Elasticsearch Service Practice Tutorial

Contact Us

We are committed to providing personalized pre-sales consultation and technical after-sale support. Don't hesitate to contact us at 4009100100 or 95716 for any inquiries or concerns.

Contents

Practice Tutorial
Data Migration and Sync
Data Migration

Last updated: 2024-10-25 09:24:42

If users have self-built ES Clusters on Tencent Cloud or purchased ES Clusters from other cloud providers, and need to migrate to Tencent Cloud ES (suitable for most regular index migrations), they can choose the appropriate Migration Plan according to their business needs. If the business can be stopped or write operations can be paused, the following methods can be used for data migration:

- COS Snapshot
- logstash
- elasticsearch-dump

The comparison of various migration methods is as follows:

Migration Method	Applicable Scenario
COS Snapshot	Scenarios with large data volumes (GB, TB, PB levels) Scenarios requiring high migration speed
logstash	Scenarios for migrating full or incremental data with low real-time requirements Scenarios requiring simple filtering of migrated data using ES query Scenarios requiring complex filtering or processing of migrated data Scenarios with large Version Span migrations, such as migrating from version 5.x to 6.x or 7.x
elasticsearch-dump	Scenarios with small data volumes

COS Snapshot

The migration method based on COS Snapshot uses the ES snapshot API interface for migration. The basic principle is to create an index snapshot from the source ES Cluster, and then restore it in the target ES Cluster. When migrating data via snapshot, special attention must be paid to the ES version issue:

```
The major version number of the target ES cluster (such as 5 in 5.6.4) must be greater than or equal to that
of the source ES cluster.
The snapshot created by the cluster in version 1.x cannot be restored in version 5.x.
```

Creating Repository in Source ES Cluster

Before creating a snapshot, you must create a repository. A repository can contain multiple snapshot files. There are mainly the following types of repositories.

- fs: Shared file system, which stores snapshot files in the file system.
- url: Specifies the URL path of the file system. Supported protocols: HTTP, HTTPS, FTP, file, jar.
- s3: AWS S3 COS. The snapshot is stored in S3 and supported by plugins. See repository-s3 for the installation of this plugin.
- hdfs: The snapshot is saved in the HDFS and supported by plugins. For installation of this plugin, see repository-hdfs.
- cos: The snapshot is stored in Tencent Cloud COS, supported by plugins. See cos-repository for the installation of this plugin.

If you need to migrate from a self-built ES Cluster to Tencent Cloud's ES Cluster, you can directly use the COS repository type. However, you need to install the cos-repository plugin on the self-built ES Cluster first (a cluster restart is required after plugin installation). First, back up the data from the self-built ES Cluster to COS, then restore it in the ES Cluster on Tencent Cloud to complete the data migration.
If it is inconvenient to install the cos-repository plugin on your self-built ES cluster but the repository-s3 or repository-hdfs plugin is already installed, you can first back up the data to S3 or HDFS. Then, upload the backed-up files from S3 or HDFS to Tencent Cloud COS, and perform the recovery on the cluster on Tencent Cloud.
When migrating data using a COS snapshot, you need to first create a COS repository. You can create the repository using the following command:

```
PUT _snapshot/my_cos_backup
{
    "type": "cos",
    "settings": {
        "app_id": "xxxxxxx",
        "access_key_id": "xxxxxx",
        "access_key_secret": "xxxxxxx",
        "bucket": "xxxxxx",
        "region": "ap-guangzhou",
        "compress": true,
        "chunk_size": "500mb",
        "base_path": "/"
    }
}
```

- app_id: APPID of your Tencent Cloud account.
- access_key_id: SecretId of your Tencent Cloud API key.
- access_key_secret: SecretKey of your Tencent Cloud API ley.
- bucket: COS bucket name,**without the appId suffix**.
- region: COS bucket region, which must be the same region as the ES cluster.
- base_path: The backup directory.

Creating Snapshhot in Source ES Cluster

You can call the snapshot API to create a snapshot so as to back up the index data. When creating a snapshot, you can specify to back up certain or all indexes. For more information on the specific API parameters, please see Snapshot and Restore.

Backing Up All Indexes

> **Note**
> Recovering all index data typically involves existing system indexes, which can cause abnormal situations.

Back up all indexes from the source ES cluster to the `my_cos_backup` repository, and name it `snapshot_1`:

```
PUT _snapshot/my_cos_backup/snapshot_1
```

This command returns immediately and executes asynchronously in the background until it completes. If you want the snapshot creation command to block execution, you can add the `wait_for_completion` parameter:

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

> **Note**
> The command execution time is related to the index size.

Backing Up Specified Index

You can specify the index to be backed up when creating a snapshot:

```
PUT _snapshot/my_cos_backup/snapshot_2
{
    "indices": "index_1,index_2"
}
```

> **Note**
> If the parameter indices contains multiple indexes, separate them with `,` without spaces.

Viewing Snapshot Status

Run the following command to check whether the snapshot is backed up. If the `state` field in the returned result is `SUCCESS`, it indicates that the snapshot has been backed up successfully:

```
GET _snapshot/my_cos_backup/snapshot_1
```

Create a repository in the target ES cluster

Creating a repository in the target ES cluster is exactly the same as creating one in the source ES cluster.

Restoring from Snapshot

Restore all indexes backed up in the snapshot to the ES cluster:

```
POST _snapshot/my_cos_backup/snapshot_1/_restore
```

If "snapshot_1" includes 5 indexes, all of them will be restored to the ES cluster. You can also rename the index with additional options. This option allows you to match the index name by schema and provide a new name through the recovery process. You can use this option if you want to recover old data to validate content or perform other actions without replacing existing data. Restore a single index from the snapshot and provide an alternative name:

```
POST /_snapshot/my_cos_backup/snapshot_1/_restore
{
    "indices": "index_1",
    "rename_pattern": "index_(.+)",
    "rename_replacement": "restored_index_$1"
}
```

- indices: Only "index_1" is restored, and other indexes existing in the snapshot are ignored.
- rename_pattern: Finds the index being restored on which the provided pattern can match.
- rename_replacement: Renames a matching index to the replacement schema.

Viewing Index Recovery Status

You can check the recovery progress of a specific index by calling the `_recovery` API:

```
GET index_1/_recovery
```

Additionally, you can call the following API to check the status of a specific index. If the `status` in the returned result is `green`, it means the index has fully recovered:

```
GET _cluster/health/index_1
```

logstash

Logstash supports reading data from one ES cluster and writing it to another. Therefore, you can use Logstash for data migration. Before using Logstash for data migration, pay attention to the following points:

- You need to create a CVM in the same VPC as the ES Cluster on Tencent Cloud, deploy logstash on it, and ensure that the CVM can access the source ES Cluster.
- It is recommended to choose a CVM with higher specifications for deploying Logstash, such as 16 CPU cores and 32GB of memory.
- Logstash should have the same major version number as the target ES cluster. For example, if the target ES cluster is version 6.8.2, Logstash should also be using version 6.8.x.
- Special attention needs to be paid to the issue of index types, as different versions of ES have different constraints on index types, and during the migration of ES clusters across major versions, there may arise issues such as failures in writing to the target cluster due to the index types.
  For more information, see the description of `document_type` parameter in the logstash-output-elasticsearch plugin.

A common configuration file for cross-cluster data migration using Logstash is as follows:

```
input {
    elasticsearch {
        hosts => "1.1.1.1:9200"
        user => "elastic"
        password => "your_password"
        index => "*"
        docinfo => true
        size => 5000
        scroll => "5m"
      }
}
output {
    elasticsearch {
        hosts => ["http://2.2.2.2:9200"]
        user => "elastic"
        password => "your_password"
        index => "%{[@metadata][_index]}"
        document_type => "%{[@metadata][_type]}"
        document_id => "%{[@metadata][_id]}"
    }
}
```

The above configuration file synchronizes all indexes of the source ES cluster to the target cluster. You can also set to only synchronize specified indexes. For more features of migration using Logstash, see logstash-input-elasticsearch and logstash-output-elasticsearch.

elasticsearch-dump

elasticsearch-dump is an open source ES data migration tool, github address.

1. Install Node.js and npm, and configure the environment variables.
2. Install elasticsearch-dump
   elasticsearch-dump is developed using Node.js and can be installed directly using the npm package manager:

   ```
   npm install elasticdump -g
   ```

3. Main Parameter Description

   ```
   --input: Source address, which can be ES cluster URL, file or stdin. An index can be specified in the
   format of: {protocol}://{host}:{port}/{index}
   =--input-index: Index in the source ES cluster
   --output: Target address, which can be ES cluster address URL, file or stdout. An index can be specified in
   the format of: {protocol}://{host}:{port}/{index}
   =--output-index: Index of the target ES cluster
   =--type: Migration type, and the default value is data, indicating that only data is migrated. Optional
   settings: analyzer, data, mapping, alias
   ```

4. If the cluster has security authentication, you can use the reindex cluster authentication method as follows.
   Add user:password@ after the corresponding HTTP. Refer to the example.

   ```
   elasticsearch-dump --input=http://192.168.1.2:9200/my_index --
   output=http://user:password@192.168.1.2:9200/my_index --type=data
   ```

   If the password contains special characters, you need to use an online tool (such as URL Encode) to URL encode the password. For example, if your original password is P@ssw0rd! , after URL encoding it will become P%40ssw0rd%21.
   If it is an HTTPS cluster, you need to add –ca with each cluster's `server-certificates.pem` certificate, and you must add `NODE_TLS_REJECT_UNAUTHORIZED=0` at the beginning

```
NODE_TLS_REJECT_UNAUTHORIZED = 0  elasticsearch-dump --
input = https://elastic:P%40ssw0rd%21@192.168.1.2:9200/my_index -ca /path1/server-certificates.pem --
output = https://elastic:P%40ssw0rd%21@192.168.1.2:9200/my_index  -ca /path2/server-certificates.pem --
type = data
```

5. Migrate a single index
   The following operation uses the elasticdump command to migrate the companydatabase index from cluster 172.16.0.39 to cluster 172.16.0.20.

   > **Note**
   > In the first command, index settings are migrated first. If you directly migrate mapping or data, the configuration information of indexes in the original cluster will be lost, such as the number of shards and replicas. Of course, you can also synchronize mapping and data after creating indexes in the target cluster.

```
elasticdump --input=http://172.16.0.39:9200/companydatabase --
output=http://172.16.0.20:9200/companydatabase --type=settings
elasticdump --input=http://172.16.0.39:9200/companydatabase --
output=http://172.16.0.20:9200/companydatabase --type=mapping
elasticdump --input=http://172.16.0.39:9200/companydatabase --
output=http://172.16.0.20:9200/companydatabase --type=data
```

6. Migrate all indexes
   The following operation uses the elasticdump command to migrate all indexes from cluster 172.16.0.39 to cluster 172.16.0.20.

   > **Note**
   > This operation does not migrate the configuration of indexes, such as the number of shards and replicas. Each index must be migrated separately or the data must be migrated after the index is created directly in the target cluster.

```
elasticdump --input=http://172.16.0.39:9200 --output=http://172.16.0.20:9200
```

Summary

1. When using elasticsearch-dump and logstash for cross-cluster data migration, the machine executing the migration task must be able to access both clusters simultaneously. Migration cannot be performed if the network is not connected. However, the snapshot method does not have this limitation as it is completely offline. Therefore, elasticsearch-dump and logstash are more suitable for migration when the source ES cluster and the target ES cluster are in the same network. For cross-cloud vendor migration, the snapshot method can be chosen, such as migrating from a competitor's ES cluster to Tencent Cloud's ES cluster. Alternatively, network interconnection can be achieved, but it involves higher costs.
2. The elasticsearch-dump tool is similar to mysqldump, which is used by MySQL database for data backup. Both of them are logical backups and need to export the data one by one before importing it. Therefore, they are suitable for migration in scenarios with small amount of data.
3. The snapshot method is suitable for migrating in scenarios with a large amount of data.

Data Ingestion into ES

Last updated: 2024-10-25 09:25:22

Tencent Cloud Elasticsearch Service allows users to access the cluster within their VPC through a VPC VIP. Users can write code to access the cluster and import their data using the Elasticsearch REST Client. They can also use the official components (such as Logstash and Beats) to connect their data.
This document takes the official components Logstash and Beats as examples to describe how to connect your data source of different types to ES.

Preparations

You need to create a CVM instance or a Docker cluster in the same VPC as the ES cluster, as accessing the ES cluster needs to be done within the VPC.

Using Logstash to Access ES Cluster

Accessing ES cluster from CVM

1. Install and deploy Logstash and Java 8.

```
wget https://artifacts.elastic.co/downloads/logstash/logstash-5.6.4.tar.gz
tar xvf logstash-5.6.4.tar.gz
yum install java-1.8.0-openjdk  java-1.8.0-openjdk-devel -y
```

> **Note**
> The Logstash version should be consistent with the Elasticsearch version.

2. Customize the configuration file `*.conf` based on the data source type. Refer to Data Source Configuration File Description for the configuration file content.
3. Run Logstash.

```
nohup .//root/logstash-5.6.4/bin/logstash -f ~/*.conf 2>&1 >/dev/null &
```

Accessing ES cluster from Docker

Creating Docker cluster

1. Pull the official image of Logstash.

```
docker pull docker.elastic.co/logstash/logstash:5.6.9
```

2. Customize the `*.conf` configuration file based on the data source type and place it in the `/usr/share/logstash/pipeline/` directory, which can be customized.
3. Run Logstash.

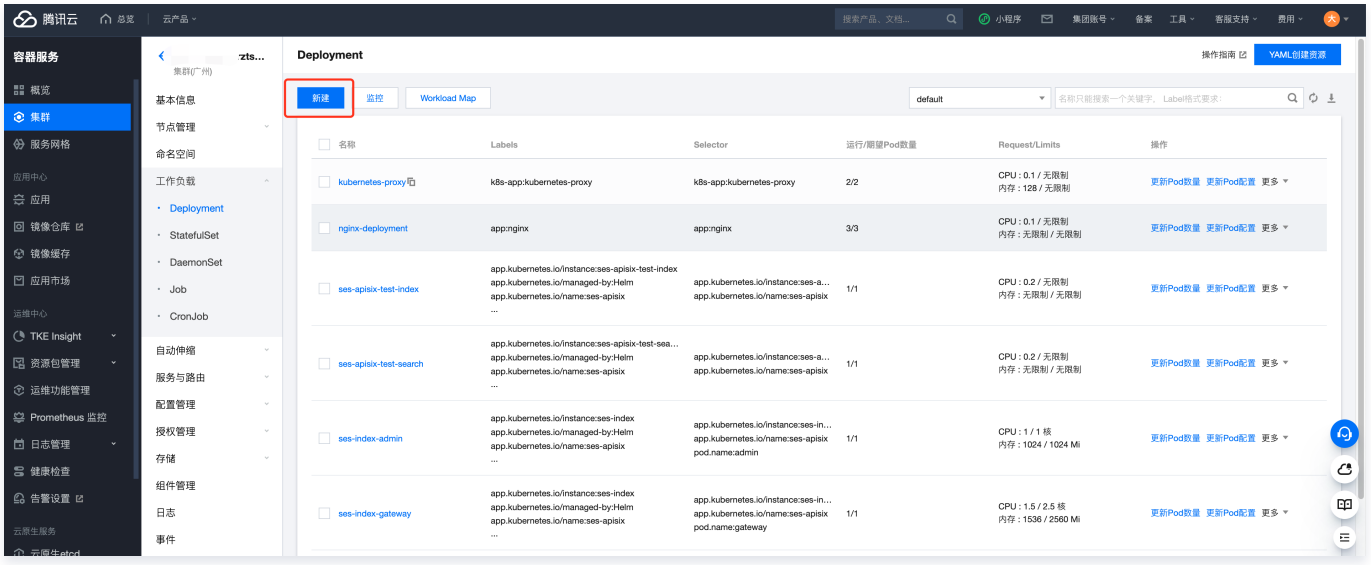
```
docker run --rm -it -v ~/pipeline/:/usr/share/logstash/pipeline/ docker.elastic.co/logstash/logstash:5.6.9
```

Using Tencent Cloud TKE

Tencent Cloud Docker clusters run on CVM instances, so you need to create a CVM cluster in the TKE console first.

1. Log in to TKE Console, select **Cluster > New** from the left menu to create a cluster.

2. Select the **Services** from the left menu bar, click **Create** to create a service.

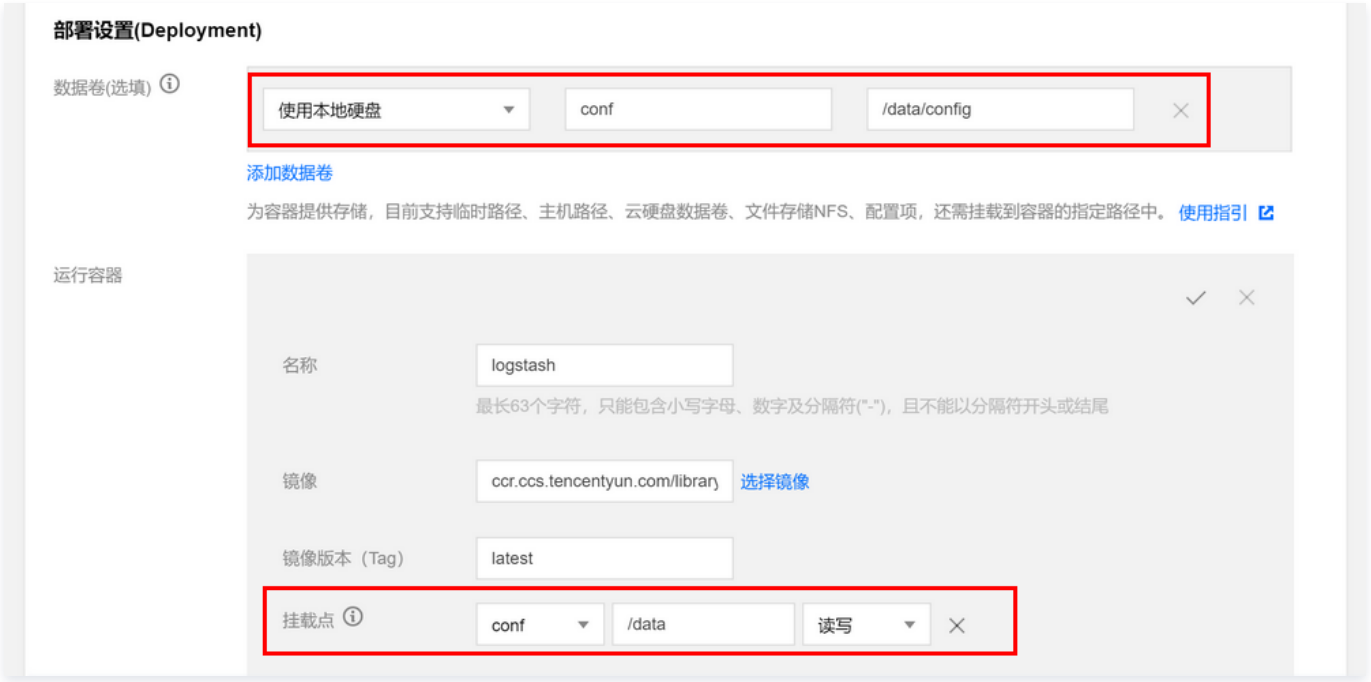


3. Select the Logstash image.
In this example, we use the Logstash image provided by TencentHub Image Repository. Users can also create their own Logstash

image.

