2/stats/statName1,statName2
```

The result example is as below:

```
{
  "_nodes": {
    "total": 1,
    "successful": 1,
    "failed": 0
  },
  "cluster_name": "my-application",
  "circuit_breaker_triggered": false,
  "nodes": {
    "HYMrXXsBSamUkcAjhjeN0w: {
      "eviction_count" : 0,
      "miss_count" : 1,
      "graph_memory_usage_kb" : 1,
      "cache_capacity_reached" : false,
      "load_exception_count" : 0,
      "hit_count" : 0,
      "load_success_count" : 1,
      "total_load_time_nanos" : 2878745
    }
  }
}
```

Cluster status parameter:

Parameter	Description
circuit_breaker_triggered	Indicate whether the circuit breaker is triggered. If any node in the cluster deletes items from the cache because it has reached the cache capacity, the circuit breaker is triggered. When the number of items in the cache is less than *bpack.knn.circuit_breaker.unset.percentage*, the circuit breaker cancels triggering.

Node status parameter:

Parameter	Description
eviction_count	Indicate the frequency the cache is eliminated in guava cache. (those caused by index deletion are not computed)
hit_count	Cache hits occurring on the node.
miss_count	Cache non-hits occurring on the node.
graph_memory_usage_kb	Total size of cached in the memory of the local machine in kb.
cache_capacity_reached	Whether to reach the cache capacity of this node.
load_exception_count	Number of anomalies occurring from loading to the cache
load_success_count	Number of successes occurring from loading to the cache
total_load_time_nanos	Total time consumption from loading to cache, unit: Nanosecond.

Performance Comparison

- Memory configuration: 30G
- cpu Configuration: Number of logical cores: 56, 2 physical cpus, each cpu cores: 14

- Elasticsearch Node: Single node

The performance comparison results are as below:

Data size	Index parameter	Cluster parameters	Top30 recall rate	Average time consumption of hnsw	Average time consumption of linear
1 million 32-dimensional vectors 1shards	"bpack.knn.hnsw.space": "cosine", "bpack.knn.hnsw.m": 16, "bpack.knn.hnsw.ef_construction": 300	"bpack.knn.cache.item.expiry.time": "1h", "bpack.knn.memory.cache.limit": "15g", "bpack.knn.memory.cache.expiry.time":"1h", "bpack.knn.memory.circuit_breaker.limit" : "70%"	99.97%	12.96ms	134.96ms
10 million 32-dimensional vectors 1shards	"bpack.knn.hnsw.space": "cosine", "bpack.knn.hnsw.m": 16, "bpack.knn.hnsw.ef_construction": 600	"bpack.knn.cache.item.expiry.time": "1h", "bpack.knn.memory.cache.limit": "15g", "bpack.knn.memory.cache.expiry.time":"1h", "bpack.knn.memory.circuit_breaker.limit" : "70%"	99.97%	24.69ms	1209.13ms
10 million 32-dimensional vectors 16shards	"bpack.knn.hnsw.space": "cosine", "bpack.knn.hnsw.m": 48, "bpack.knn.hnsw.ef_construction": 600	"bpack.knn.cache.item.expiry.time": "1h", "bpack.knn.memory.cache.limit": "15g", "bpack.knn.memory.cache.expiry.time":"1h", "bpack.knn.memory.circuit_breaker.limit" : "70%"	99.99%	20.26ms	609.56ms

Algorithm Summary

- Applicable scenarios of linear algorithm:
  - Small data volume (the single segmentation is usually below 100w);
  - First execute the normal search filter condition, and then conduct vector retrieval computation on the filtered result set.
  - The recall rate is 100%, and the query performance is slower compared with hnsw.
- Applicable scenarios of hnsw algorithm:
  - The data volume is relatively large (cluster data volume is at the level of tens of millions)
  - The vector retrieval computation and other filterings are conducted simultaneously. It is recommended to appropriately increase the query parameter *k* of hnsw to guarantee that the data satisfying filtering conditions can be involved in the computation;
  - The query performance requirement is high, and the recall rate is 90% above.

Best Practices

- It is recommend to conduct regularly forceMerge in the business low peak period after writing to reduce the query delay.
- When using the linear algorithm for query, you should define the "bpack.knn.memory.cache.limit" parameter according to the data volume. For example, if the node data volume is 10G, and the default value of "bpack.knn.memory.cache.limit" is used (the default value of computation 2 type is 30G*10%=3G), the cache is unavailable. The bulk query may trigger the fusing operation of Elasticsearch, and the error circuitBreakingException is reported.
- When a vector index is built for a larger data volume, the building may be slow. You can appropriately adjust the "bpack.knn.hnsw.index_thread_qty" before writing data according to the partitions and node cpu cores. For example, 1kw data, 1 node and 2 partitions, and 16-core cpu for the node. We can set "bpack.knn.hnsw.index_thread_qty" as 4-6 (if we set it as 8, the cpu can be fully loaded, and the production environment may be in risk) to improve the building efficiency.

  It should be noted that a higher "bpack.knn.hnsw.index_thread_qty" parameter set may cause excessive start threads in the building. In a cluster with a higher load, it is not recommend to adjust this parameter to avoid full load of the cluster. If it is slow to write and build the vector, you can quicken building by temporarily reducing the cluster loads (reduce other writes and queries) and enlarging "bpack.knn.hnsw.index_thread_qty", and then adjust "bpack.knn.hnsw.index_thread_qty" to 1 after building.
- When the written data volume is 1kw (for example about 10G), 1 node 1 partition, and computation 2 type node (16-core cpu and 64G memory), it is recommend to set the parameter as:

```
 PUT /_cluster/settings
{
   "persistent" : {
      "bpack.knn.hnsw.index_thread_qty" : 1,
      "bpack.knn.cache.item.expiry.time": "1h",
      "bpack.knn.memory.cache.limit": "12g",
      "bpack.knn.memory.cache.expiry.time":"1h",
      "bpack.knn.memory.circuit_breaker.limit" : "70%"
   }
}
```

> Analysis:
>
> 1. `"bpack.knn.hnsw.index_thread_qty" : 1` : Generally, it is recommended to set it as 1; when the index building is too slow, you may appropriately adjust this parameter by reference to the recommendations above.
> 2. `"bpack.knn.cache.item.expiry.time": "1h"` : You can set the timeout according to your own business.
> 3. `"bpack.knn.memory.cache.limit": "12g"` : The data volume is about 10G. The cache should accommodate all the data.
> 4. `"bpack.knn.memory.cache.expiry.time":"1h"` : You can set the timeout according to your own business.
> 5. `"bpack.knn.memory.circuit_breaker.limit" : "70%"` : The default jvm memory of computation 2 type Elasticsearch is 30G. The "bpack.knn.memory.circuit_breaker.limit" is 70%*(64-30)=23.8G, it can accommodate the out-of-core memory occupied by data.

FAQ

- Q : How is the recall rate defined?

  A: Use the same vector to query two query mode. Compare the recalled documents, and get the ratio of two identical documents and recall documents. Now we can get the recall rate of the vector to be measured. We use the recall rate to characterize the accuracy rate of query.
- Q : Why don't the indexed documents increase or completely reach the writing volume, and may the query fail when the writing has been successful?

A: The vector index is built in the refresh or flush period. Although the writing is completed, the vector index building tasks at the background may still continue.

- Q： How to install a vector retrieval plug-in?

  A: The newly applied 7.4.2 cluster has its own vector retrieval plug-in; if you already install the vector retrieval plug-in, you can contact the customer service personnel to assist installation.

Best Practices

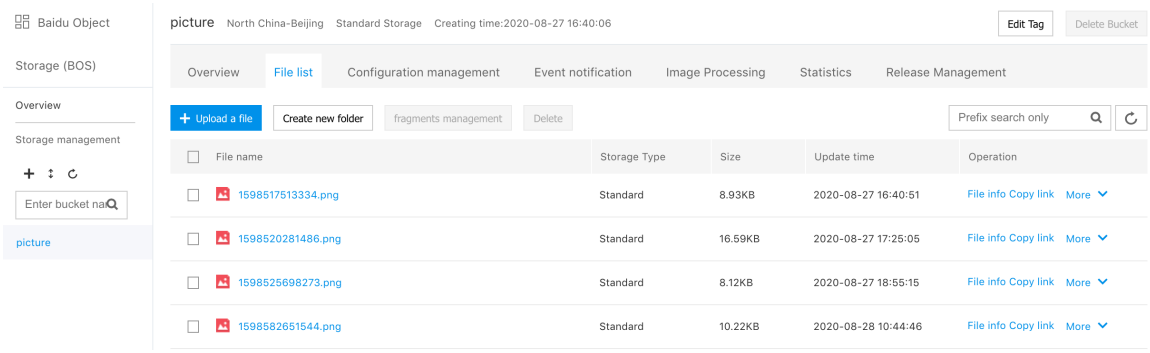
Use the BSC to Import the BOS Data to the Es

Introduction

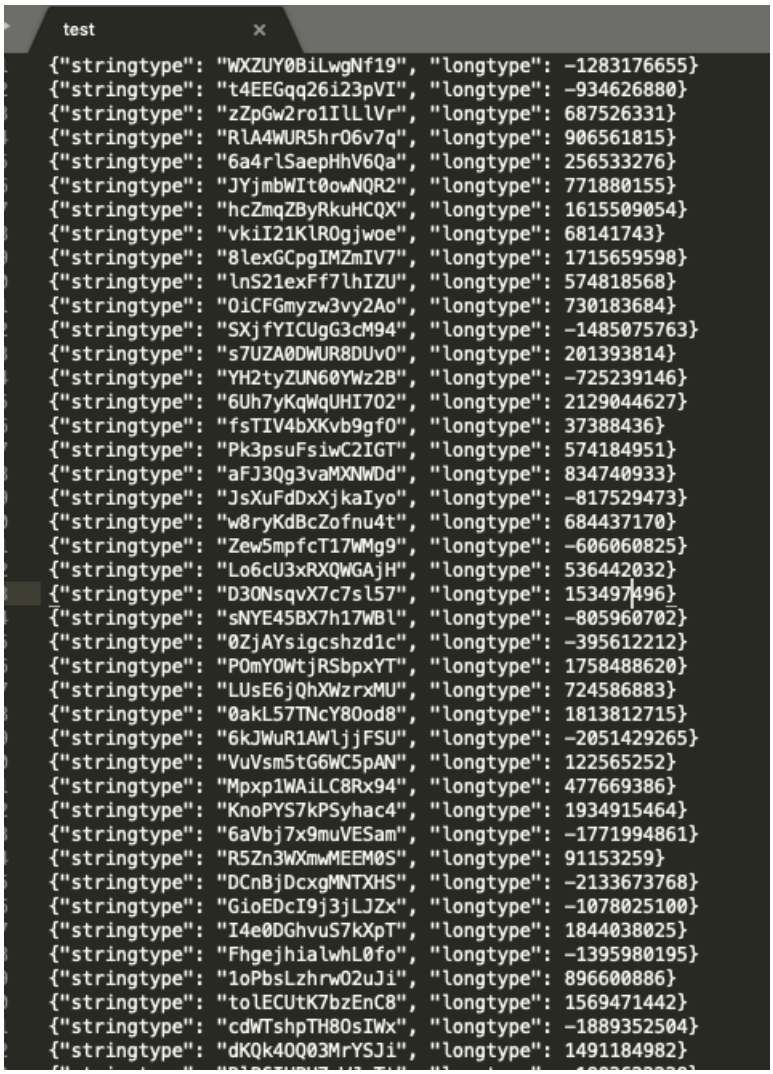
This document mainly introduces how to import data from BOS (Baidu Object Storage System) to the Es through the BSC (Baidu Streaming Computing Service).

Upload the Data to the BOS

Log in to the management console, enter the BOS product interface, create “bucket”, and then upload the test file:



The content of the test file is as follows:



Edit a BSC Job

Create a BOS source

Enter the BSC edit job interface, and create a “bos source table”. The “sql” code is as follows:

```
 CREATE table source_table_bos(
    stringtype STRING,
    longtype LONG
) with(
    type = 'BOS',
    path = 'bos://es-sink-test/test',
    encode = 'json'
);
```

Where, the path is the “bos” path specified in the red box in the above figure, and add the prefix “bos://” before the path.

Create an Es Sink Table

The “sql” code is as follows:

```
 create table sink_table_es(
    stringtype String,
    longtype Long
)with(
    type = 'ES',
    es.net.http.auth.user = 'superuser',
    es.net.http.auth.pass = 'bbs_2016',
    es.resource = 'bsc_test_2/doc_type',
    es.clusterId = '296245916518715392',
    es.region = 'bd',
    es.port = '8200',
    es.version = '6.5.3'
);
```

Where:

- “Es.resource” corresponds to the index and type of “es”. “Es” automatically creates the specified index when “bsc” writes data.
- “Es.clusterId” corresponds to the cluster ID of “es”.
- “Es.region” indicates the code of the region for the “Es” service. You can refer to the “Es” service region code to query the correspondence between the region and code.

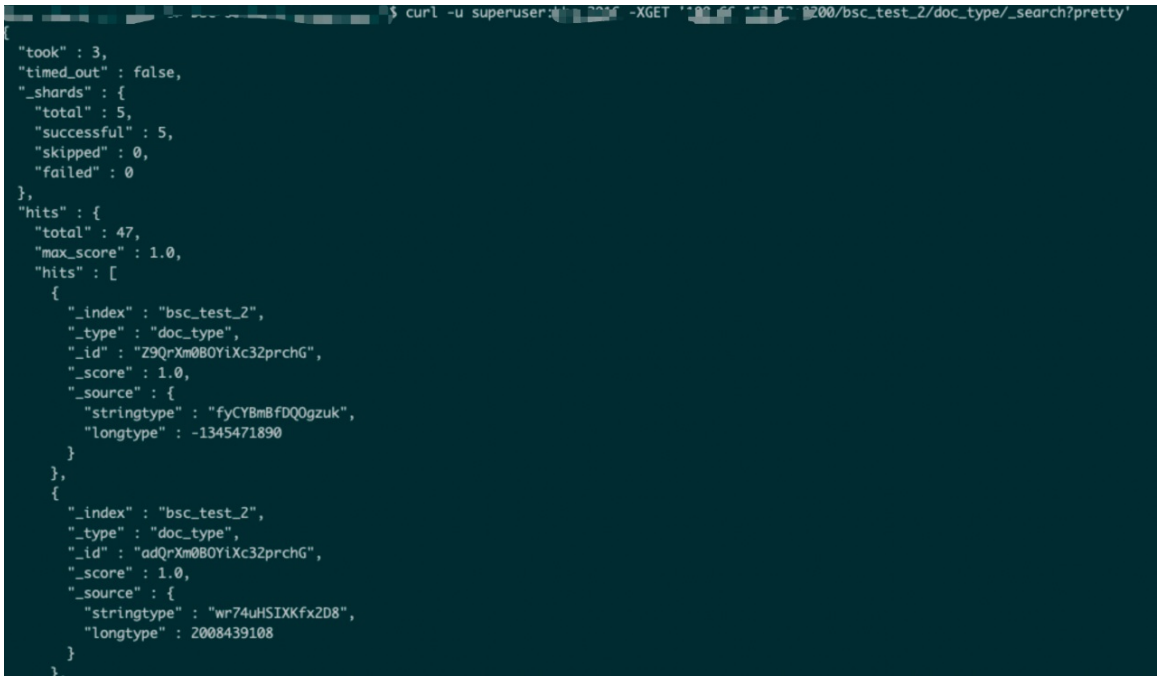
Edit an import statement

The “sql” statement is as follows:

```
 insert into
    sink_table_es(stringtype, longtype) outputmode append
select
    stringtype,
    longtype
from
    source_table_bos;
```

Save a job, release and run the job

Show the Data in the Es



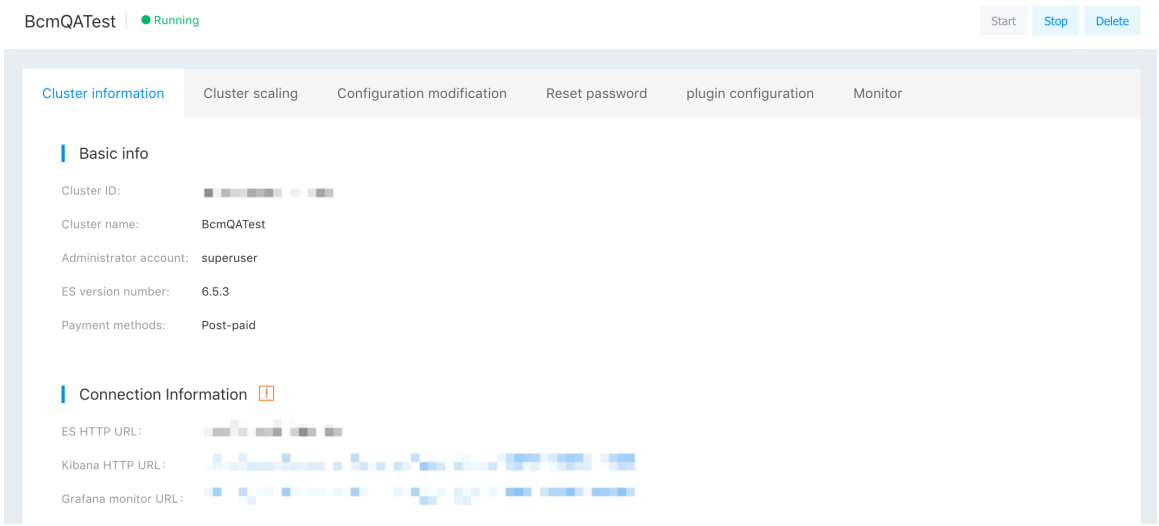
Use the BSC to Import the Kafka Data to the Es

Introduction

This document mainly introduces how to import the data from "Kafka" to "Es" through the BSC 【Baidu Streaming Computing Service】.

Create a Cluster

Before importing the data into the "Es", you need to create an "Es" cluster on the Baidu Cloud. Assume the created cluster information is as follows:

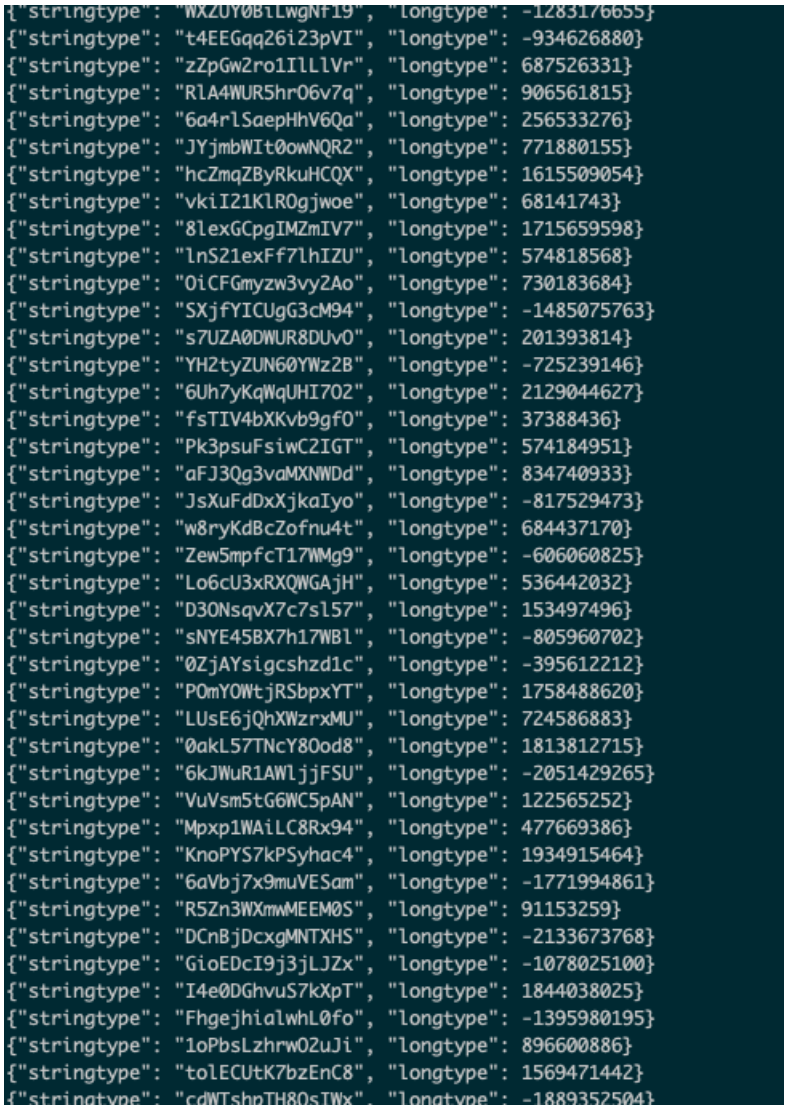


Here, record the following information:

- Cluster ID: 296245916518715392
- Password set during cluster creation: bbs_2016

Create Kafka Topic

Log in to Baidu AI Cloud management console, enter the kafka product interface, create a topic, and inject the data into the created topic. In the example, the injected data is as follows, which includes two json fields. The sample data is as follows:



In the example, create a topic: a15fdd9dd5154845b32f7c74ae155ae3__demo_test, and ensure there's a corresponding certificate under this "topic". Then, download the certificate to the local device.

Edit BSC Job

Create a Kafka Source

Enter the BSC edit job interface and create "kafka source table". The "sql" code is as follows:

```
 CREATE table source_table_kafka(
stringtype STRING,
longtype LONG
) with(
    type = 'BKAFKA',
    topic = 'a15fdd9dd5154845b32f7c74ae155ae3__demo_test',
    kafka.bootstrap.servers = 'kafka.bj.baidubce.com:9091',
    sslFilePath = 'kafka_key.zip',
    encode = 'json'
);
```

Where, "sslFilePath" = 'kafka-key.zip', which is the "kafka" certificate downloaded to the local device at the previous step.

Upload a Kafka Certificate

Click "Advanced Setting", and upload the "kafka" certificate.

After uploading, display the following figure.

Create Es Sink Table

The "sql" code is as follows:

```
 create table sink_table_es(
    stringtype String,
    longtype Long
)with(
    type = 'ES',
    es.net.http.auth.user = 'superuser',
    es.net.http.auth.pass = 'bbs_2016',
    es.resource = 'bsc_test/doc_type',
    es.clusterId = '296245916518715392',
    es.region = 'bd',
    es.port = '8200',
    es.version = '6.5.3'
);
```

Where:

- "Es.resource" corresponds to the index and type of "es". "Es" automatically creates the specified index when "bsc" writes data.
- “Es.clusterId” corresponds to the cluster "ID" of "es".
- “Es.region” indicates the code of the region for "Es" service. You can refer to the "Es" service region code to query the correspondence between the region and code.

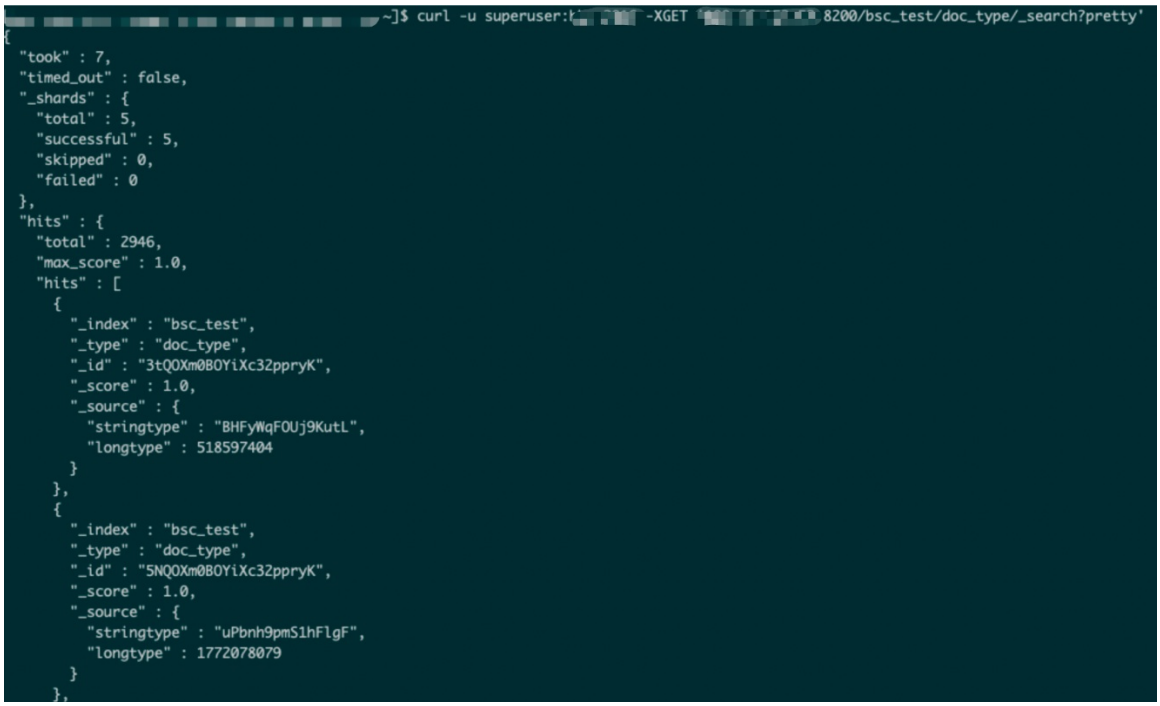
Edit Import Statement

The "sql" statement is as follows:

```
 insert into
    sink_table_es(stringtype, longtype) outputmode append
select
    stringtype,
    longtype
from
    source_table_kafka;
```

Save a job, and release and run it

View the Data in the Es



FAQs

FAQ Overview

FAQs About the Elasticsearch System

- How to show plug-ins that installed by Es
- Error caused by full queue in thread pool
- Too Many Open Files error
- How large a multipart is in Es
- What to do when the cluster is red or yellow
- How to cancel slow log
- PageCache plays a very important role in query
- Disable privilege verification
- Supported Client types
- Does Es support Spark and Hadoop to write or read data
- Several situations of JVM FULL GC
- How to improve import performance
- FAQs about cluster onfiguration
- FAQs about cluster restart
- Disable _field_names
- Several cases when the imported data is found to be slower and slower
- Control the number of indexes on nodes
- Control the number of multiparts and replicas of the index

- When the cluster is in a recovery state, the recovery speed may be slow

FAQ About Spark Accessing Es

- Notes
- Version number detection exception
- Data node discovery
- User privilege problem
- Way to solve configuration Bulk import error issue
- How to get other document metadata except _source
- Specify the “id” and “version” methods when importing
- By default, the "refresh" Is called at the end of each "bulk" when “spark” writes ES
- Control the amount of each Bulk writing
- Read related settings
- Control the document fields to write
- Run the upsert operation

FAQs About the Elasticsearch System

The primary purpose of this document is as follows:

1. To accumulate users' FAQs.
2. To complete all operations in the document using sense, and need to install the sense plug-in into the chrome browser.

How to check plug-ins installed in the Es

You can use the following API to list the plug-ins installed on each node.

```
GET /_cat/plugins
```

Error caused by full thread pool queue

The exceptions thrown by the ES in this scenario are as follows:

```
 rejected execution of org.elasticsearch.transport.TransportService$4@c8998f4
on EsThreadPoolExecutor[bulk, queue capacity = 50,
org.elasticsearch.common.util.concurrent.EsThreadPoolExecutor@553aee29
[Running, pool size = 4, active threads = 4, queued tasks = 50, completed tasks = 0]]
```

There are many thread pools in the ES, such as index, search, and bulk, which are the three typical thread pools we can see. If the system is under great pressure and the backend thread cannot handle it, the tasks initiated by the user pile up in the queue of the thread pools. In the case of reaching the upper limit of the queue, the ES throws a corresponding exception. In the case of such an error, you need to take the following two steps:

- Check the system's CPU and IO utilization. If the system's IO and CPU utilization remain high, this indicates that the

system has encountered a resource bottleneck and cannot avoid such error by optimizing the system parameters. The user on the Baidu AI Cloud can view the ES CPU utilization at the ES console or through the ES's command. You can see the CPU utilization and a load of each node of the ES through the following command:

```
GET /_cat/nodes?v
```

- If there is no problem with resources, check the configuration of the leading thread pool. For the error above, check the configuration of bulk's thread pool, and execute the following command in the sense:

```
GET /_cluster/settings
```

The result is as follows:

```
 "thread_pool": {
       "bulk": {
         "type": "fixed",
         "size": "4",
         "queue_size": "50"
       }
    }
```

The result indicates that there are 4 execution threads for the thread pool handling the bulk task, and the queue number is 50. According to our experience, this value is relatively small so that you can handle it with the following operation:

ES 5.5.0 version:

```
 PUT /_cluster/settings
{
  "persistent": {
     "thread_pool.bulk.size": 32,
     "thread_pool.bulk.queue_size": 300
  }
}
```

ES 6.5.3 + cluster:

```
 PUT /_cluster/settings
{
   "persistent": {
      "thread_pool.write.size": 32,
      "thread_pool.write.queue_size": 300
   }
}
```

Too Many Open Files error

If this error occurs in the es log, generally, there are too many open files. The ES recommends that the limit of file handles should be at least 65536. The user can modify it by modifying /etc/security/limits.conf, or modify it using the command ulimit. In the es, each shard is a separate lucene index writer. Each shard consists of multiple segments, and each segment contains multiple files. Thus, the number of open files = shard number * segment number * the number of files contained in each segment. So, we suggest that the number of shards on a physical machine node is about 1000, and not too many shards. Also, the lucene can effectively reduce the number of files in each segment by using the compound file format.

How large is a shard in the Es?

Each shard of the ES is an index of the lucene, and an index of the lucene can only store 2 billion documents, so a shard can

only store up to 2 billion documents. Additionally, the size of a shard should be between 10 G and 50 G. If it is too large, the query is slow. Also, it takes more time to repair the replica. If the shard size is too small, it leads to excessive shards in an index, and too large fanout brought by the query.

What to do when the cluster is red or yellow?

If the cluster is RED, it indicates no primary shard assigned in the cluster, and yellow indicates a failure to assign a replica. Recommend you to use the following API to see the reason for not assigning the shard to a node.