4. Create a data volume.
   Create a data volume to store the Logstash configuration files. In this example, a configuration file named logstash.conf is added to the `/data/config` directory on the CVM and mounted to the `/data` directory in Docker, allowing the container to read the

logstash.conf file upon startup.



5. Configure runtime parameters.

6. Configure the service parameters as needed and create the service.

Configuring File Description

File Data Source

```
input {
    file {
        path => "/var/log/nginx/access.log" # file path
```

```
        }
}
filter {
}
output {
  elasticsearch {
    hosts => ["http://172.16.0.89:9200"] # Private network VIP address and port of Elasticsearch cluster
    index => "nginx_access-%{+YYYY.MM.dd}" # Customize the index name, suffixed by date, and generate an
index every day
 }
}
```

For more information on connecting file data sources, please see file input plugin.

Kafka Data Sources

```
input{
      kafka{
        bootstrap_servers => ["172.16.16.22:9092"]
        client_id => "test"
        group_id => "test"
        auto_offset_reset => "latest" #consumption from latest offset
        consumer_threads => 5
        decorate_events => true #This attribute carries information such as the current topic, offset, group,
and partition to the message
        topics => ["test1", "test2"] #array type, multiple topics can be configured
        type => "test" #data source tag field
      }
}

output {
  elasticsearch {
    hosts => ["http://172.16.0.89:9200"] # Private network VIP address and port of Elasticsearch cluster
    index => "test_kafka"
 }
}
```

For more information on connection Kafka data sources, please see Kafka input plugin.

JDBC-connected database data source

```
input {
    jdbc {
      # mysql database address
      jdbc_connection_string => "jdbc:mysql://172.16.32.14:3306/test"
      # Username + Password
      jdbc_user => "root"
      jdbc_password => "Elastic123"
      # Drive the jar package. If you install and deploy logstash by yourself, download this jar (logstash is
not provided by default).
      jdbc_driver_library => "/usr/local/services/logstash-5.6.4/lib/mysql-connector-java-5.1.40.jar"
      # Drive class name
      jdbc_driver_class => "com.mysql.jdbc.Driver"
      jdbc_paging_enabled => "true"
      jdbc_page_size => "50000"
      # Executed SQL file path + name
      #statement_filepath => "test.sql"
      # SQL statements to be executed
      statement => "select * from test_es"
      # Set the listening interval. The meaning of each field (from left to right) is minute, hour, day,
month and year. All are *, which is updated every minute by default
      schedule => "* * * * *"
```

```
      type => "jdbc"
    }
}

output {
    elasticsearch {
        hosts => ["http://172.16.0.30:9200"]
        index => "test_mysql"
        document_id => "%{id}"
    }
}
```

For more information on connecting JDBC data sources, please see J<1>DBC input plugin<1/>.

Using Beats to Access ES Cluster

Beats include multiple single-purpose collectors. These collectors are lightweight and can be deployed and run on servers to collect logs, monitoring data, etc. Compared to Logstash, Beats occupy fewer system resources.
Beats comprises Filebeat for collecting file-type data, Metricbeat for collecting monitoring metrics, Packetbeat for collecting network packet data, and so on. You can also develop your own Beats components based on the official libbeat library according to their specific needs.

Accessing ES cluster from CVM

1. Install and deploy Filebeat.

```
wget https://artifacts.elastic.co/downloads/beats/filebeat/filebeat-5.6.4-linux-x86_64.tar.gz
tar xvf filebeat-5.6.4-linux-x86_64.tar.gz
```

2. Configure the filebeat.yml file.
   Refer to the filebeat.yml configuration example:

```
################### Filebeat Configuration Example #########################

############################# Filebeat ######################################
filebeat:
  # List of prospectors to fetch data.
  prospectors:
    # Each - is a prospector. Below are the prospector specific configurations
    -
      # Paths that should be crawled and fetched. Glob based paths.
      # To fetch all ".log" files from a specific level of subdirectories
      # /var/log/*/*.log can be used.
      # For each file found under this path, a harvester is started.
      # Make sure not file is defined twice as this can lead to unexpected behaviour.
      # Specify the logs to monitor. You can specify specific files or directories
      paths:
        - /var/log/*.log  (This is the default. You can modify it, for example, to /home/hadoop/app.log)
        #- c:\programdata\elasticsearch\logs\*

      # Configure the file encoding for reading files with international characters
      # following the W3C recommendation for HTML5 (http://www.w3.org/TR/encoding).
      # Some sample encodings:
      #   plain, utf-8, utf-16be-bom, utf-16be, utf-16le, big5, gb18030, gbk,
      #    hz-gb-2312, euc-kr, euc-jp, iso-2022-jp, shift-jis, ...
      # Specify the encoding type of the monitored files. Both plain and utf-8 can handle Chinese logs
      #encoding: plain

      # Type of the files. Based on this the way the file is read is decided.
      # The different types cannot be mixed in one prospector
      #
```

```
      # Possible options are:
      # * log: Reads every line of the log file (default)
      # * stdin: Reads the standard in
      # Specify the input type of the file: log (default) or stdin
      input_type: log

      # Exclude lines. A list of regular expressions to match. It drops the lines that are
      # matching any regular expression from the list. The include_lines is called before
      # Exclude lines from the input that match the regular expression list.
      # exclude_lines. By default, no lines are dropped.
      # exclude_lines: ["^DBG"]

      # Include lines. A list of regular expressions to match. It exports the lines that are
      # matching any regular expression from the list. The include_lines is called before
      # exclude_lines. By default, all the lines are exported.
      # Include lines from the input that match the regular expression list (default is to include all
lines). include_lines is executed before exclude_lines
      # include_lines: ["^ERR", "^WARN"]

      # Exclude files. A list of regular expressions to match. Filebeat drops the files that
      # are matching any regular expression from the list. By default, no files are dropped.
      # Ignore files that match the regular expression list
      # exclude_files: [".gz$"]

      # Optional additional fields. These field can be freely picked
      # to add additional information to the crawled log files for filtering
      # Add extra information to each log entry in the output, such as "level:debug", for easier grouping
and statistics later.
      # By default, new fields are created in a specified subdirectory under the fields subdirectory in the
output, such as fields.level
      # This means an additional field will be added in es with the format "fields":{"level":"debug"}"
      #fields:
      #  level: debug
      #  review: 1

      # Set to true to store the additional fields as top level fields instead
      # of under the "fields" sub-dictionary. In case of name conflicts with the
      # fields added by Filebeat itself, the custom fields overwrite the default
      # fields.
      # If this option is set to true, the new fields become top-level fields instead of being placed under
the fields directory.
      # Custom fields defined here will overwrite Filebeat's default fields
      # If set to true, the new fields in es will be formatted as: "level":"debug"
      #fields_under_root: false

      # Ignore files which were modified more then the defined timespan in the past.
      # In case all files on your system must be read you can set this value very large.
      # Time strings like 2h (2 hours), 5m (5 minutes) can be used.
      # You can specify Filebeat to ignore modifications outside a specified time range, such as 2h (two
hours) or 5m (five minutes).
      #ignore_older: 0

      # Close older closes the file handler for which were not modified
      # for longer then close_older
      # Time strings like 2h (2 hours), 5m (5 minutes) can be used.
      # If a file has not been updated within a certain time frame, the monitored file handle is closed.
Default: 1h
      #close_older: 1h

      # Type to be published in the 'type' field. For Elasticsearch output,
      # the type defines the document type these entries should be stored
      # in. Default: log
```

```
      # Set the type field of the document when outputting to Elasticsearch. It can be used to categorize
logs. Default: log
      #document_type: log

      # Scan frequency in seconds.
      # How often these files should be checked for changes. In case it is set
      # to 0s, it is done as often as possible. Default: 10s
      # How frequently Filebeat scans the directories specified by prospectors for file updates (e.g., new
files)
      # If set to 0s, Filebeat will detect updates as quickly as possible (CPU usage will be higher). The
default is 10s
      #scan_frequency: 10s

      # Defines the buffer size every harvester uses when fetching the file
      # The buffer size used by each harvester when monitoring files
      #harvester_buffer_size: 16384

      # Maximum number of bytes a single log event can have
      # All bytes after max_bytes are discarded and not sent. The default is 10MB.
      # This is especially useful for multiline log messages which can get large.
      # Adding a line in the log file counts as one log event. The max_bytes limits the number of bytes
uploaded per log event; excess bytes will be discarded
      #max_bytes: 10485760

      # Mutiline can be used for log messages spanning multiple lines. This is common
      # for Java Stack Traces or C-Line Continuation
      # Suitable for cases where each log entry spans multiple lines, such as various language error stack
traces
      #multiline:

        # The regexp Pattern that has to be matched. The example pattern matches all lines starting with [
        # Pattern matching the first line where multi-line logs begin
        #pattern: ^\[

        # Defines if the pattern set under pattern should be negated or not. Default is false.
        # Whether to transpose the pattern condition. Set to true for no transpose, set to false for
transpose.  [Recommended to set to true]
        #negate: false

        # Match can be set to "after" or "before". It is used to define if lines should be append to a
pattern
        # that was (not) matched before or after or as long as a pattern is not matched based on negate.
        # Note: After is the equivalent to previous and before is the equivalent to to next in Logstash
        # After matching the pattern, whether to merge with the contents before (before) or after (after)
it into one log
        #match: after

        # The maximum number of lines that are combined to one event.
        # In case there are more the max_lines the additional lines are discarded.
        # Default is 500
        # Maximum number of lines to merge (including the line that matches the pattern)
        #max_lines: 500

        # After the defined timeout, an multiline event is sent even if no new pattern was found to start a
new event
        # Default is 5s.
        # After the timeout, even if no new pattern is matched (a new event occurs), the matched log event
will be sent out
        #timeout: 5s

      # Setting tail_files to true means filebeat starts readding new files at the end
      # instead of the beginning. If this is used in combination with log rotation
```

```
      # this can mean that the first entries of a new file are skipped.
      # If set to true, Filebeat monitors new content added from the end of the file, sending each new line
as an event in sequence,
      # instead of resending all content from the beginning of the file
      #tail_files: false

      # Backoff values define how agressively filebeat crawls new files for updates
      # The default values can be used in most cases. Backoff defines how long it is waited
      # to check a file again after EOF is reached. Default is 1s which means the file
      # is checked every second if new lines were added. This leads to a near real time crawling.
      # Every time a new line appears, backoff is reset to the initial value.
      # After Filebeat detects that a file has reached EOF (end of file), how long to wait before checking
if the file has been updated, default is 1s
      #backoff: 1s

      # Max backoff defines what the maximum backoff time is. After having backed off multiple times
      # from checking the files, the waiting time will never exceed max_backoff idenependent of the
      # backoff factor. Having it set to 10s means in the worst case a new line can be added to a log
      # file after having backed off multiple times, it takes a maximum of 10s to read the new line
      # Maximum wait time for Filebeat to detect file updates after reaching EOF, default is 10 seconds
      #max_backoff: 10s

      # The backoff factor defines how fast the algorithm backs off. The bigger the backoff factor,
      # the faster the max_backoff value is reached. If this value is set to 1, no backoff will happen.
      # The backoff value will be multiplied each time with the backoff_factor until max_backoff is reached
      # Definition of the speed to reach max_backoff. The default factor is 2. After reaching max_backoff,
it will wait max_backoff time before each backoff,
      # It will reset to backoff only after the file is updated
      # According to the current default configuration, it checks the file changes every 1s. If the file
does not change after two consecutive checks, the next check interval will change to 10s
      #backoff_factor: 2

      # This option closes a file, as soon as the file name changes.
      # This config option is recommended on windows only. Filebeat keeps the files it's reading open. This
can cause
      # issues when the file is removed, as the file will not be fully removed until also Filebeat closes
      # the reading. Filebeat closes the file handler after ignore_older. During this time no new file with
the
      # same name can be created. Turning this feature on the other hand can lead to loss of data
      # on rotate files. It can happen that after file rotation the beginning of the new
      # file is skipped, as the reading starts at the end. We recommend to leave this option on false
      # but lower the ignore_older value to release files faster.
      # This option closes a file when the file name changes. #This configuration option is recommended
only on Windows
      #force_close_files: false

    # Additional prospector
    #-
      # Configuration to use stdin input
      #input_type: stdin

  # General filebeat configuration options
  #
  # Event count spool threshold - forces network flush if exceeded
  # Size of the spooler. When the number of events in the spooler exceeds this threshold, it will be
cleared and sent out (regardless of whether the timeout has been reached)
  #spool_size: 2048

  # Enable async publisher pipeline in filebeat (Experimental!)
  # Whether to use asynchronous sending mode (experimental feature)
  #publish_async: false
```

```
  # Defines how often the spooler is flushed. After idle_timeout the spooler is
  # Flush even though spool_size is not reached.
  # Spooler timeout. If the timeout is reached, the spooler will be cleared and sent out (regardless of the
capacity threshold)
  #idle_timeout: 5s

  # Name of the registry file. Per default it is put in the current working
  # directory. In case the working directory is changed after when running
  # filebeat again, indexing starts from the beginning again.
  # File recording the location of filebeat processing log files, defaults to the root directory on startup
  #registry_file: .filebeat

  # Full Path to directory with additional prospector configuration files. Each file must end with .yml
  # These config files must have the full filebeat config part inside, but only
  # the prospector part is processed. All global options like spool_size are ignored.
  # The config_dir MUST point to a different directory then where the main filebeat config file is in.
  # If you want to include configuration files from other locations in this configuration file, you can
write them here (need to write the full path), but only the prospector part is processed
  #config_dir:

###############################################################################
############################# Libbeat Config ##################################
# Base config file used by all other beats for using libbeat features

############################# Output ##########################################

# Configure what outputs to use when sending the data collected by the beat.
# Multiple outputs may be used.
output:

  ### Elasticsearch as output
  elasticsearch:                    (This is the default, filebeat collects and then sends to ES. You can
modify it, for example, if you want filebeat to collect and then send to Redis, and then to ES, you can
uncomment this line)
    # Array of hosts to connect to.
    # Scheme and port can be left out and will be set to the default (http and 9200)
    # In case you specify and additional path, the scheme is required: http://localhost:9200/path
    # IPv6 addresses should always be defined as: https://[2001:db8::1]:9200
    hosts: ["localhost:9200"]         (This is the default, filebeat collects and then sends to ES. You can
modify it, for example, if you want filebeat to collect and then send to Redis, and then to ES, you can
uncomment this line)
```

3. Execute filebeat.

```
nohup ./filebeat-5.6.4-linux-x86_64/filebeat 2>&1 >/dev/null &
```

Accessing ES cluster from Docker

Creating Docker cluster

1. Pull the official image of filebeat.

```
 docker pull docker.elastic.co/beats/filebeat:5.6.9
```

2. Customize the <1>*.conf<1/> configuration file based on the data source type and place it in the <3>/usr/share/logstash/pipeline/</3> directory which can be customized.
3. Run filebeat.

```
 docker run docker.elastic.co/beats/filebeat:5.6.9
```

Using Tencent Cloud TKE

Deploying filebeat with Tencent Cloud TKE is similar to deploying logstash, and the official Tencent Cloud filebeat image can be used.

Configuring File Description

Configure the filebeat.yml file as follows:

```
// Input source configuration
filebeat.prospectors:
- input_type: log
    paths:
    - /usr/local/services/testlogs/*.log

// Output to ES
output.elasticsearch:
  # Array of hosts to connect to.
  hosts: ["172.16.0.39:9200"]
```

Using Filebeat to collect TKE container logs

Last updated: 2024-10-25 09:25:50

Background

When deploying services using Tencent Cloud TKE, you can use Filebeat to collect logs from each pod in TKE and write them to a downstream Elasticsearch cluster. Then, you can query and analyze the logs on Kibana. This article explains how to use a Filebeat DaemonSet to collect logs from containers.

Practical Process

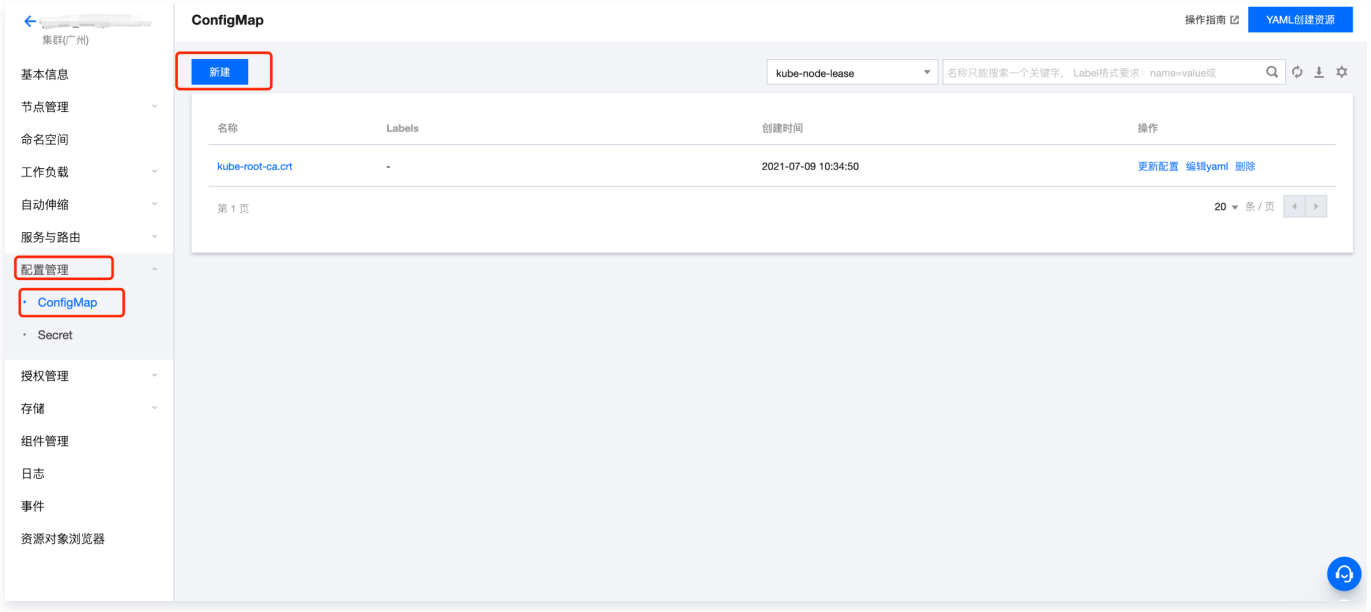
The following example demonstrates the process of using Filebeat to collect logs from Nginx pods running in a TKE container cluster with containerd and writing them to Elasticsearch.