```
GET _cluster/allocation/explain
```

According to our experience, the ES does assign the shard due to the following circumstances:

- No node has the storage space to put down this shard.
- If the shard is a replica, the primary shard may be unassigned or in the initializing state.

When the shard is in the unassigned state for a long time, the ES tries to assign an unassigned shard in ES 5 times, and if failed, do not try again. In this case, you need to call and manually control the cluster for handling the unassigned shard:

```
POST /_cluster/reroute?retry_failed=true
```

How to cancel a slow log?

Sendings a query by the user may cause a cluster to be very slow, and the CPU utilization rate very high. Thus, the user sometimes wants to cancel the query occupying too many resources. After the release of the ES 5.0, the es provides the command to cancel the query. The es encapsulates all the execution tasks into a task. You can view a list of tasks being executed by a node through the task api, or you can cancel tasks using the task api. For example, if desired to query all tasks executing the search type, you can use the following API:

```
GET /_tasks?actions=*search
```

Cancel all search tasks being executed:

```
POST _tasks/_cancel?actions=*search
```

For more methods for use, see the introduction of the Official Website.

PageCache plays a vital role in the query

Recommend that the ES should be given as much pagecache as possible if the condition permits. It may significantly optimize the query speed. If the pagecache is insufficient, the ES reads the disk every time for querying the 【fetch Document, fetch posting list】. And, the system gets slow down. The user can use the iostat to view the IO information of the system or search for "io_stats" in the information returned by GET _nodes/state. If the iops remains high, the io of the system keeps high, due to the possible cause that the pagecache is small.

Disable the privilege verification

Sometimes the original ES service of the business system is not subject to privilege verification. However, the ES service on the cloud has passed the privilege verification. If you do not want to change the code when migrating the business system, you can disable the privilege verification for a smooth migration. The operation method is as follows:

```
 PUT /_cluster/settings
{
   "persistent": {
      "simpleauth.enable":false
   }
}
```

Supported client types

At present, the products on the Baidu AI Cloud only support the http-based restful api, and do not support the tcp-based transport client api. The main reason for this setting is that the transport client is deeply bound to the cluster running version. During the cluster upgrading, you need to upgrade the frontend services as well.

Does the Es support the Spark and Hadoop to write or read data?

Yes, you need to download the es-hadoop package from the es official website and put it into the spark or hadoop to read and write the es with spark or Hadoop.

Several situations for JVM FULL GC

FullGC caused by Scroll

When some users use the scroll to do paging query or export data with scroll, they often set a long scroll timeout, e.g., 1 day. In this case, the es backend always saves the corresponding search context for this scroll, and each search context corresponds to the lucene's searcher. At this moment, the failure to release the searcher results in the lucene-merged files being not deleted. Some leafreader and fst remain in the JVM for a long time, resulting in FullGC with more and more search contexts. The user can use the following 2 APIs to view and clear these contexts.

```
 GET /_nodes/stats/indices/search
DELETE /_search/scroll/_all
```

FullGC caused by query

The user sets a long from+size of the result set when querying, e.g., it is due to size=Integer.MAX_VALUE. At present, the ES opens a priority queue according to the set from+size. When the concurrency is large, the memory assignment gets failed. So, FULL GC or even OOM results, due to many large queues.

FullGC caused by aggregation

If there are many different values when the user executes a similar terms agg, generate many buckets, such as tens of millions of buckets. These buckets also exist in memory, causing the fullgc.

How to improve import performance?

Reduce the number of replicas and extend the refresh interval

```
 PUT /index_name/_settings
{
    "index.number_of_replicas": 0,
      "index.refresh_interval": "10s"
}
```

The multi-replica mechanism of the ES sends the original json document to multiple replicas during the writing. And, it performs segmentation, index building, and other operations on multiple replicas, respectively. For the import is a CPU intensive operation, changing the number of replicas to 0 can reduce the CPU utilization. After the import is complete, restore the number of replicas, thus directly replicating the physical files rapidly.

The refresh interval aims to control how long the data in the memory gets brushed out of the segment. The es merges the brushed out segments. If the merge fails, the es can prevent writing. So, you can extend the refresh interval, or increase the brushed segment, reduce the merging frequency, and improve the import performance.

Increase the import speed limit for index

```
 PUT /_cluster/settings
{
    "persistent" : {
        "indices.store.throttle.max_bytes_per_sec" : "200mb"
    }
}
```

The ES has a speed limit when writing data to prevent occupying too much disk IO. If the cluster import is large and there are a small number of queries, you can increase this speed limit.

Cluster configuration problem

- Oracle JDK version 1.8 or above is required.
- Set the maximum number of files:

  ```
   Modify /etc/security/limits.conf:

  *      soft      nofile      65536

  *      hard      nofile      65536
  ```

- Increase the mmap counts:

  ```
   Modify /etc/sysctl.conf:

  vm.max_map_count=262144

  Then execute: sysctl -p
  ```

Cluster restart problem

In some cases (such as modify the configuration file), you need to restart the cluster. You can restart the clusters one-by-one or by restarting the whole clusters. Restarting the es may cause the redistribution of data. The following describes how to restart the services in these two cases.

Restart of the whole cluster

- Set the whole cluster to the read-only state

  ```
   PUT /_cluster/settings
  {
    "persistent": {
      "cluster.blocks.read_only":true
    }
  }
  ```

- Flush all data in the node memory to the disk.

  ```
   POST /_flush/synced
  ```

- Restart all es nodes

- After the cluster is green, modify the cluster to the writable state.

```
  PUT /_cluster/settings
{
   "persistent": {
      "cluster.blocks.read_only":false
   }
}
```

Restart one by one

This method does not result in the restart service being interrupted and is applicable to online service.

- The shard assignment is prohibited, so that when we disable one Es service, without the shard redistributed.

```
  PUT /_cluster/settings
{
         "transient" : {
         "cluster.routing.allocation.enable" : "none"
        }
}
```

- Disable a single node, modify the configuration or replace the jar package, and start the node.
- Start the shard redistribution.

```
  PUT /_cluster/settings
{
    "transient" : {
       "cluster.routing.allocation.enable" : "all"
    }
}
```

- After the cluster becomes green, repeat Steps 1-3 until the configuration of all nodes gets modified.

Disable _field_names

The `_field_names` field is an internal metadata field of the Elasticsearch. This field indexes the name of each field in the document (except the field name with a field value of null), which is mainly to execute the Elasticsearch `exists` query. The Elasticsearch only performs indexing for this field and does not store this field. After version 6.3, this field only indexes those fields with doc_value and norms not disabled. Recommend you to disable this field without using the `exists` query for your business. Thus, it is possible to slightly reduce the storage space occupied by the inverted index list and appropriately enhance the utilization of pagecache.

```
 PUT index
{
  "mappings": {
    "_doc": {
      "_field_names": {
        "enabled": false
      }
    }
  }
}
```

Several cases where the data import gets slower and slower

Update contained in the imported data

In fact, the update of the Es is to read the data, and then write it after being changed. When writing more and more data, it becomes slow to read and write the data.

Control the number of indexes on nodes

By default, the ES cluster tries to balance the number of indexes and shards on all nodes. However, it may cause too many shards of an index concentrated on a few nodes. In this case, you can set the number of index shards stored in each node in the cluster:

```
 PUT {index name}/_settings
{
 "index.routing.allocation.total_shards_per_node": 10
}
```

Control the number of shards and replicas of the index

Without modifying the parameters, an index has 5 shards and 2 replicas (including the main shard), which can be controlled by modifying the parameters of the index:

```
 PUT /{index name}
{
  "settings": {
     "number_of_shards": 20,
     "number_of_replicas": 2
  }
}
```

```
 number_of_shards: Means the number of shards, which cannot be modified after creating the index. You must specify it during creation.
number_of_replicas: The number of replicas, excluding the main shard.
```

The recovery speed may be slow when the cluster is in a recovery state

You can view the recovering index shards through

```
 GET /_recovery?active_only=true
```

By default, 4 index shards get recovered simultaneously recovered on one node, including 2 for source nodes and 2 for target nodes. When the number of shards is huge, the recovery may be slow. By default, the maximum speed limit is 40 mb during recovery. In this case, you can set cluster parameters:

```
 curl -XPUT "host:port/_cluster/settings" -d'
{
   "transient": {
      "cluster.routing.allocation.node_concurrent_recoveries": 8,
      "indices.recovery.max_bytes_per_sec": "120mb"
   }
}'
```

```
 indices.recovery.max_bytes_per_sec： Means the maximum bandwidth for node recovery. This setting should be smaller than the current network bandwidth to avoid affecting other network services.
cluster.routing.allocation.node_concurrent_recoveries： Means the maximum number of concurrent recoveries when the node is a source node or target node.
```

How to recover after the disk is full

When the disk usage rate of the DataNode of the Es reaches a certain threshold (95%), the Es can prevent further writing, and add a block to all Indexes. When the user continues to write, they can receive the following error.

```
cluster_block_exception [FORBIDDEN/12/index read-only / allow delete (api)];
```

In this case, the user must release the disk space to solve the problem. There are two ways to release the disk space:

- Delete the unused Index
- Reduce the number of replicas of the Index. For example, reduce the number of replicas from 2 to 1.

After the disk space is released, the Es cannot automatically remove the block. In this case, the user still cannot write in data, but need to execute the following command:

```
 curl -XPUT "host:port/_all/_settings" -d '
{
   "index.blocks.read_only_allow_delete": null
}'
```

FAQ About Spark Accessing Es

Introduction

This document mainly introduces the meaning of common configuration items when accessing the ES through `Spark` in `elasticsearch-hadoop`. The `es-spark` in this document is the package in `elasticsearch-hadoop` associated with the `Spark`. The user reads and writes the ES cluster through their own `Spark` cluster. The `elasticsearch-hadoop` is substantially compatible with all versions of the ES.

Version number detection exception

The "es-spark" detects the version number of the ES cluster automatically during running. The version number obtained is mainly used for API compatibility processing of the version of different clusters.

Generally, the user doesn't need to focus on the ES version number. However, sometimes some undetectable errors occur when automatically detecting the cluster version number on the Baidu AI Cloud. You can solve the errors through the following configuration:

Configuration items:

```
es.internal.es.version： "6.5.3"
```

Also, you need to perform configuration in some `es-spark` packages of the new version:

```
es.internal.es.cluster.name： "Your Cluster Name"
```

Implementation principle:

After the configuration is complete, the "es-spark" does not request the / directory and parse version, but directly uses the user-configured version:

```
 INTERNAL_ES_VERSION = "es.internal.es.version"
INTERNAL_ES_CLUSTER_NAME = "es.internal.es.cluster.name"

public static EsMajorVersion discoverEsVersion(Settings settings, Log log) {
     return discoverClusterInfo(settings, log).getMajorVersion();
}

// It may vary in the elasticsearch-hadoop of different versions.
public static ClusterInfo discoverClusterInfo(Settings settings, Log log) {
       ClusterName remoteClusterName = null;
       EsMajorVersion remoteVersion = null;
       // Try to acquire the cluster name from the configuration
       String clusterName = settings.getProperty(InternalConfigurationOptions.INTERNAL_ES_CLUSTER_NAME);
       // Try to aquiire the cluster UUID from the configuration
       String clusterUUID = settings.getProperty(InternalConfigurationOptions.INTERNAL_ES_CLUSTER_UUID);
       // Try to acquire the ES version from the configuration
       String version = settings.getProperty(InternalConfigurationOptions.INTERNAL_ES_VERSION);
       // If failed to acquire the cluster name and version number from the configuration file, initiate a network request
(request root directory).
       if (StringUtils.hasText(clusterName) && StringUtils.hasText(version)) {
          if (log.isDebugEnabled()) {
             log.debug(String.format("Elasticsearch cluster [NAME:%s][UUID:%s][VERSION:%s] already present in
configuration; skipping discovery",
                   clusterName, clusterUUID, version));
          }
          remoteClusterName = new ClusterName(clusterName, clusterUUID);
          remoteVersion = EsMajorVersion.parse(version);
          return new ClusterInfo(remoteClusterName, remoteVersion);
       }
     ....
}
```

With the automatic detection feature of the cluster name and version number enabled, you need to guarantee that the user assigned to the "es-spark" has the `GET` privileges for access to the root directory `/`.