Deploying Nginx DaemonSet

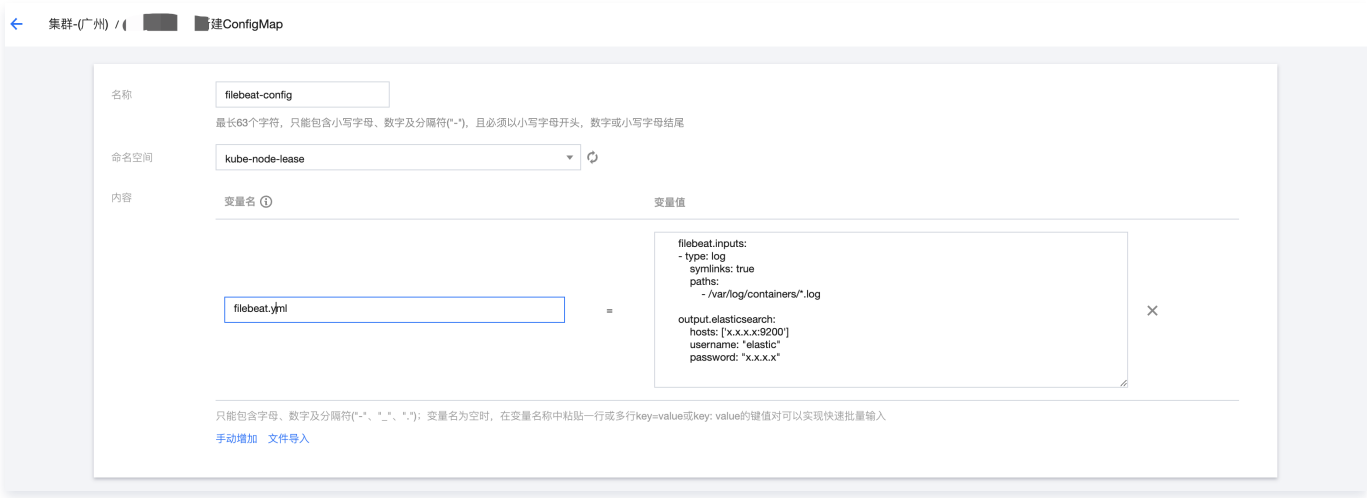
First, use an Nginx image from Docker Hub to deploy a DaemonSet named nginx. For more details, please refer to Creating a Simple Nginx Service.

Deploying Filebeat DaemonSet

1. Creating a ConfigMap
   1.1 Filebeat uses filebeat.yml as its main configuration file. First, create a ConfigMap with the filebeat.yml configuration.
   1.2 Go to **Configuration Management** > **ConfigMap**, click **New.**



   1.3 In the newly created ConfigMap, the variable name is filebeat.yml, and the content is the specific configuration details. Below is a simple filebeat.yml configuration.



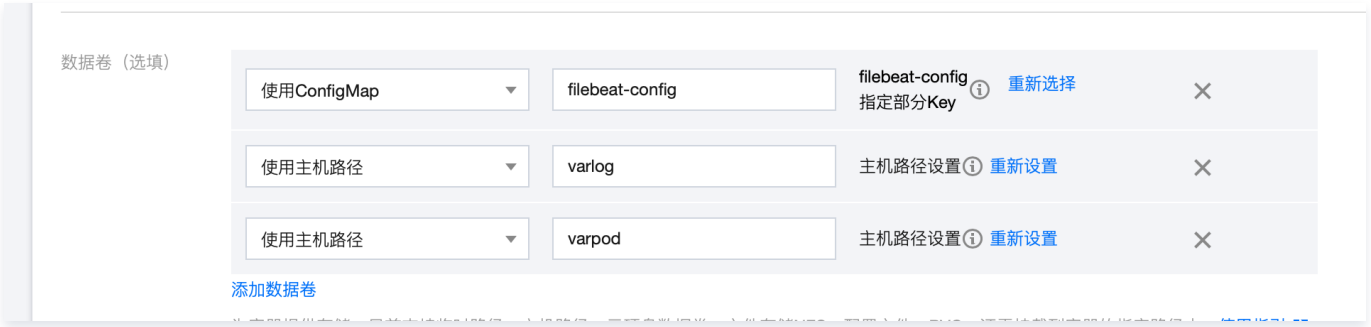

```
filebeat.inputs:
- type: log
    symlinks: true
    paths:
        - /var/log/containers/*.log

output.elasticsearch:
    hosts: ['x.x.x.x:9200']
    username: "elastic"
    password: "x.x.x.x"
```

2. Deploy the Filebeat DaemonSet using a Filebeat image from a public repository.

Search for the required CAM policy as needed, and click to complete policy association.

3. Configure Data Volume



4. Configure Runtime Parameters

5. Configure Mount Point

> **Note**
> Description of Data Volumes and Mount Points:
> 1. Use ConfigMap data volumes to allow the filebeat pod to read the custom-defined filebeat.yml configuration.
> 2. Use host path `/var/log/containers`, so that the filebeat pod can read logs from other pods, since other pods' logs will be printed under the host machine path `/var/log/containers`.
> 3. Since the pod logs under the host path `/var/log/containers` are symbolic links pointing to the logs of each pod in the directory `/var/log/pods`, you need to mount the host path `/var/log/containers` to the filebeat pod. This is why you need to define symlinks: true in filebeat.yml, because by default, filebeat does not read link files.

6. View logs in Kibana
Enter the Kibana corresponding to the ES cluster defined in filebeat.yml to check if the corresponding index is generated and if the NGINX logs can be viewed normally.

By following the above steps, you can check that the NGINX logs are being collected properly. However, the above configuration collects logs from all pods on the host machine. Sometimes you need to collect logs from only certain pods. How to achieve this?

Deploy a filebeat daemonSet through YML configuration that can obtain Pod metadata

In actual business scenarios, it is usually necessary to use Filebeat to collect logs from multiple pods deployed on the same host. Typically, it's also required to obtain metadata of the collected pods, such as namespace, pod name, tags, etc., to facilitate filtering or searching. To get the pod metadata, the Kubernetes API needs to be called. Filebeat has implemented this feature internally, so you can directly use Filebeat's container input and add_kubernetes_metadata processors to achieve this.

1. In the TKE console, click the **YML Create Resource** button and use the following YAML configuration to create a Filebeat DaemonSet.

```
---
        apiVersion: v1
        kind: ConfigMap
        metadata:
            name: filebeat-config
            namespace: default
            labels:
                k8s-app: filebeat
        data:
            filebeat.yml: |-
                filebeat.inputs:
                - type: container
                    paths:
                        - /var/log/containers/*.log
                    processors:
                        - add_kubernetes_metadata:
                                host: ${NODE_NAME}
                                matchers:
                                - logs_path:
                                        logs_path: "/var/log/containers/"

                processors:
                    - add_cloud_metadata:
                    - add_host_metadata:
                output.elasticsearch:
                    hosts: ['${ELASTICSEARCH_HOST:elasticsearch}:${ELASTICSEARCH_PORT:9200}']
                    username: ${ELASTICSEARCH_USERNAME}
                    password: ${ELASTICSEARCH_PASSWORD}
        ---
        apiVersion: apps/v1
        kind: DaemonSet
        metadata:
            name: filebeat
            namespace: default
            labels:
                k8s-app: filebeat
        spec:
            selector:
                matchLabels:
                    k8s-app: filebeat
            template:
                metadata:
                    labels:
                        k8s-app: filebeat
                spec:
                    serviceAccountName: filebeat
                    terminationGracePeriodSeconds: 30
                    containers:
                    - name: filebeat
                        image: ccr.ccs.tencentyun.com/tke-market/filebeat:7.10.1
                        args: [
                            "-c", "/etc/filebeat.yml",
                            "-e",
                        ]
                        env:
                        - name: ELASTICSEARCH_HOST
                            value: x.x.x.x
                        - name: ELASTICSEARCH_PORT
                            value: "9200"
```

```
                    - name: ELASTICSEARCH_USERNAME
                      value: elastic
                    - name: ELASTICSEARCH_PASSWORD
                      value: Elastic123
                    - name: NODE_NAME
                      valueFrom:
                        fieldRef:
                          fieldPath: spec.nodeName
                    securityContext:
                      runAsUser: 0
                      # If using Red Hat OpenShift uncomment this:
                      #privileged: true
                    resources:
                      limits:
                        memory: 200Mi
                      requests:
                        cpu: 100m
                        memory: 100Mi
                    volumeMounts:
                    - name: config
                      mountPath: /etc/filebeat.yml
                      readOnly: true
                      subPath: filebeat.yml
                    - name: varlibdockercontainers
                      mountPath: /var/lib/docker/containers
                      readOnly: true
                    - name: varlog
                      mountPath: /var/log
                      readOnly: true
                    - name: varpods
                      mountPath: /var/log/pods
                      readOnly: true
                volumes:
                - name: config
                  configMap:
                    defaultMode: 0640
                    name: filebeat-config
                - name: varlibdockercontainers
                  hostPath:
                    path: /var/lib/docker/containers
                - name: varlog
                  hostPath:
                    path: /var/log
                - name: varpods
                  hostPath:
                    path: /var/log/pods
        ---
        apiVersion: rbac.authorization.k8s.io/v1
        kind: ClusterRoleBinding
        metadata:
            name: filebeat
        subjects:
        - kind: ServiceAccount
            name: filebeat
            namespace: default
        roleRef:
            kind: ClusterRole
            name: filebeat
            apiGroup: rbac.authorization.k8s.io
        ---
        apiVersion: rbac.authorization.k8s.io/v1
        kind: RoleBinding
```

```
metadata:
    name: filebeat
    namespace: default
subjects:
    - kind: ServiceAccount
        name: filebeat
        namespace: default
roleRef:
    kind: Role
    name: filebeat
    apiGroup: rbac.authorization.k8s.io
---
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
    name: filebeat-kubeadm-config
    namespace: default
subjects:
    - kind: ServiceAccount
        name: filebeat
        namespace: default
roleRef:
    kind: Role
    name: filebeat-kubeadm-config
    apiGroup: rbac.authorization.k8s.io
---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
    name: filebeat
    labels:
        k8s-app: filebeat
rules:
- apiGroups: [""] # "" indicates the core API group
    resources:
    - namespaces
    - pods
    - nodes
    verbs:
    - get
    - watch
    - list
- apiGroups: ["apps"]
    resources:
        - replicasets
    verbs: ["get", "list", "watch"]
---
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
    name: filebeat
    # should be the namespace where filebeat is running
    namespace: default
    labels:
        k8s-app: filebeat
rules:
    - apiGroups:
            - coordination.k8s.io
        resources:
            - leases
        verbs: ["get", "create", "update"]
---
```

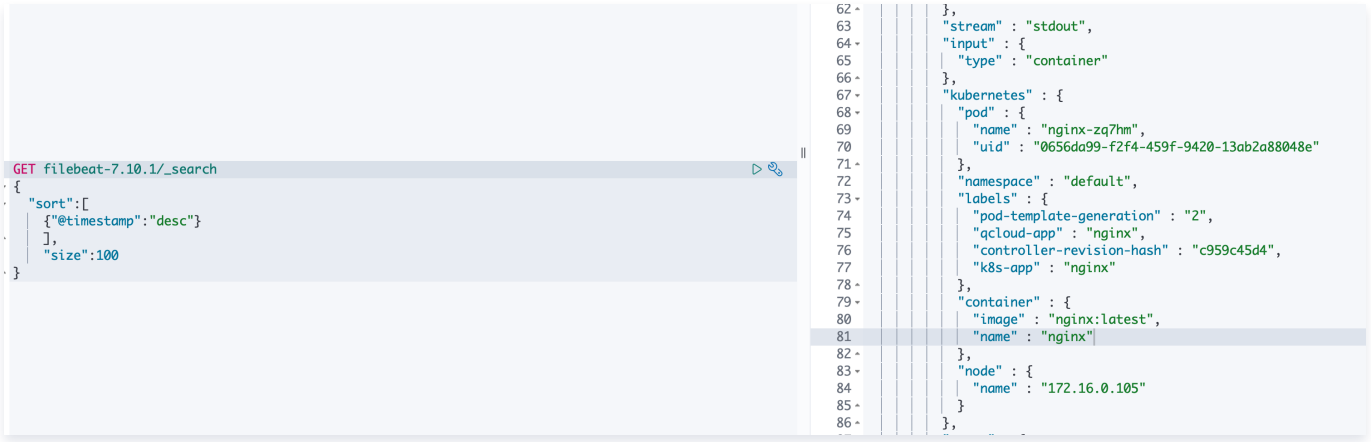
```
        apiVersion: rbac.authorization.k8s.io/v1
        kind: Role
        metadata:
            name: filebeat-kubeadm-config
            namespace: default
            labels:
                k8s-app: filebeat
        rules:
            - apiGroups: [""]
                resources:
                    - configmaps
                resourceNames:
                    - kubeadm-config
                verbs: ["get"]
        ---
        apiVersion: v1
        kind: ServiceAccount
        metadata:
            name: filebeat
            namespace: default
            labels:
                k8s-app: filebeat
---
```

Note

Configuration instructions:

1. The namespace is default, and it can be directly replaced as needed.
2. Created a service account named filebeat and granted it permissions to access pods list, get pod details, and other interfaces. Filebeat will use this account to get the metadata of the pods.
3. Collect logs from the `/var/log/containers/` directory through container input. The container input can collect the stdout and stderr of the containers.

2. View logs in Kibana. You can see that each log contains Kubernetes fields.



3. The above configuration collected logs of all pods on the node where the filebeat pod is located directly through container input. Of course, it also includes the logs of filebeat itself. In real business scenarios, it is usually only necessary to collect the logs of the pods that the business cares about.

- Method one: Filter all collected log events through the Definition processor in filebeat.yml, and filter out the log events that are not of interest (for example, filter out logs of certain namespace and pods not of interest through drop event processor);
- Method two: Through the Autodiscover Definition in the new filebeat.yml configuration file, collect only the logs of the fixed pods through the Definition template. Below is a simple Autodiscover configuration that only collects the logs of the pod named nginx:

```
    filebeat.autodiscover:
                providers:
                    - type: kubernetes
```

```
                        templates:
                            - condition:
                                    equals:
                                        kubernetes.container.name: nginx
                                config:
                                     - type: container
                                       paths:
                                          - /var/log/containers/${data.kubernetes.pod.name}_*.log

    processors:
                - add_cloud_metadata:
                - add_host_metadata:
    output.elasticsearch:
                hosts: ['${ELASTICSEARCH_HOST:elasticsearch}:${ELASTICSEARCH_PORT:9200}']
                username: ${ELASTICSEARCH_USERNAME}
                password: ${ELASTICSEARCH_PASSWORD}
```

4. By modifying the ConfigMap named filebeat-config and redeploying the pod to make the new configuration take effect, you can see that only the logs of the nginx pod are collected in Kibana.

MySQL Data Synchronization Scheme Selection

Last updated: 2024-10-25 09:26:12

When you need to synchronize data from MySQL to Elasticsearch, different synchronization schemes can be selected based on varying application scenarios, data scale, and synchronization delay requirements. This article introduces common synchronization schemes and their applicable scenarios to help you choose the appropriate scheme for data synchronization.

Synchronization Scheme	Basic Principle	Applicable Scenario	Use Limits	Reference
Use Oceanus for Real-time Synchronization	Use Tencent Cloud's stream computing service Oceanus for real-time synchronization of MySQL data to Elasticsearch by parsing binlogs	Scenarios with high real-time requirements	1. MySQL's binlog must be in ROW Mode (enabled by default in TencentDB for MySQL products) 2. The MySQL data table to be synchronized must contain a primary key 3. Modifying the structure of the MySQL table being synchronized is not supported during the synchronization process 4. You need to grant the account used to connect to MySQL the RELOAD and REPLICATION permission	Using Tencent Cloud Oceanus to Real-Time Sync Data from MySQL to Elasticsearch
Logstash enables full synchronization and incremental synchronization	The input-jdbc plugin of Logstash can achieve full data synchronization from MySQL to Elasticsearch by batching the data from MySQL tables and writing them in bulk to ES, thereby achieving full synchronization Additionally, when there is only new data in the MySQL table, a scheduled task can be set in Logstash's configuration to periodically query the newly added data within a recent period for simple incremental synchronization	Scenarios requiring full table data import into ES, and scenarios with low real-time performance requirements	1. For incremental synchronization, the MySQL table needs to have an auto-increment field or a timestamp field, such as UpdateTime 2. Incremental synchronization does not support data deletion 3. The real-time performance of incremental synchronization is relatively poor	Using Tencent Cloud Logstash to synchronize data from MySQL to Elasticsearch

Sync MySQL Data to ES in Real Time using SCS

Last updated: 2024-10-25 09:26:52

This document primarily introduces the practice of collecting data from a MySQL database to the stream computing service Oceanus for analysis, and finally outputting it to the ElasticSearch service. This solution can be used for scenarios like log search. It uses TencentDB for MySQL, SCS, Elasticsearch, Kibana, and VPC.

Building an Environment

Creating Oceanus cluster

In the SCS Console, create a new cluster in **Cluster Management > Create Cluster**, select region, availability zone, VPC, logs, storage, set initial password, etc.

> **Note:**
> - If you haven't used components like VPC, CLS logs, COS before, you need to create them first.
> - When creating a new cluster, ensure that the VPC and subnet selection are the same as the MySQL and ES clusters in the following text; otherwise, you need to connect them manually (e.g., Peering Connection).

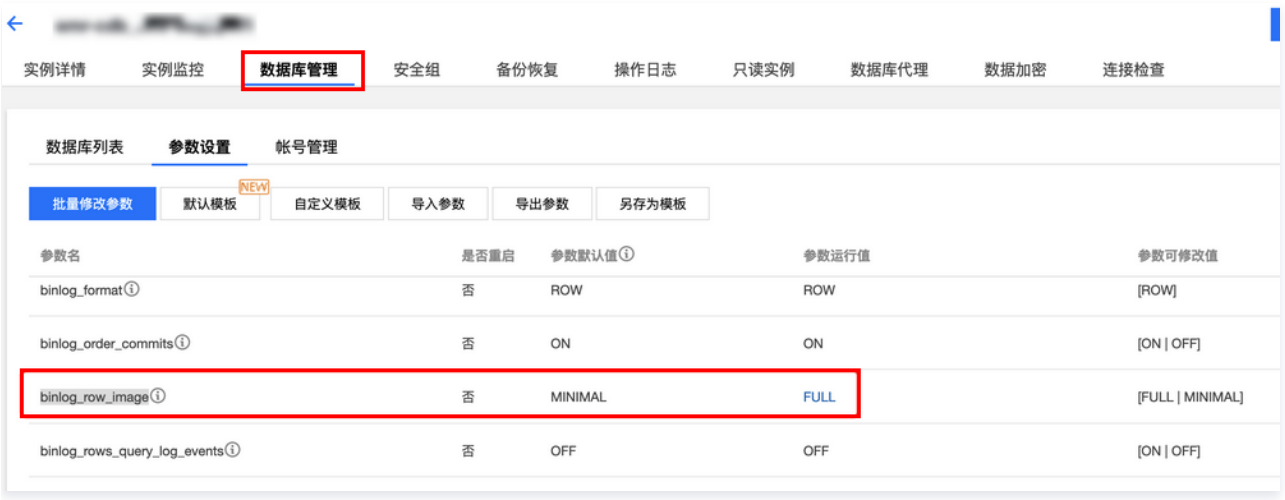
The created cluster is as follows:
Search for the required CAM policy as needed, and click to complete policy association.

Creating a MySQL Cluster

In the TencentDB Console, click **Create** to create a MySQL cluster. Then, in **DMC > Parameter Settings**, modify the following parameters.

```
binlog_row_image=FULL
```

Search for the required CAM policy as needed, and click to complete policy association.



Create a table in MySQL

Execute the following SQL, or use a visual interface to create a database and a table.

```
-- Create a database
create database test;
-- For example, a student grade table
CREATE TABLE cdc_source4es (
  id int(11) NOT NULL AUTO_INCREMENT COMMENT 'Student ID',
  score int(11) NOT NULL COMMENT 'Score',
  PRIMARY KEY (id)
) ENGINE=InnoDB AUTO_INCREMENT=7 DEFAULT CHARSET=utf8 COMMENT='create for student score'
```

and insert a few rows into the table.

```
insert into cdc_source4es values(1, 99);
insert into cdc_source4es values(2, 88);
insert into cdc_source4es values(3, 77);
```

Create an Elasticsearch Cluster

In the ES Console, click **Create New**, and create an ES cluster. It is recommended to select the same region, availability zone, and network as Oceanus. This example uses an Elasticsearch 7.5.1 version cluster.

Once the cluster is created, cluster information can be viewed through Kibana. Execute the following command on the Dev Tools panel.

> **Note:**
> In ES, there is no need to create a table-like entity in advance.

```
# View Cluster Nodes
GET _cat/nodes
```

```
# If node information is returned, it is normal
172.28.1.1 43 99 1 0.06 0.06 0.12 dilm - 1627027760001130832
172.28.1.2  65 99 3 0.03 0.12 0.13 dilm - 1627027760001130732
172.28.1.3 29 99 3 0.08 0.08 0.12 dilm * 1627027760001130632
```

Job Creation

Creating SQL Job

In the SCS Console, under **Job Management > Create a new job**, create a **SQL job**. Choose to create a new job in the newly created cluster. Then, in the job's **Development & Debugging > Job Parameters**, add the necessary connectors, such as the mysql-cdc connector, elasticsearch6/7 connector.

> **Note**
> The version of the ES connector should match the version of the purchased ES component.

Search for the required CAM policy as needed, and click to complete policy association.

Creating Source End

Select MySQL as the data source and continuously update subsequent data to ES.

```
-- mysql-cdc connector
CREATE TABLE mysql_source (
    id int,
    score int,
    PRIMARY KEY (id) NOT ENFORCED -- If the database table to be synchronized has a primary key defined, it
needs to be defined here as well
) WITH (
    'connector' = 'mysql-cdc',       -- Must be 'mysql-cdc'
    'hostname' = '172.28.28.213',    -- Database IP
    'port' = '3306',                 -- Database access port
    'username' = 'youruser',         -- Database access username (requires SHOW DATABASES, REPLICATION SLAVE,
REPLICATION CLIENT, SELECT, RELOAD permissions)
    'password' = 'yourpassword',     -- Database access password
    'database-name' = 'test',    -- The database that needs to be synchronized
    'table-name' = 'cdc_source4es'       -- The table that needs to be synchronized
);
```

Create Sink endpoint

Sink does not need to be initialized in the ES cluster in advance; data can be written directly.

```
-- Note! If you have enabled the Elasticsearch username password authentication feature, you can currently
only use Flink 1.10's old syntax. If no authentication is required, you can use Flink 1.11's new syntax.
-- See https://ci.apache.org/projects/flink/flink-docs-release-1.10/dev/table/connect.html#elasticsearch-
connector

CREATE TABLE es_old_sink (
    id INT,
    score INT
) WITH (
    'connector.type' = 'elasticsearch', -- Output to Elasticsearch
    'connector.version' = '7',          -- Specify the version of Elasticsearch, such as '6', '7'. Make sure
it matches the selected built-in Connector version
    'connector.hosts' = 'http://172.28.1.175:9200',  -- Connection address of Elasticsearch
    'connector.index' = 'connector-test-index',        -- Name of the Elasticsearch index
    'connector.document-type' = '_doc',  -- Type of the Elasticsearch document
    'connector.username' = 'elastic',  -- Optional: Elasticsearch username
    'connector.password' = 'yourpassword',  -- Optional: Elasticsearch password
```

```
    'update-mode' = 'upsert',            -- Optional 'append' mode without primary key or 'upsert' mode with
primary key
    'connector.key-delimiter' = '$',     -- Optional parameter, the delimiter for composite primary keys
(defaults to '_', e.g., key1_key2_key3)
    'connector.key-null-literal' = 'n/a',  -- The substitute string when the primary key is null, defaults to
'null'
    'connector.failure-handler' = 'retry-rejected',   -- Optional error handling. Options are 'fail' (throw
exception), 'ignore' (ignore any error), 'retry-rejected' (retry)

    'connector.flush-on-checkpoint' = 'true',   -- Optional parameter, disallows batch writes (flush) during
snapshots, defaults to true
    'connector.bulk-flush.max-actions' = '42',  -- Optional parameter, maximum number of actions per batch
    'connector.bulk-flush.max-size' = '42 mb',  -- Optional parameter, maximum accumulated size per batch
(supports only mb)
    'connector.bulk-flush.interval' = '60000',  -- Optional parameter, interval for batch writes (ms)
    'connector.connection-max-retry-timeout' = '1000',      -- Maximum timeout for each request (ms)
    --'connector.connection-path-prefix' = '/v1'           -- Optional field, the path prefix appended to each
request

    'format.type' = 'json'        -- Output data format, currently supports only 'json'
);
```

Operator operations

In the syntax below, only simple data insertion is performed, without complex calculations.

```
-- Computing with Flink SQL
INSERT INTO es_old_sink select id, score  from mysql_source;
```

Validation summary

Query in Kibana's Dev Tools to check if the data has been successfully inserted into ES.

```
# Query all data under this index
GET connector-test-index/_search
```

Self-built/Other Cloud Vendors Cluster Migration Analysis and Operation Guide for Online Fusion Migration of Self-Built ES Clusters

Last updated: 2024-10-25 09:27:13

As the stability of Tencent Cloud ES clusters continues to improve, the product experience is becoming increasingly better. More and more external customers are expressing their desire to migrate their self-built ES clusters to Tencent Cloud. This article will introduce a unique migration plan in the industry offered by Tencent Cloud ES, which ensures service continuity without downtime: the Online Fusion Migration Plan. To date, this plan has successfully migrated hundreds of self-built ES clusters to the cloud. Drawing from our valuable experience accumulated during customer migrations, we will detail the basic principles, core advantages, migration steps, and precautions of the Online Fusion Migration technical solution.

Why Online Fusion Migration is Needed

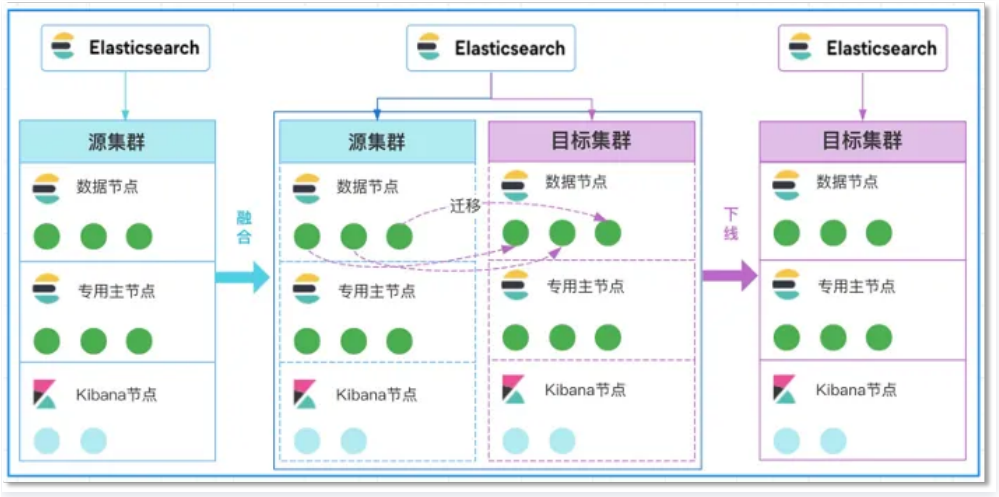
To answer this question, first, let's look at the commonly used migration solutions in the industry, which are Logstash, Reindex, Snapshot Backup & Restore, and Cross-Cluster Replication (CCR), as shown in the following diagram.

迁移方案	Logstash	Reindex	快照备份与恢复	跨集群复制CCR
迁移原理	通过Logstash管道连通源集群和目标集群，从源集群中读取数据写入到目标集群中。	类似于Logstash迁移，批量读取源集群数据写入到目标集群中。	通过对源集群索引数据打快照，然后在目标集群中进行恢复的方式。	采用类似于MySQL的主从复制机制来实现数据迁移
迁移类型	离线	离线	离线	半在线
是否需要停服	是	是	是	短暂停服
适用场景	集群规模适中，索引个数较少的场景	集群规模较小，且对迁移速度要求不高的场景	集群规模适中，索引个数较多的场景	集群规模中等，对停服时间有严格要求的场景
痛点	需要先在目标集群中创建好索引模版，且需要逐个校验数据一致性，有丢数据风险	离线迁移，业务停服，且只能迁移小规模集群	源集群需要安装COS插件，且需要滚动重启集群	1、有版本限制，低版本不支持 2、切割时需停服且全量重启集群 3、写入压力大时，有性能问题

From the diagram, we can see that all four migration solutions share a common characteristic: they require the service to be temporarily shut down, meaning they are all offline migrations. The biggest pain point of offline migration is that business operations need to halt during the migration, and the migration process is very cumbersome with a huge workload. For some core business clusters, customers can almost never accept any downtime. Therefore, those responsible for migrations urgently need a smooth, business-transparent, and highly available migration technology solution. This is the problem that Tencent Cloud ES Online Fusion Migration technology solution aims to address.

Principle of the Online Fusion Migration Solution

The greatest advantages of the Online Fusion Migration solution are encapsulated in the word 'online'. Unlike offline migrations, it can truly achieve zero downtime and transparency for businesses; and the essence of its principle is 'fusion', by combining the self-built cluster and the cloud-based cluster, which are two originally independent clusters, into one larger cluster, and leveraging ES cluster's native shard allocation and migration features to complete the data migration. The schematic diagram of fusion migration is shown as follows.



From the migration process schematic diagram above, we can see that the entire migration mainly consists of three steps: fusion, migration, and decommissioning.

- **Integration**: First, we need to apply for an empty cluster of the same scale as the self-built ES cluster on the Tencent Cloud ES console, which is the target cluster in the above diagram. Then, fully restart the cloud cluster and add it to the self-built ES cluster, integrating the two clusters into a larger one.
- **Migration**: After integration, use the ES cluster cluster/settings to set the exclude attribute for shard migration. Once the following API is executed, the ES cluster will automatically migrate the shards from the self-built cluster nodes to the cloud nodes, completing the shard relocation and data migration.

```
PUT _cluster/settings
{
  "persistent": {
      "cluster.routing.allocation.exclude_name": "{source_node_names}"
      }
}
```

- **Decommissioning**: After all the shards on the self-built cluster nodes are migrated, shut down all the self-built cluster nodes, thereby completing the migration process to the cloud. The following API can be used to check if the number of shards on the self-built nodes is zero.

```
GET /_cat/allocation?v=true
```

cat apis

allocation

当shards数量为0时，表示该节点上已经没有任何分片分配。

shards	disk.indices	disk.used	disk.avail	disk.total	disk.percent
11	26.8mb	1.2gb	18.3gb	19.5gb	6
11	8.8mb	1.1gb	18.3gb	19.5gb	6
10	20mb	1.1gb	18.3gb	19.5gb	6

Advantages of online fusion migration solution

Smooth migration, business continuity, no service interruption

Compared to offline migration solutions such as Logstash, snapshot backup, and cross-cluster replication, online fusion migration truly achieves smooth migration and uninterrupted business.

No need to restart the cluster, no need for data consistency check

Online fusion migration incorporates the self-built cluster into the cloud ES cluster through a full restart, thus avoiding any invasive operations on the customer's self-built cluster. This differs from snapshot migration, which requires pre-installing the COS plugin and

restarting the cluster. It also does not require consistency checks of document counts before and after migration like Logstash migration.

The version requirements are relatively low, only requiring that the cloud cluster version is not lower than the self-built cluster

The version requirements for online fusion migration only stipulate that the cloud cluster version must not be lower than the self-built cluster version. Currently, the cloud offers multiple versions including 5.6.4, 6.4.3, 6.8.2, 7.10.1, and 7.14.2. There is always a version that can meet customer needs.

Greatly reduces labor costs and operational costs

The reason it can significantly save labor costs and operational costs is mainly due to the numerous aspects involved in the offline migration solution. For example, it requires prior communication with business colleagues about acceptable service suspension time and duration. Most business off-peak periods are early in the morning, making offline migration timing requirements very stringent. Additionally, it requires data consistency checks, which are very time-consuming and labor-intensive. In contrast, the online fusion migration solution is smooth and transparent, with no service downtime. Therefore, the entire migration process doesn't require any participation from the business side, has no strict time requirements, and doesn't need data consistency checks. All these factors can greatly reduce labor and operational costs.

Restrictions of Online Fusion Migration

Although online integrated migration can completely solve the pain points of business downtime and significantly improve migration efficiency, not all self-built clusters qualify for integrated migration. Based on our extensive testing and actual migration experiences, we have summarized the following limitations and required conditions.

Self-built ES cluster versions must not be higher than the cloud-based cluster versions

For example, if the highest version currently offered by the cloud-based ES is 7.14.2, then if the version of the self-built cluster is higher than 7.14.2, it cannot be migrated using the online integrated migration solution. The main reason is that nodes of lower versions in cloud-based clusters cannot join higher version self-built clusters. Additionally, for the version numbers, it's better if the major versions of both clusters are consistent; for instance, if the self-built version is 6.4.6, choose 6.8.2 in the cloud-based environment as a priority. If the self-built version is 7.5.2, choose 7.10.1 in the cloud-based environment as a priority.

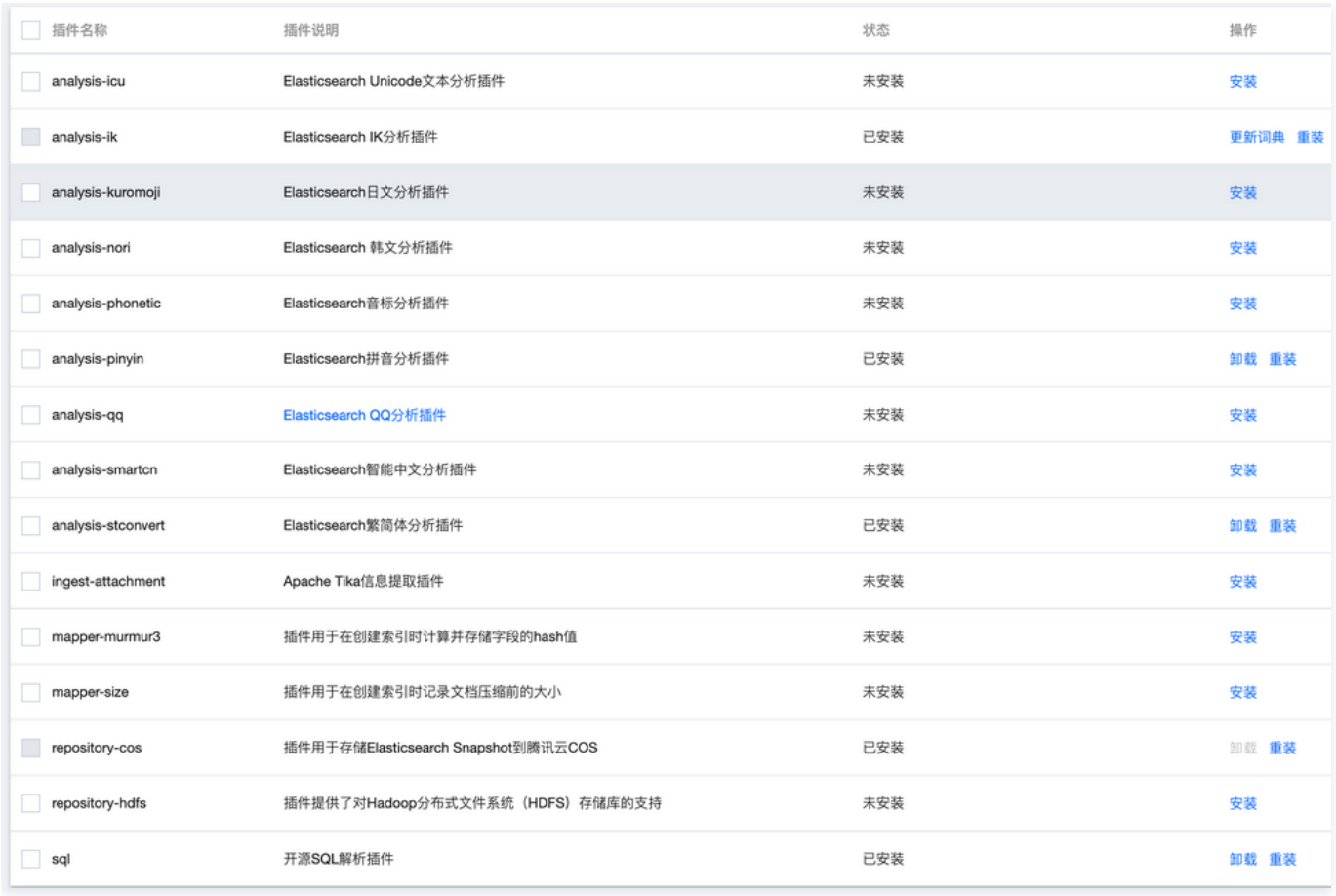
Self-built ES clusters must not have plugins not present in the cloud-based clusters

Before migration, you need to execute the following API on self-built cluster nodes to check which plugins are installed on the self-built cluster.

```
curl  http://127.0.0.1:9200/_cat/plugins
```

If there are plugins installed on the self-built cluster that do not exist in the cloud-based cluster, it will cause the cluster status to become abnormal due to shard allocation failures after the nodes are integrated into the cloud-based ES cluster. The following figure 4

shows the list of plugins supported by the cloud-based ES cluster.