```
 GET /
```

Discover the data node

Configuration items:

```
 es.nodes.wan.only :  false It is false by default
es.nodes.discovery: true   It is true by default
```

There is a BLB before the Baidu AI Cloud ES cluster. Write the BLB address when configuring the `es.nodes`, and guarantee that `es-spark` can access the BLB address.

- `es.nodes.wan.only: false`,`es.nodes.discovery: true`: Means that the Spark obtains all the`ip' and`port`with 'HTTP` service nodes of the ES cluster enabled by accessing a `host` (or multiple hosts) specified in the 'es .nodes. The subsequent access to data leads to direct access to the node for the shard data. Make sure that the`Spark' cluster can access all nodes of the ES cluster.
- `es.nodes.wan.only: true`, `es.nodes.discovery: false` or not set: You need to forward all requests sent by the Spark to the ES through this node, so the efficiency is low.

The specific code logic is as follows:

```
 ES_NODES_DISCOVERY = "es.nodes.discovery"
ES_NODES_WAN_ONLY = "es.nodes.wan.only"
ES_NODES_WAN_ONLY_DEFAULT = "false"

InitializationUtils#discoverNodesIfNeeded
    public static List<NodeInfo> discoverNodesIfNeeded(Settings settings, Log log) {
        if (settings.getNodesDiscovery()) { // The configuration items that need to be read.
            RestClient bootstrap = new RestClient(settings);

            try {
                List<NodeInfo> discoveredNodes = bootstrap.getHttpNodes(false);
                if (log.isDebugEnabled()) {
                    log.debug(String.format ("Nodes discovery enabled - found %s", discoveredNodes));
                }

                SettingsUtils.addDiscoveredNodes(settings, discoveredNodes);
                return discoveredNodes;
            } finally {
                bootstrap.close();
            }
        }

        return null;
    }

public boolean getNodesDiscovery() {
        // by default, if not set, return a value compatible with the WAN setting
        // otherwise return the user value.
        // this helps validate the configuration
        return Booleans.parseBoolean(getProperty(ES_NODES_DISCOVERY), !getNodesWANOnly()); //The default value is
“!getNodesWANOnly()”.
    }

public boolean getNodesWANOnly() {
        return Booleans.parseBoolean(getProperty(ES_NODES_WAN_ONLY, ES_NODES_WAN_ONLY_DEFAULT));
    }
```

User privilege problem

To enable the node discovery feature, you need to guarantee the user of the es-spark has the privileges to access

```
 GET /_nodes/http
GET /{index}/_search_shards
```

Otherwise, it may fail to do the whole `Job`.

Way to solve the configuration bulk import error

When the Spark writes ES, an error occurs. The "Job" is interrupted after several attempts. By default, the current "Job" is interrupted directly, causing the failure of the whole task. If you want to print the document failed to import into the log, you can solve it through the following configuration:

Handling mechanism for configuration errors

```
 es.write.rest.error.handlers = log
es.write.rest.error.handler.log.logger.name :  es_error_handler
```

When a "bulk" write error occurs after setting, the "Job" should bot be interrupted and output to the log in the form of a log with the prefix `es_error_handler`.

How to get other document metadata except for the "_Source"

Under normal circumstances, each data that we return by calling the _search API of the ES is as follows:

```
        {
          "_index": "d_index",
          "_type": "doc",
          "_id": "51rrB2sBaX4YjyPY-2EG",
          "_score": 1,
          "_source": {
            "A": "field A value",
            "B": "field B value"
          }
        }
```

When using the Spark to read the ES, do not to read other fields except for the`_source` field by default, such as `_id` `_version`. In some scenarios, you need to get `_id` for your business, through the following configuration:

```
  es.read.metadata:true  //The configuration is false by default.
es.read.metadata.field: "_id" //Configure the metadata fields you need to read.
es.read.metadata.version: It is false by default. Read the version number of "es".
```

The metadata field information of the document is in a `_metadata` field.

Method for specifying the "id" and "version" during import

During data migration, such as migrating from a low version ES cluster to a high version ES cluster, we can read and write at the same time by using the es-spark. To specify information such as`_id`, `_rouring` and`_version`, you can set:

```
  es.mapping.id : ”_meta._id" Specifies the "id" path in the "json".
es.mapping.routing : ” _meta._rouring" Specifies the "id" path in the "json".
Es.mapping.version : ”_meta._version" Specifies the "id" path in "json".
```

`_meta .xxx` is the path of the required field in the Json document.

By default, the "refresh" is called at the end of each "bulk" when the spark writes the ES

Recommend to set it to `False`. `refresh` is controlled by `refresh_interval` of `index` inside the ES. Otherwise, many threads in the ES cluster may bring high CPU and disk pressure during `refesh`.

```
  es.batch.write.refresh: False. It is true by default.
```

Control the amount of each Bulk writing

```
  es.batch.size.bytes:1mb means the size of the bulk writing file, which is 1 MB by default.
es.batch.size.entries:1000 means the number of "bulk" writing files, which is 1,000 by default.
```

The user can reasonably set it according to the ES cluster package.

Read-related settings

```
  es.scroll.size: It is 50 by default. This value is relatively small, so it can increase to 1,000-10,000.
es.input.use.sliced.partitions：  This value is true. To improve concurrency, the “es” performs the “scroll-slice” for slicing.
es.input.max.docs.per.partition：  This value is 100,000, based on which the “Slice” gets done.
```

When the "es-spark" reads the ES cluster data, it performs slicing for `scroll-slice` handling according to the total number of shards.

```
int numPartitions = (int) Math.max(1, numDocs / maxDocsPerPartition);
```

The "numDocs" means the total number of files for a single shard. If there are 50 million files, have it divided into 500 slices, which causes a considerable CPU pressure on the online ES clusters at the backend. Thus, recommend you to disable the `scroll-slice` to avoid affecting the online business.

Recommended parameters:

```
es.scroll.size: 2,000 //Try to select this vaue according to the file size.
es.input.use.sliced.partitions: False
```

Control the file field to be written

Sometimes, when importing data for a business, you may hope not to write some fields into the ES. You can set it as follows :

```
Es.mapping.exclude: None by default.
// You can separate multiple fields with commas, such as ".", and "*".
es.mapping.exclude = *.description
es.mapping.include = u*, foo.*
```

Execute the "upsert" operation

Example:

```
String update_params = "parmas:update_time";
String update_script = "ctx._source.update_time = params.update_time";
// Set sparkConfig
SparkConf sparkConf = new SparkConf()
        .setAppName("YourAppName”)
        .set("es.net.http.auth.user", user)
        .set("es.net.http.auth.pass", pwd)
        .set("es.nodes", nodes)
        .set("es.port", port)
        .set("es.batch.size.entries", "50")
        .set("es.http.timeout","5m")
        .set("es.read.metadata.field", "_id")
        .set("es.write.operation","upsert")
        .set("es.update.script.params", update_params)
        .set("es.update.script.inline", update_script)
        .set("es.nodes.wan.only", "true");
```

Notice:

```
“es.update.script.params” is the parameter list required for updating.
es.update.script.inline is the “script” used for updating.
```