插件名称	插件说明	状态	操作
analysis-icu	Elasticsearch Unicode文本分析插件	未安装	安装
analysis-ik	Elasticsearch IK分析插件	已安装	更新词典 重装
analysis-kuromoji	Elasticsearch日文分析插件	未安装	安装
analysis-nori	Elasticsearch 韩文分析插件	未安装	安装
analysis-phonetic	Elasticsearch音标分析插件	未安装	安装
analysis-pinyin	Elasticsearch拼音分析插件	已安装	卸载 重装
analysis-qq	Elasticsearch QQ分析插件	未安装	安装
analysis-smartcn	Elasticsearch智能中文分析插件	未安装	安装
analysis-stconvert	Elasticsearch繁简体分析插件	已安装	卸载 重装
ingest-attachment	Apache Tika信息提取插件	未安装	安装
mapper-murmur3	插件用于在创建索引时计算并存储字段的hash值	未安装	安装
mapper-size	插件用于在创建索引时记录文档压缩前的大小	未安装	安装
repository-cos	插件用于存储Elasticsearch Snapshot到腾讯云COS	已安装	卸载 重装
repository-hdfs	插件提供了对Hadoop分布式文件系统（HDFS）存储库的支持	未安装	安装
sql	开源SQL解析插件	已安装	卸载 重装

Ensure network interconnection between the self-built ES cluster and the cloud-based cluster

The mentioned network interconnection requires that the cloud-based nodes can directly access the self-built cluster nodes, and similarly, the self-built cluster nodes can directly access the cloud-based cluster nodes. There should be no proxy in between. Access via proxy is considered one-way network communication, not two-way network communication. Below are the two common network environments and network communication solutions for self-built clusters:

1. Tencent Cloud CVM Self-built ES Cluster Migration to Tencent Cloud ES:
   Since Tencent Cloud ES Cluster uses cross-tenant ENI to directly connect ES Cluster VPC and customer VPC network environments, for Tencent Cloud CVM self-built ES Clusters, you can achieve two-way network interconnection by purchasing a cluster in the cloud ES console using the VPC where the self-built cluster is located. If the self-built cluster and Tencent Cloud ES Cluster are not in the same region, you can use Cloud CCN or Peering Connection solutions to connect the networks.
2. IDC Self-built ES Cluster Migration to Tencent Cloud ES:
   If the customer's self-built ES cluster is in their IDC data center or other cloud provider CVM, you can connect the networks via dedicated lines. Once the networks of both clusters are connected, you can verify via telnet from the cloud-based cluster nodes to the self-built IP 9300.

Ensure that the self-built ES cluster has not enabled security

Usually, the customer's self-built clusters are open-source versions. If it is the basic version and security is enabled, you need to disable security before initiating integration. Additionally, platinum version clusters are not supported for integration. If you want to use the platinum version after migrating to the cloud, you can upgrade to the platinum version after the migration is complete.

The cloud-based cluster name must be consistent with the self-built cluster name

By default, the Tencent Cloud ES Cluster generates a fixed-format, fixed-length string as the cluster name, which is the cluster.name in the elasticsearch.yml configuration file, such as cluster.name: "es-cohesszwr" (this cluster ID configuration name is fictional). However, if you need to integrate with a self-built cluster, you must ensure that the cluster name of both clusters is the same. Since the cloud-based cluster needs to join the self-built cluster, you must ensure the cloud-based cluster's name is the same as the self-built cluster's name. The custom cloud-based cluster configuration name needs to enable allowlist support. After enabling the

allowlist, you will see the entry fields as shown in the figure below. If the cluster is created without setting a custom cluster name, you can contact Tencent Cloud ES migration staff for backend processing.

Basic steps for online integrated migration

The process of online integrated migration requires close cooperation between the customer's operations team and the Tencent Cloud ES migration team. Since ES optimized the Master node selection logic in version 7.0, versions below 7.0 and above 7.0 require different integration strategies. Below are the basic steps and differences for migrating different versions.

Migration for versions below 7.0

Migration for versions below 7.0 refers to when both the customer's self-built ES cluster version and the Tencent Cloud ES cluster version are below 7.0, for example, the customer's version is 6.4.5 and the cloud-based version is 6.8.2.

1. Shard locking (customer-side operation)
   Before migration, execute the following API on the self-built cluster to prioritize shard locking on self-built cluster nodes. This prevents shards from automatically drifting to cloud-based nodes after integration, which could cause unknown risks.

```
curl -H "Content-Type: application/json" -XPUT _cluster/settings -d '{
     "cluster.routing.allocation.include._name" : "List of self-built cluster node names"
}'
```

2. Cluster integration (Tencent Cloud side operation)
   At this step, the Tencent Cloud ES migration staff will call the backend interface to add the self-built ES cluster nodes to the cloud-based nodes and then perform a full restart of the cloud-based cluster. After the cloud-based cluster nodes come back online, they will automatically join the self-built cluster. This successfully integrates both clusters into one large cluster.

```
curl localhost:5100/cluster/update -d '{
"cluster_name": "es-cohesszwr",
"operator": "wr",
"es_config": {
    "discovery.zen.ping.unicast.hosts": "
    [\"10.0.0.xx:9300\",\"10.0.0.xx:9300\",\"10.0.0.xx:9300\",
    \"10.0.0.xx:9300\", \"10.0.0.xx:9300\",, \"10.0.0.xx:9300\"]"
  },
  "restart_type": "full_cluster_restart"
}'
```

3. Data Migration (Tencent Cloud-side Operation)
   Once the cluster integration is successful and no anomalies are found, the formal migration can be initiated by executing the following API.

```
curl -H "Content-Type: application/json" -XPUT _cluster/settings -d '{
"cluster.routing.allocation.include._name" : "Cloud Cluster Node Name List",
"cluster.routing.allocation.exclude._name" : "Self-built Cluster Node Name List"
}'
```

   The purpose of this API is to evict shards from self-built cluster nodes to cloud nodes. During the intermediate state of integration, shard migration will consume some performance. Therefore, for stability considerations, the concurrency and speed of the migration can be reduced through the following API.

```
PUT _cluster/settings
{ "persistent": {
        "cluster.routing.allocation.node_concurrent_recoveries": 2,
        "indices.recovery.max_bytes_per_sec": "40mb/s" }
}
```

In the intermediate state of migration, the business side can change the client configuration to connect to the Tencent Cloud ES using the access VIP.

4. Cluster Separation (Customer-side Operation)
   After confirming that all data has been migrated to the cloud nodes, the customer needs to shutdown and offline the self-built cluster nodes and separate the two integrated clusters. Before taking the self-built nodes offline, modify the self-built cluster configuration name cluster.name to another value, such as adding -local after the original cluster configuration name, and check if there are any scripts for automatic ES process pull-up. If there are, remove them altogether. Additionally, the self-built nodes need to be taken offline one by one, with the elected master node being the last to go offline. This approach minimizes the number of master elections.
5. Configuration Recovery (Tencent Cloud-side Operation)
   After confirming with Tencent Cloud ES migration colleagues that the self-built cluster and cloud cluster are completely separated, reconfirm with the customer-side business to ensure there are no anomalies. If no anomalies are found, restore the cloud configuration to the state before integration by calling the following API.

```
curl localhost:5100/cluster/update -d '{
"cluster_name": "es-cohesszwr",
"operator": "wr",
"es_config": { "discovery.zen.ping.unicast.hosts": "null"},
"restart_type": "no_restart"
}'
```

> **Note**
> Here, the discovery.seed_hosts is set to null because the backend interface will automatically assign the initial node IPs to the configuration. Additionally, restart_type must be set to no_restart for updating without a restart, otherwise, it will affect business stability.

Version 7.0 and above Migration

Migration for version 7.0 and above refers to both the customer-built ES cluster version and the Tencent Cloud ES cluster version being 7.0 or above, or the cloud-based cluster being version 7.0 and above. For example, if the customer's version is 6.8.3 and the cloud version is 7.5.1, or if the self-built version is 7.5.2 and the cloud version is 7.10.1.

1. Shard Locking (Client-Side Operation)
   This step is the same as the operation for versions below 7.0. For specific operation details, refer to the above steps.
2. Cluster Integration (Tencent Cloud-Side Operation)
   The biggest difference in this step from versions below 7.0 is that after adding the self-built cluster node list to the cloud and fully restarting the cloud-based cluster, the cloud-based cluster cannot directly join the self-built cluster. This is mainly related to the master selection algorithm for versions 7.0 and above. Therefore, to achieve normal integration after a full restart of the cloud-based cluster, you need to execute the following commands step by step on the cloud-based cluster nodes to erase the cloud-based cluster's metadata. The detailed principle can be found in the official documentation provided for the elasticsearch-node command.
   First, execute the API for adding self-built node IP configuration:

```
curl localhost:5100/cluster/update -d '{
  "cluster_name": "es-cohesszwr",
  "operator": "wr",
  "es_config": {
  "discovery.seed_hosts": "
      [\"10.0.0.xx:9300\",\"10.0.0.xx:9300\",\"10.0.0.xx:9300\",
       \"10.0.0.xx:9300\", \"10.0.0.xx:9300\",, \"10.0.0.xx:9300\"]"
  },
  "restart_type": "full_cluster_restart"
```

```
}'
```

> **Note**
> In the above API, the key for the es_config field is discovery.seed_hosts, no longer discovery.zen.ping.unicast.hosts.

Then, execute the following command on each node of the cloud-based cluster via a script in batch mode to erase the cluster metadata. After that, you will see that the cloud-based cluster nodes can successfully join the self-built cluster.

```
ps -ef | grep java | grep c_log | awk '{print $2}' |xargs kill -9
cd /data1/containers/*/es/
./bin/elasticsearch-node detach-cluster (select y)
```

3. Data Migration (Tencent Cloud-Side Operation)
   This step is the same as the operation for versions below 7.0. The main purpose is to smoothly migrate the shards from the self-built cluster nodes to the cloud nodes. For specific operation details, refer to the above steps.
4. Cluster Separation (Client-Side Operation)
   This step is the same as the operation for versions below 7.0. For specific operation details, refer to the above steps.
5. Configuration Recovery (Tencent Cloud operation)
   The operation for configuration recovery is similar to versions below 7.0. The only difference lies in the key of the es_config parameter, which should be discovery.seed_hosts. Updates are performed using the no_restart method.

```
curl localhost:5100/cluster/update -d '{
     "cluster_name": "es-cohesszwr",
     "operator": "wr",
     "es_config": { "discovery.seed_hosts": "null"},
     "restart_type": "no_restart"
}'
```

Frequently Asked Questions about Online Fusion Migration

Do not decommission self-built cluster nodes simultaneously. Each node should be shut down and decommissioned individually

We conducted a series of tests for versions above 7.0. When the cluster is in a fusion state and all self-built cluster nodes are decommissioned simultaneously, the cloud cluster immediately enters an ownerless state, rendering the cluster unavailable and causing numerous errors. This is mainly because once the cluster detects that half of its nodes (with the node.master role) have left the cluster simultaneously, it no longer meets the basic conditions to elect a Master Node, causing the cluster to remain in an ownerless state for a prolonged period.
To avoid the risk of cluster unavailability during cluster separation, do not decommission self-built nodes simultaneously. Instead, decommission them one at a time, ensuring that the elected Master Node on the self-built nodes is the last to be decommissioned. Otherwise, multiple master elections may occur, and there is a chance that the master role could be cut to self-built nodes multiple times.

Secondary fusion of the same cluster is prohibited, as it poses a risk of index loss

The online fusion migration plan requires that all self-built cluster indexes be migrated to the cloud after cluster fusion. It is strongly advised against migrating only a portion of indexes or migrating in batches, such as migrating some business indexes after the first fusion, separating the cluster, and then performing a secondary fusion and migrating the remaining indexes over time.
Based on our test results, once data migration is complete and the two clusters are separated, the self-built ES cluster will encounter a Red status. This is primarily because the indexes have been migrated to the cloud, leaving only the index metadata on the self-built cluster, hence the indexes cannot be allocated, causing the Red status after cluster separation.
In such cases, customers typically choose to delete these indexes to restore the cluster to Green. If the two clusters are merged again to migrate the remaining indexes, since the cloud cluster nodes are added to the self-built cluster through a full restart, the merged cluster will rely on the metadata of the elected Master in the self-built cluster. Consequently, once the fusion is successful, the self-built cluster’s metadata is synced to the cloud cluster nodes, directly deleting the indexes previously migrated during the first fusion, resulting in total data loss of previously migrated indexes. The fundamental reason for this risk is that we deleted the Red indexes on the self-built cluster after the first separation, and these deleted indexes are recorded in the self-built cluster's metadata in a place

called the Index Tombstone. These metadata will be synchronized to the cloud cluster after fusion. For more details, refer to the official documentation Index tombstones.
Although we strongly advise against or even prohibit secondary fusion of the same cluster, if secondary fusion is indeed necessary to address migration issues at different stages of business, the following solution can prevent data loss during secondary fusion.
Secondary fusion solution to avoid data loss: Migration index Reopen solution:

1. **Step 1:** For the first fusion, after cluster separation, do not delete the Red indexes on the self-built cluster; instead, execute the close operation on these indexes, i.e., Close Index.

```
POST {index_name}/_close
```

2. **Step 2:** For the second fusion, after successful fusion, execute the open operation on the previously closed indexes, i.e., Open Index.

```
POST {index_name}/_open
```

Based on multiple tests, the Index Migration Reopen Solution can resolve data loss issues in secondary fusion scenarios.

Before fusion, ensure that the client has disabled the sniffing feature

The sniffing feature is enabled to allow the client-side to automatically discover dynamic changes in the cluster-side ES nodes. The cloud ES connects the customer's VPC and the cloud cluster's VPC network through PrivateLink. If the client switches its access configuration to the cloud after fusion while keeping the sniffing feature enabled, it can result in different IP accesses for discovered nodes, leading to a large number of timeout requests on the client-side, affecting business usage. The client's error message may be as follows:

```
NoNodeAvailableException[None of the configured nodes are available: [{#transport#-1}
{VTss4h8SRsCpP6EW5PoLxxxrQ}{192.168.106.xxx}{192.168.106.xxx:9300}] ]at
org.elasticsearch.client.transport.TransportClientNodesService.ensureNodesAreAvailable(TransportClientNodesSe
rvice.java:347) at
org.elasticsearch.client.transport.TransportClientNodesService.execute(TransportClientNodesService.java:245)
at org.elasticsearch.client.transport.TransportProxyClient.execute(TransportProxyClient.java:59) ......
```

Therefore, to ensure the integration migration solution remains transparent and smooth for business, check if the client has the sniffing feature enabled before fusion; if so, it needs to be disabled. For information on how different language clients can access the cloud ES cluster, refer to the Tencent Cloud ES official documentation Access a Cluster via Client.

Use Case Construction
Building a Log Analysis System

Last updated: 2024-10-25 09:27:53

An instance provided by Tencent Cloud ES consists of an ES cluster and a Kibana console. The ES cluster can be accessed via the VPC VIP address and port within your VPC, and the Kibana console provides a public network address for browser access. Currently, you can only connect your data source to the ES cluster by yourself.
The following describes how to import your logs into ES and access Kibana from a browser to perform query and analysis, using the most typical log analysis architectures Filebeat + Elasticsearch + Kibana and Logstash + Elasticsearch + Kibana as examples.

Filebeat + Elasticsearch + Kibana

Deploying Filebeat

1. Download the Filebeat package and decompress it

> **Note**
> The Filebeat version should be compatible with the ES version.

```
wget https://artifacts.elastic.co/downloads/beats/filebeat/filebeat-6.4.3-linux-x86_64.tar.gz
tar xvf filebeat-6.4.3-linux-x86_64.tar.gz
```

2. Configure Filebeat

This example uses Nginx logs as the input source, with the output configured to the intranet VIP address and port of the ES cluster. If you are using a platinum cluster, you need to add username and password authentication in the output.
Enter the filebeat-6.4.3-linux-x86_64 directory and modify the filebeat.yml configuration file as follows:

```
filebeat.inputs:
- type: log
    enabled: true
    paths:
        - /var/log/nginx/access.log
output.elasticsearch:
    hosts: ["10.0.130.91:9200"]  # Private VIP address and port of the ES cluster
    protocol: "http" # ES communication mode, http or https
    username: "elastic" # username
    password: "test" # password
```

3. Execute Filebeat

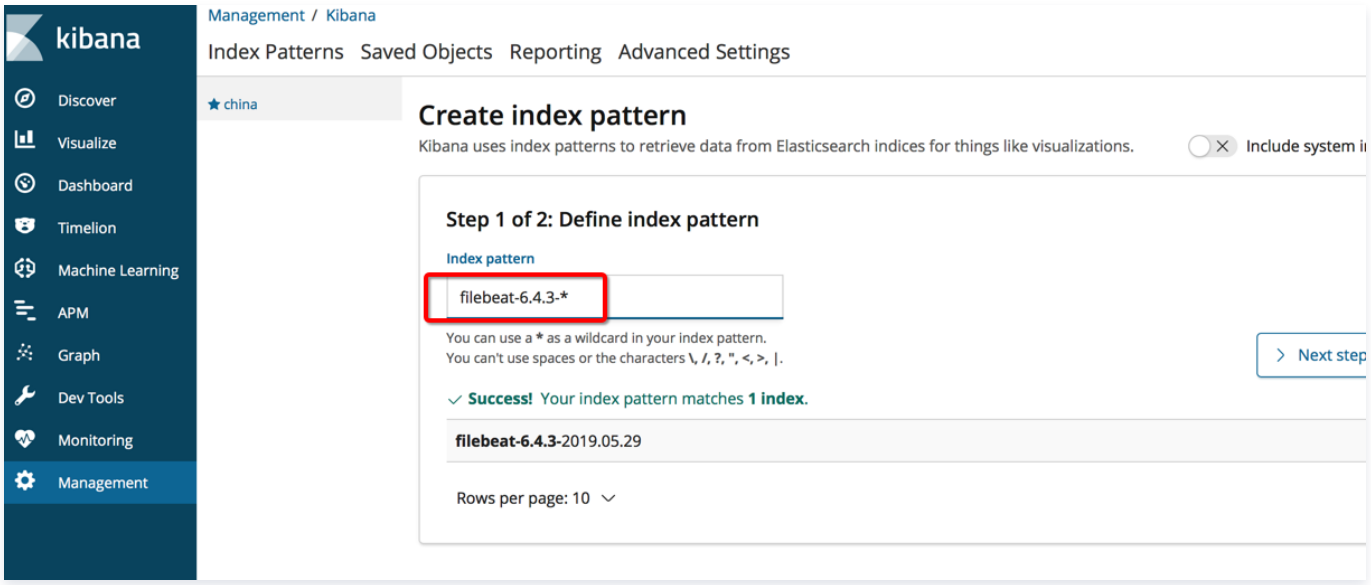
In the filebeat-6.4.3-linux-x86_64 directory, execute:

```
nohup ./filebeat -c filebeat.yml 2>&1 >/dev/null &
```

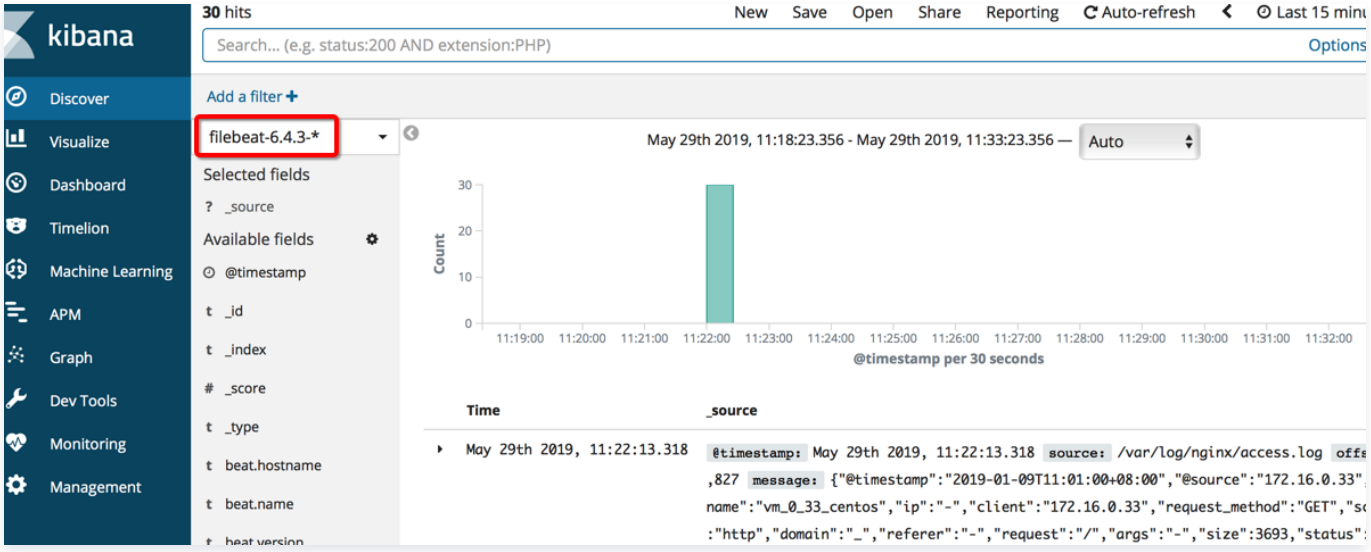
Querying logs.

1. In the ES console cluster list page, select **Operation > Kibana** to enter the Kibana console.

2. Enter **Management > Index Patterns** and add an index pattern named `filebeat-6.4.3-*`.



3. Click **Discover** and select the `filebeat-6.4.3-*` index item to retrieve nginx access logs.



Logstash + Elasticsearch + Kibana

Environment Preparation

- You need to create one or more CVM instances as needed in the same VPC as the ES cluster and deploy the Logstash component on them.
- The CVM needs to have more than 2 GB of memory.
- Install Java 8 or a later version in the created CVM.

Deploying Logstash

1. Download the Logstash package and decompress it

> **Note**
> The Logstash version should be compatible with the ES version.

```
wget https://artifacts.elastic.co/downloads/logstash/logstash-6.4.3.tar.gz
tar xvf logstash-6.4.3.tar.gz
```

2. Configure Logstash
In this example, the input source is the Nginx log, and the output item is configured to the private network VIP address and port of the ES cluster. Create the test.conf configuration file with the following content:

```
input {
 file {
     path => "/var/log/nginx/access.log" # Path to the NGINX access log
     start_position => "beginning" # Read the log from the beginning of the file. If this parameter is not
set, the log will be read when data is written to the file, just like tail -f
 }
}
filter {
}
output {
 elasticsearch {
     hosts => ["http://172.16.0.145:9200"] # Private VIP address and port of the ES cluster
     index => "nginx_access-%{+YYYY.MM.dd}" # Index name. Indices are automatically created on a daily
basis
     user => "elastic" # Username
     password => "yinan_test" # Password
  }
}
```

The ES cluster is configured to automatically create an index by default. The `nginx_access-%{+YYYY.MM.dd}` index in the above test.conf configuration file will be automatically created, and unless you need to set the mapping of the fields in the index in advance, you don't need to additionally call the API of ES to create indices.

3. Start Logstash
Enter the Logstash compressed package extraction directory logstash-6.4.3, and execute the following command to run Logstash in the background, **filling the configuration file path with your created path**.

```
nohup ./bin/logstash -f test.conf 2>&1 >/dev/null &
```

Check the logs directory under the logstash-6.4.3 directory to confirm that Logstash has started normally. If it starts normally, it will log the following:

```
Sending Logstash logs to /root/logstash-6.4.3/logs which is now configured via log4j2.properties
[2019-05-29T12:20:26,630][INFO ][logstash.setting.writabledirectory] Creating directory
{:setting=>"path.queue", :path=>"/root/logstash-6.4.3/data/queue"}
[2019-05-29T12:20:26,639][INFO ][logstash.setting.writabledirectory] Creating directory
{:setting=>"path.dead_letter_queue", :path=>"/root/logstash-6.4.3/data/dead_letter_queue"}
[2019-05-29T12:20:27,125][WARN ][logstash.config.source.multilocal] Ignoring the 'pipelines.yml' file
because modules or command line options are specified
[2019-05-29T12:20:27,167][INFO ][logstash.agent           ] No persistent UUID file found. Generating new
UUID {:uuid=>"2e19b294-2b69-4da1-b87f-f4cb4a171b9c", :path=>"/root/logstash-6.4.3/data/uuid"}
[2019-05-29T12:20:27,843][INFO ][logstash.runner          ] Starting Logstash {"logstash.version"=>"6.4.3"}
[2019-05-29T12:20:30,067][INFO ][logstash.pipeline        ] Starting pipeline {:pipeline_id=>"main",
"pipeline.workers"=>1, "pipeline.batch.size"=>125, "pipeline.batch.delay"=>50}
```

```
[2019-05-29T12:20:30,871][INFO ][logstash.outputs.elasticsearch] Elasticsearch pool URLs updated
{:changes=>{:removed=>[], :added=>[http://elastic:xxxxxx@10.0.130.91:10880/]}}
[2019-05-29T12:20:30,901][INFO ][logstash.outputs.elasticsearch] Running health check to see if an
Elasticsearch connection is working {:healthcheck_url=>http://elastic:xxxxxx@10.0.130.91:10880/,
:path=>"/"}
[2019-05-29T12:20:31,449][WARN ][logstash.outputs.elasticsearch] Restored connection to ES instance
{:url=>"http://elastic:xxxxxx@10.0.130.91:10880/"}
[2019-05-29T12:20:31,567][INFO ][logstash.outputs.elasticsearch] ES Output version determined
{:es_version=>6}
[2019-05-29T12:20:31,574][WARN ][logstash.outputs.elasticsearch] Detected a 6.x and above cluster: the
type event field won't be used to determine the document _type {:es_version=>6}
[2019-05-29T12:20:31,670][INFO ][logstash.outputs.elasticsearch] New Elasticsearch output
{:class=>"LogStash::Outputs::ElasticSearch", :hosts=>["http://10.0.130.91:10880"]}
[2019-05-29T12:20:31,749][INFO ][logstash.outputs.elasticsearch] Using mapping template from {:path=>nil}
[2019-05-29T12:20:31,840][INFO ][logstash.outputs.elasticsearch] Attempting to install template
{:manage_template=>{"template"=>"logstash-*", "version"=>60001, "settings"=>
{"index.refresh_interval"=>"5s"}, "mappings"=>{"_default_"=>{"dynamic_templates"=>[{"message_field"=>
{"path_match"=>"message", "match_mapping_type"=>"string", "mapping"=>{"type"=>"text", "norms"=>false}}},
{"string_fields"=>{"match"=>"*", "match_mapping_type"=>"string", "mapping"=>{"type"=>"text",
"norms"=>false, "fields"=>{"keyword"=>{"type"=>"keyword", "ignore_above"=>256}}}}}], "properties"=>
{"@timestamp"=>{"type"=>"date"}, "@version"=>{"type"=>"keyword"}, "geoip"=>{"dynamic"=>true,
"properties"=>{"ip"=>{"type"=>"ip"}, "location"=>{"type"=>"geo_point"}, "latitude"=>
{"type"=>"half_float"}, "longitude"=>{"type"=>"half_float"}}}}}}}}
[2019-05-29T12:20:32,094][INFO ][logstash.outputs.elasticsearch] Installing elasticsearch template to
_template/logstash
[2019-05-29T12:20:33,242][INFO ][logstash.inputs.file     ] No sincedb_path set, generating one based on
the "path" setting {:sincedb_path=>"/root/logstash-
6.4.3/data/plugins/inputs/file/.sincedb_d883144359d3b4f516b37dba51fab2a2", :path=>
["/var/log/nginx/access.log"]}
[2019-05-29T12:20:33,329][INFO ][logstash.pipeline       ] Pipeline started successfully
{:pipeline_id=>"main", :thread=>"#<Thread:0x12bdd65 run>"}
[2019-05-29T12:20:33,544][INFO ][logstash.agent          ] Pipelines running {:count=>1,
:running_pipelines=>[:main], :non_running_pipelines=>[]}
[2019-05-29T12:20:33,581][INFO ][filewatch.observingtail  ] START, creating Discoverer, Watch with file and
sincedb collections
[2019-05-29T12:20:34,368][INFO ][logstash.agent          ] Successfully started Logstash API endpoint
{:port=>9600}
```

For more information on the features of Logstash, see Elastic's official documentation.

Querying logs.

Please refer to Querying logs.
For more information on the features of the Kibana Console, see Elastic's official documentation.

Tencent Cloud ES + SCF for Quick Search Service

Last updated：2024-10-25 09:28:18

Search Service

Search services are everywhere, such as Google search in daily life, wiki search in work, and product search in shopping. Data in such scenarios is generally large in size and structured and has more reads than writes. However, due to their transaction characteristics, traditional databases cannot well utilize space in search scenarios and are slow in full-text searches (such as LIKE statement). Elasticsearch thus came into being.
Elasticsearch is an open-source search engine widely used in full-text search. It can quickly index, search, and analyze a massive amount of text data. Tencent Cloud ES is a highly available and scalable cloud-hosted Elasticsearch service. It well supports both structured and unstructured data and provides easy-to-use RESTful APIs and clients in various languages to help you quickly build stable search services.
This document uses the official Tencent Cloud ES documentation as a corpus to describe how to quickly build a search service using Tencent Cloud ES + SCF. The interface of the search service is shown as follows:

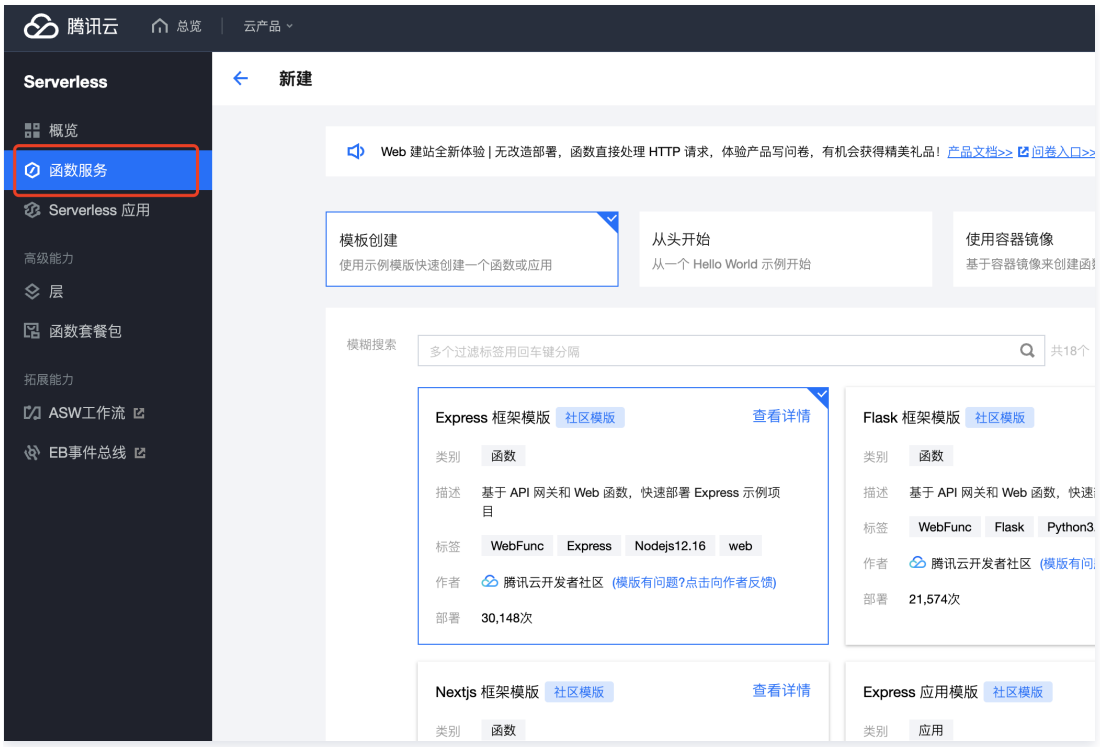
Resource Preparations

Purchase an ES cluster. The cluster size is determined by the QPS of the search service and the data volume of the stored documents. For more details, refer to Evaluation of Cluster Specification and Capacity Configuration.

Deploying Search Service

Use Tencent Cloud's **free** SCF tool to deploy the frontend interface and backend services of the search service.
1. In the **SCF** > **Function Services** page, select the region of the purchased ES cluster in the top-left corner.

Search for the required CAM policy as needed, and click to complete policy association.



2. Create a function service, **remember the function name you set. The rest can be selected as per the screenshot content.**

Search for the required CAM policy as needed, and click to complete policy association.

3. In the advanced configuration below the creation process, enable **VPC** and ensure it is consistent with the VPC of the ES cluster.

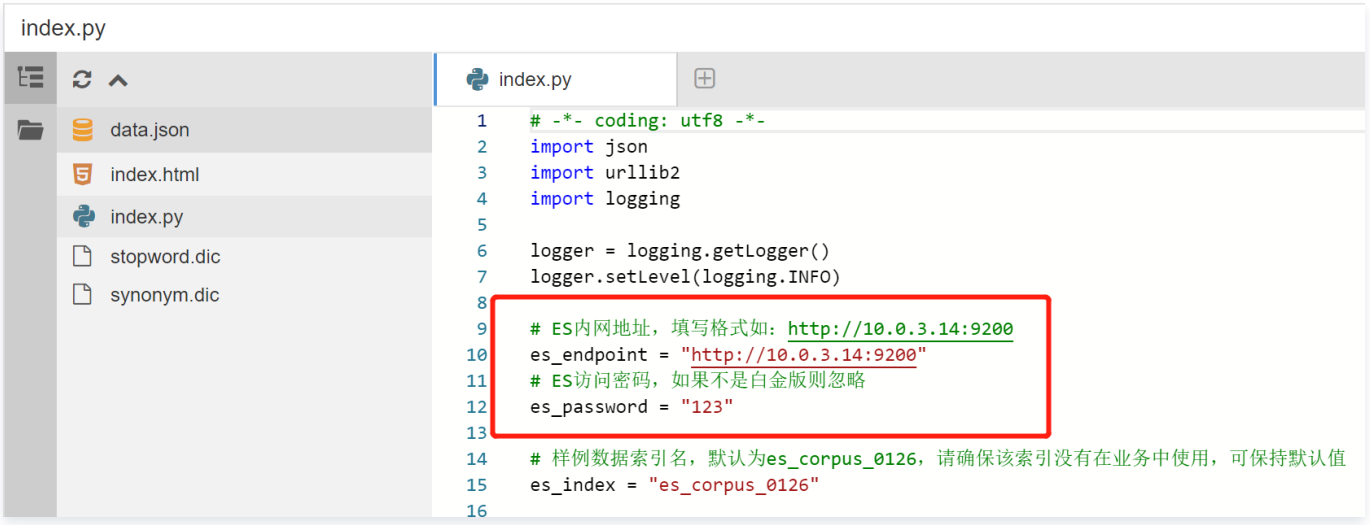
Search for the required CAM policy as needed, and click to complete policy association.

4. In the function code interface, first download the code zip package to your local machine. Then, in **Function Code > Submission Method**, select **Upload Local Zip Package**, choose the zip package you just downloaded, and click **Save**.

Search for the required CAM policy as needed, and click to complete policy association.

5. Modify the code on the **Function Code** page. You need to modify the `index.py` and `index.html` files:

- Change `index.py` `es_endpoint` to the private network address of your ES cluster, formatted as: `http://10.0.3.14:9200`.
- Change `index.py` `es_password` to the Platinum ES password. If it is not Platinum, do not change it.



- In `index.html`, change `server_name` to the name of the SCF function you created. The default is `myserver`.

Note

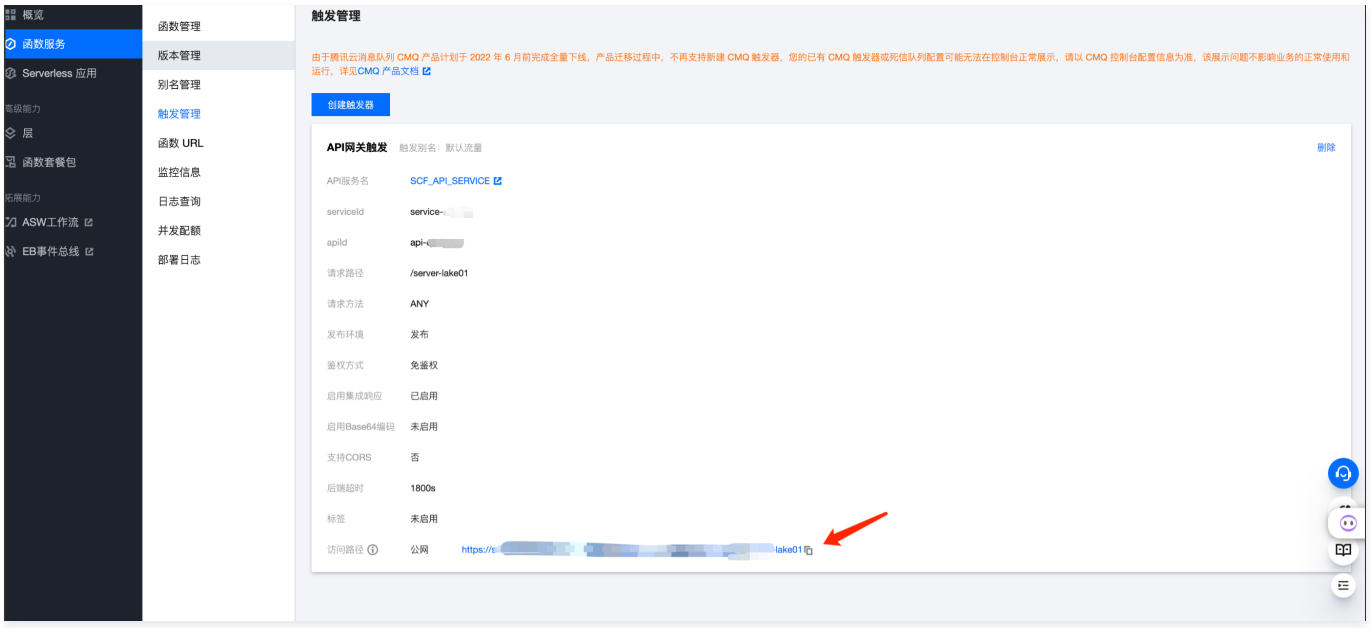
By default, the sample uses `es_corpus_0126` as the index name. Please ensure this index is not in use by any business. If needed, you can modify the `es_index` variable in `index.py`.

6. On the Trigger Management Page, click to create a **Trigger Method**, add an API Gateway trigger as shown below, enable Integration Response, and then click **Save**.

Search for the required CAM policy as needed, and click to complete policy association.

7. You can see the function's **Access Path** in Trigger Management. Click this path to access the page.

Search for the required CAM policy as needed, and click to complete policy association.



8. Upload Tencent Cloud ES official documentation sample data. Click the text above the search box to automatically import data.

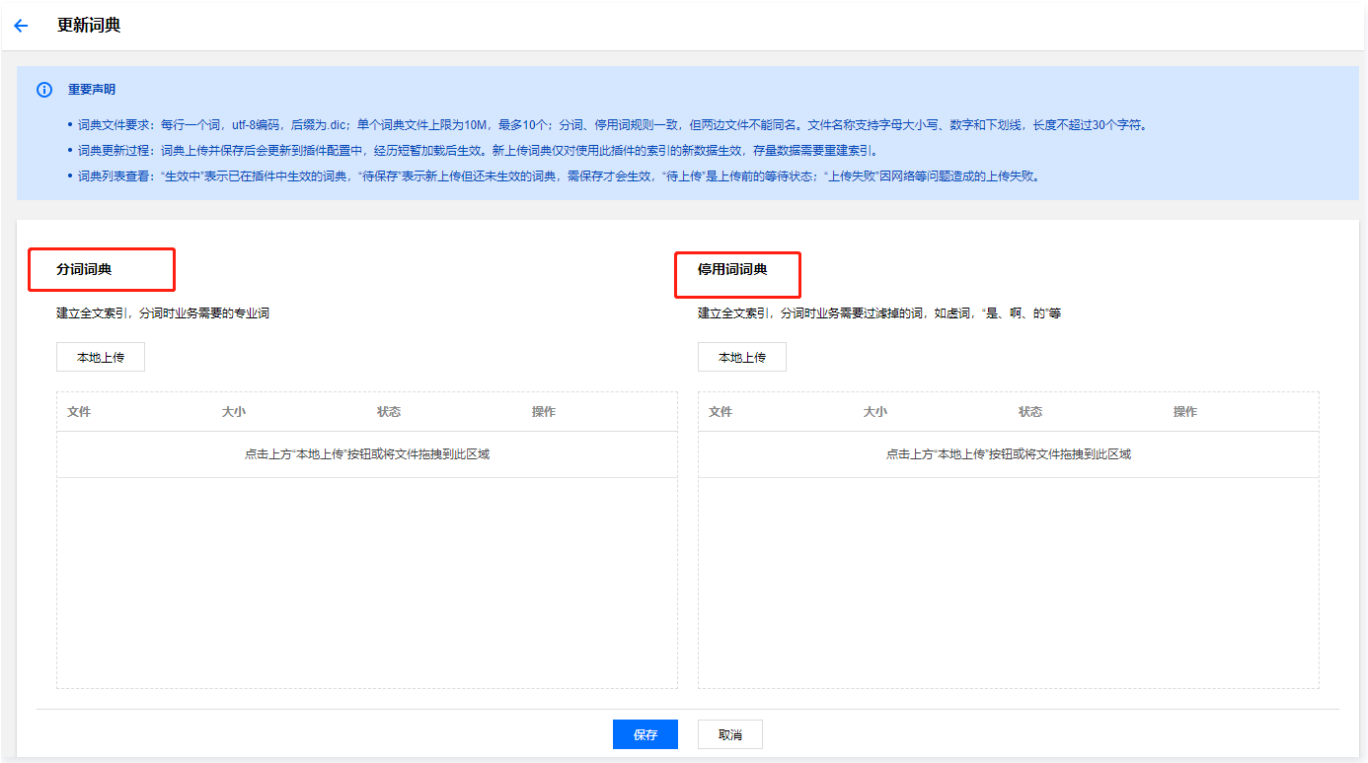
9. At this point, you have deployed a simple ES-based Q&A search service backend.

Learn More

Importing stopword and custom dictionary

Stopwords will not be searched by ES, and words in the custom dictionary will be retained during segmentation. In the previous sample, the default stopword library and custom dictionary are imported. You can also import your own stopword and custom

dictionaries in the **plugin list > update dictionary** on the ES cluster details page.



Synonym configuration

To configure synonyms, you need to specify them when creating an index. Synonyms in Solr and WordNet formats are supported. For more information on the format, see Solr synonyms.

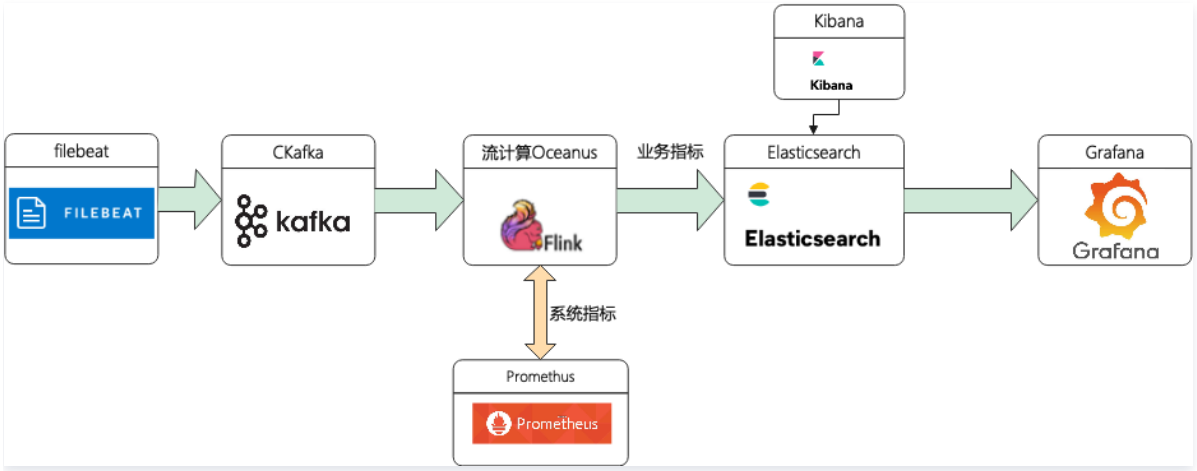
Real-time monitoring solution based on SCS

Last updated: 2024-10-25 09:28:53

This solution combines Tencent Cloud CKafka, SCS, Tencent Cloud Database Elasticsearch, TCOP, etc. Using Filebeat to monitor system logs and application logs in real-time, the monitoring data is transmitted to Tencent Cloud CKafka. The data in Kafka is then integrated into SCS, undergoes simple business logic processing, and is output to the Cloud Database Elasticsearch. Cloud Prometheus is used to monitor system metrics, and Cloud Grafana is utilized for personalized business data monitoring of Oceanus jobs.

Scheme Architecture

Search for the required CAM policy as needed, and click to complete policy association.



Preparation

Before using, please ensure that the corresponding big data components have been purchased and created.

Creating VPC

A VPC is a logically isolated network space self-defined by you on Tencent Cloud. You need to use Peering Connection, NAT Gateway, etc., to connect the network. For specific creation steps, please refer to Creating VPC.

> **Note**
> When building CKafka, Oceanus, Elasticsearch clusters, the network selected must be consistent for network intercommunication.

Create CKafka instance

Enter Message Queue CKafka Console, in the instance list click **Create**, for details on creating CKafka instance, refer to Creating Instance. After purchasing, create a Kafka topic (`topic-app-info`), for detailed operations, refer to Creating Topic.

> **Note:**
> - **VPC and subnet: select the network and subnet created.**
> - It is recommended to choose the latest version of Kafka to ensure better compatibility with the Filebeat collection tool.

Search for the required CAM policy as needed, and click to complete policy association.

Creating Oceanus cluster

SCS services are compatible with native Flink tasks. In the SCS console, go to **Computing Resources > + Create New** to create a cluster. Select the region, availability zone, VPC, log, storage, set the initial password, etc. **Select the newly created network for VPC and subnet**. Specific creation steps can refer to Creating Dedicated Cluster. After creation, the Flink cluster is as follows:
Search for the required CAM policy as needed, and click to complete policy association.

Create Elasticsearch Instance

Go to the ES console, click new in the ES cluster management**Create Cluster**,**select the previously created VPC and subnet**, and set the account and password. Detailed operations can refer to Creating Cluster.

Search for the required CAM policy as needed, and click to complete policy association.

Create TCOP Service

To display system metrics from the Definition, you need to purchase Prometheus services. If only Definition business metrics are needed, this step can be skipped.
Go to the TCOP page, click on the left Prometheus Monitoring, and click **Create New,select the previous VPC and subnet**, and set the instance name and Grafana password. Detailed operations can refer to Creating Instance.

Search for the required CAM policy as needed, and click to complete policy association.

Create Independent Grafana Resources

In the Grafana management page, make a separate purchase to display business monitoring metrics. Click **Create New** to enter the Grafana visualization service purchase page. For details, see Creating Instance. When purchasing, you still need to select the same VPC network as other resources.

Install and Configure Filebeat

Filebeat is a lightweight log data collection tool that collects information by monitoring designated files. Install Filebeat on the CVMs requiring monitoring of host and application information under this VPC (the configuration is generally in filebeat.yml).

- Method 1: Download and install Filebeat Filebeat Download Link.
- Method 2: Use the Filebeat provided in the **Elasticsearch Management Page > Beats Management**. In this example, Method 1 is used.
  Download to CVM and configure Filebeat. Add the following configuration items in the filebeat.yml file:```shell.

Log file monitoring configuration

- type: log
  enabled: true
  paths:
  - /tmp/test.log
    #- c:\programdata\elasticsearch\logs*

```
```shell
#  Monitoring data output configuration
output.kafka:
version: 2.0.0                          # Kafka version
hosts: ["xx.xx.xx.xx:xxxx"]             # Please fill in the actual IP address and port
topic: 'topic-app-info'                   # Please fill in the actual topic
```

Please configure the corresponding filebeat.yml file according to actual business needs. Refer to Filebeat official documentation.

> **Note**
> The example uses version 2.4.1 of Ckafka. Here, the configuration is version: 2.0.0. If the versions don't match, an `ERROR [kafka] kafka/client.go:341 Kafka (topic=topic-app-info): dropping invalid message` error might occur.

Solution Implementation

Next, we will introduce how to achieve personalized monitoring through SCS through a case study.

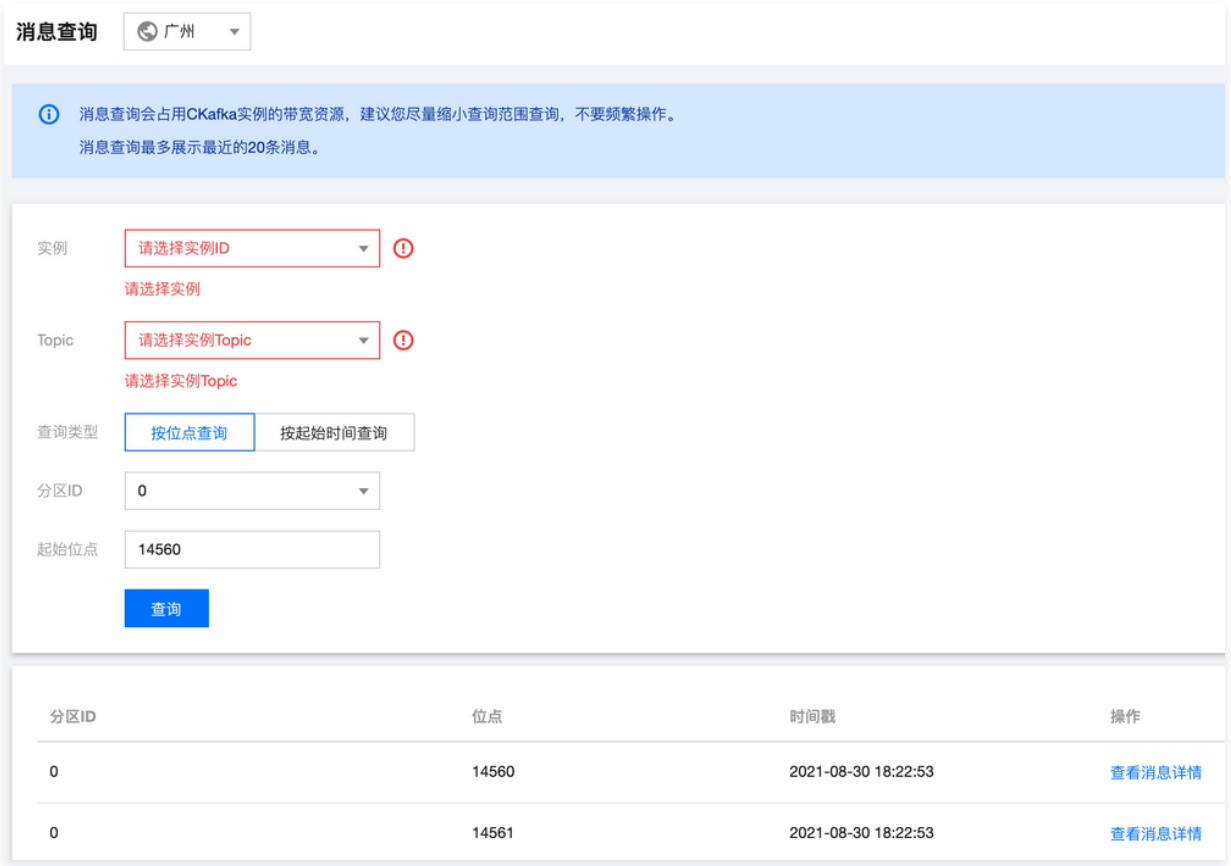
Filebeat Data Transmission

1. Go to the root directory of Filebeat and start Filebeat for data acquisition. In the example, data such as CPU, memory shown in the top command are collected. Logs of jar applications, JVM usage, listening ports, etc. can also be collected. Refer to Filebeat official website for more details.

```
# Start Filebeat
./filebeat -e -c filebeat.yml

# Write monitored system information to the test.log file
top -d 10 >>/tmp/test.log
```
```

2. Go to the Ckafka page, click **Message Query** to query the corresponding topic messages and verify if the data has been collected.



Data format collected by Filebeat:


```
{
  "@timestamp": "2021-08-30T10:22:52.888Z",
  "@metadata": {
"beat": "filebeat",
"type": "_doc",
"version": "7.14.0"
  },
  "input": {
"type": "log"
  },
  "host": {
"ip": ["xx.xx.xx.xx", "xx::xx:xx:xx:xx"],
"mac": ["xx:xx:xx:xx:xx:xx"],
"hostname": "xx.xx.xx.xx",
"architecture": "x86_64",
"os": {
  "type": "linux",
  "platform": "centos",
  "version": "7(Core)",
  "family": "redhat",
  "name": "CentOSLinux",
  "kernel": "3.10.0-1062.9.1.el7.x86_64",
  "codename": "Core"
},
"id": "0ea734564f9a4e2881b866b82d679dfc",
"name": "xx.xx.xx.xx",
"containerized": false
  },
  "agent": {
"name": "xx.xx.xx.xx",
```

```
  "type": "filebeat",
  "version": "7.14.0",
  "hostname": "xx.xx.xx.xx",
  "ephemeral_id": "6c0922a6-17af-4474-9e88-1fc3b1c3b1a9",
  "id": "6b23463c-0654-4f8b-83a9-84ec75721311"
   },
   "ecs": {
  "version": "1.10.0"
   },
   "log": {
  "offset": 2449931,
  "file": {
    "path": "/tmp/test.log"
  }
   },
   "message": "?(B?[m16root0-20000S0.00.00:00.00kworker/1:0H?(B?[m?[39;49m?[K"
}
```

SQL Assignment Writing

In SCS, process the data accessed from Kafka and store it in Elasticsearch.

1. Definition source

Construct Flink Source according to the JSON message formats in Filebeat.

```
CREATE TABLE DataInput (
    @timestamp VARCHAR,
    host       ROW<id VARCHAR,ip ARRAY<VARCHAR>>,
    log        ROW<offset INTEGER,file ROW<path VARCHAR>>,
    message    VARCHAR
) WITH (
    'connector' = 'kafka',   -- Optional: 'kafka', 'kafka-0.11'. Make sure to select the corresponding built-in Connector
    'topic' = 'topic-app-info',  -- Replace with the Topic you want to consume
    'scan.startup.mode' = 'earliest-offset', -- Can be any of latest-offset / earliest-offset / specific-offsets / group-offsets
    'properties.bootstrap.servers' = '10.0.0.29:9092',  -- Replace with your Kafka connection address
    'properties.group.id' = 'oceanus_group2',  -- Required parameter, make sure to specify a Group ID
    -- Definition Data formats (JSON formats)
    'format' = 'json',
    'json.ignore-parse-errors' = 'true',     -- Ignore JSON structure parsing exceptions
    'json.fail-on-missing-field' = 'false'   -- If set to true, an error will be thrown when a field is missing. If set to false, the missing field will be set to null
);
```

2. Definition sink

```
CREATE TABLE es_output (
    id VARCHAR,
    ip ARRAY<VARCHAR>,
    path VARCHAR,
    num INTEGER,
    message VARCHAR,
    createTime VARCHAR
) WITH (
    'connector.type' = 'elasticsearch', -- Output to Elasticsearch
    'connector.version' = '6',          -- Specifies the version of Elasticsearch, e.g., '6', '7'.
    'connector.hosts' = 'http://10.0.0.175:9200',  -- Elasticsearch connection address
    'connector.index' = 'oceanus_test2',       -- Name of the Elasticsearch Index
    'connector.document-type' = '_doc',  -- Type of the Elasticsearch document
```

```
    'connector.username' = 'elastic',
    'connector.password' = 'yourpassword',
    'update-mode' = 'upsert',            -- Optional 'append' mode without primary key or 'upsert' mode with
primary key
    'connector.key-delimiter' = '$',     -- Optional parameter, the delimiter for composite primary keys
(defaults to '_', e.g., key1_key2_key3)
    'connector.key-null-literal' = 'n/a',  -- The substitute string when the primary key is null, defaults to
'null'
    'connector.failure-handler' = 'retry-rejected',   -- Optional error handling. Options are 'fail' (throw
exception), 'ignore' (ignore any error), 'retry-rejected' (retry)

    'connector.flush-on-checkpoint' = 'true',   -- Optional parameter, disallows batch writes (flush) during
snapshots, defaults to true
    'connector.bulk-flush.max-actions' = '42',  -- Optional parameter, maximum number of actions per batch
    'connector.bulk-flush.max-size' = '42 mb',  -- Optional parameter, maximum accumulated size per batch
(supports only mb)
    'connector.bulk-flush.interval' = '60000',  -- Optional parameter, interval for batch writes (ms)
    'connector.connection-max-retry-timeout' = '1000',      -- Maximum timeout for each request (ms)
    --'connector.connection-path-prefix' = '/v1'          -- Optional field, the path prefix appended to each
request
    'format.type' = 'json'        -- Output data format, currently supports only 'json'
);
```

3. Configure the business logic

```
INSERT INTO es_output
SELECT
host.id as id,
host.ip as ip,
log.file.path as path,
log.offset as num,
message,
@timestamp as createTime
from DataInput;
```

4. ES Data Query

Query data on the Kibana page in the ES console, or enter a CVM in the same subnet and use the following command to query:

```
# Query index. Replace username:password with the actual account password
curl -XGET -u username:password http://xx.xx.xx.xx:xxxx/oceanus_test2/_search -H 'Content-Type:
application/json' -d'
{
    "query": { "match_all": {}},
    "size":  10
}
'
```

For more access methods, please refer to Accessing the ES Cluster.

System Metrics Monitoring

This section mainly implements system information monitoring and monitoring and alerting for the running status of Flink jobs. Prometheus is a highly flexible time series database, commonly used for monitoring data storage, computation, and alerting. SCS recommends users to use the Prometheus service provided by TCOP to avoid deployment and operation and maintenance expenses; it also supports Tencent Cloud's notification templates, allowing alerts to be easily sent to different recipients via SMS, telephone, email, Enterprise WeChat robot, etc.

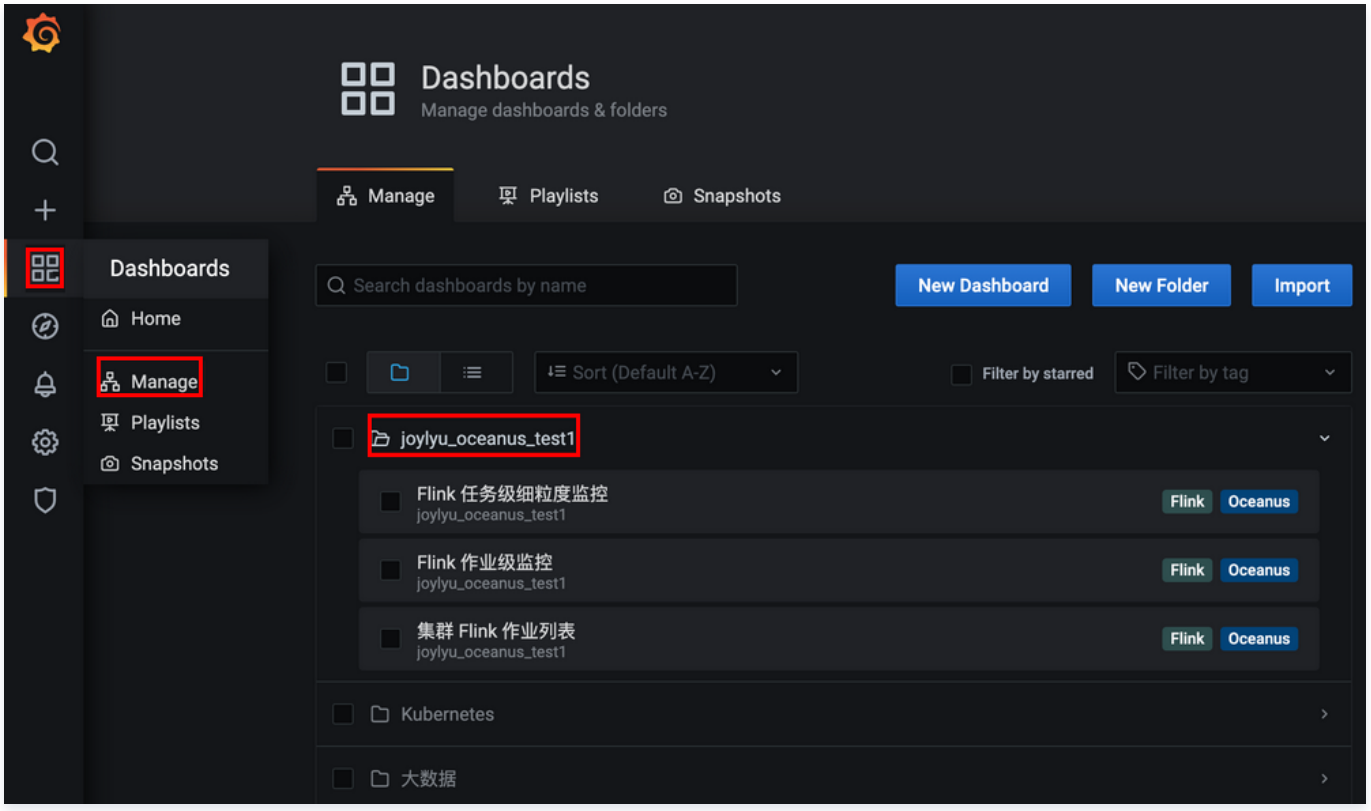
Configuring Monitoring (Oceanus Job Monitoring)

In addition to the monitoring information provided by the Oceanus Console, you can configure task-level fine-grained monitoring, job-level monitoring, and cluster Flink job list monitoring.

1. On the Oceanus Job Details Page, click **Job Parameters**, and add the following configuration under **Advanced Parameters**:

```
pipeline.max-parallelism: 2048
metrics.reporters: promgateway
metrics.reporter.promgateway.host: xx.xx.xx.xx              # Prometheus Instance Address
metrics.reporter.promgateway.port: 9090                     # Prometheus Instance Port
metrics.reporter.promgateway.needBasicAuth: true
metrics.reporter.promgateway.password: xxxxxxxxxx           # Prometheus Instance Password
metrics.reporter.promgateway.interval: 10 SECONDS
```

2. In any Oceanus job, click **Cloud Monitor** to enter the Cloud Prometheus instance. Click the link to enter Grafana (you cannot enter Grafana in grayscale from this link), import the JSON file, and for more details, please refer to Accessing Prometheus Custom Monitoring.

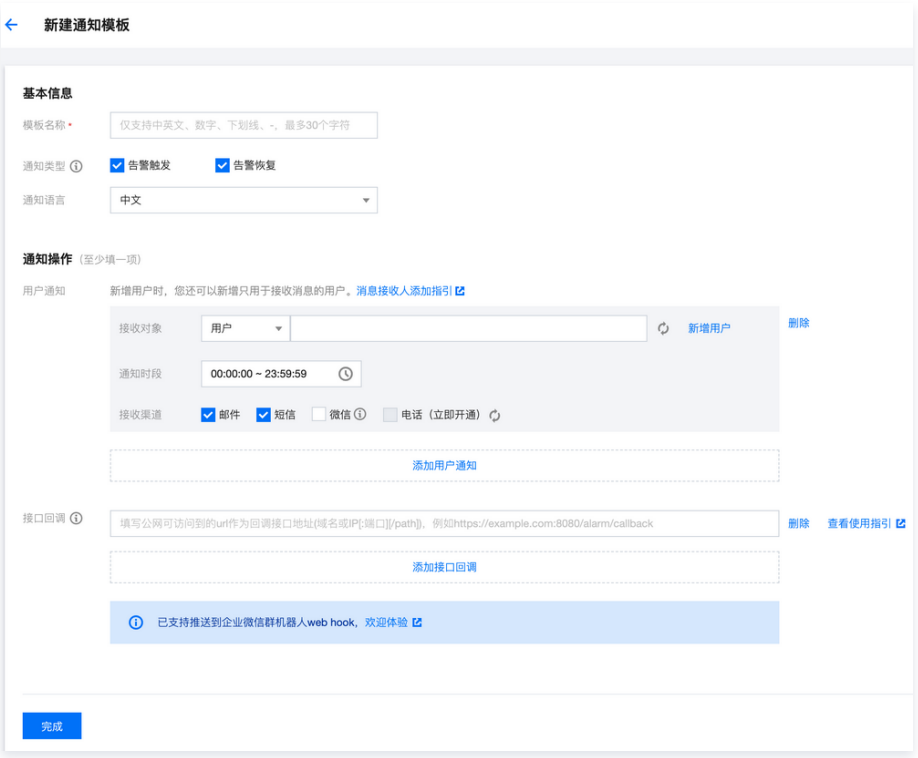


3. The displayed Flink task monitoring effect is as follows; users can also click **Edit** to set different panels to optimize the display effect.

Alarm Configuration

1. Enter the TCOP interface, click **Prometheus Monitoring**, click on the purchased instance to go to the service management page, then select **Alert Policy > New**, and configure the relevant information. For detailed operations, refer to Accessing Prometheus Custom Monitoring.

2. Set alarm notification. Choose **Select Template** or **New**, and set the notification template.



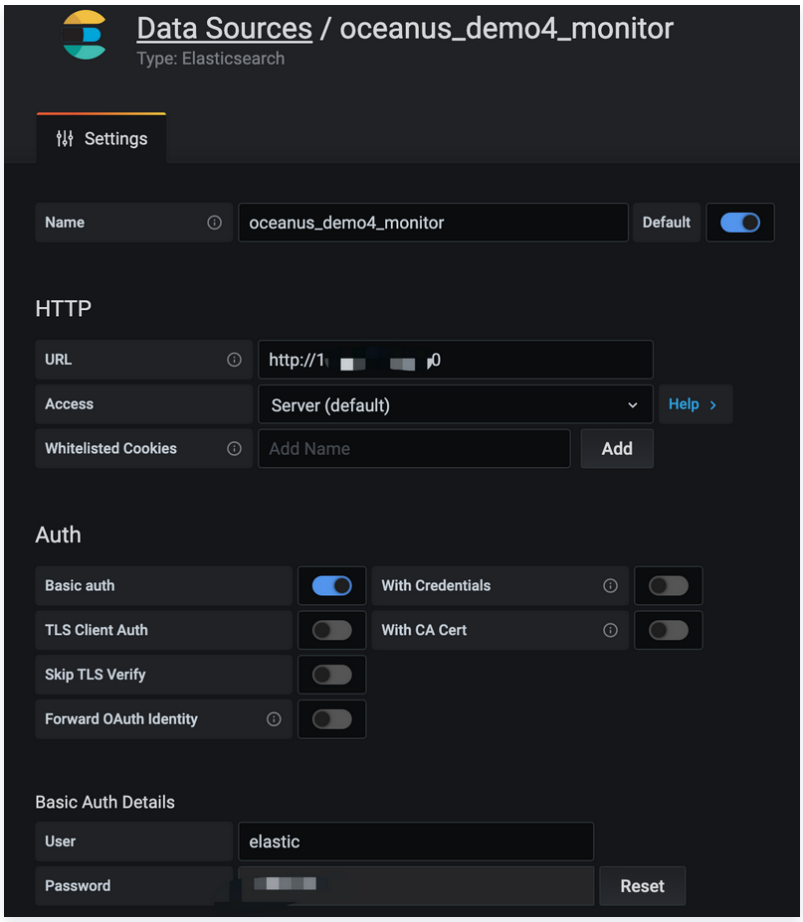
3. SMS notification message.

Business Metric Monitoring

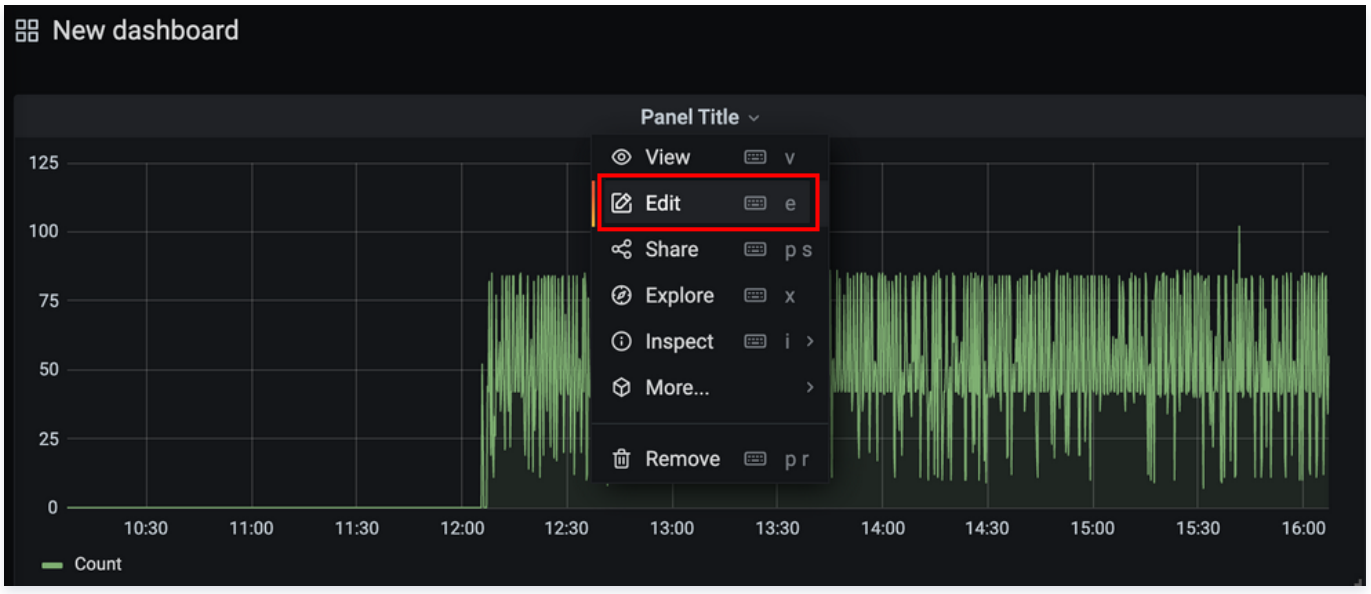
The application business data collected by Filebeat, processed by Oceanus Service, has been stored into ES. Business data monitoring can be realized through ES and Grafana.

1. Grafana configures the ES data source. Enter the Grafana Console in the grayscale release, go to the newly created TCMG, find and open the public network address. The Grafana account is admin, log in to and then select **Configuration > Add Source**, search

`elasticsearch`, fill in the relevant ES instance information, and add the data source.



2. Select **Dashboards > Manage** on the left side, click **New Dashboard** on the top right corner, and create a new panel.



Display effect as follows:

- `Total Data Volume Write Real-time Monitoring`: Monitoring the total data volume written to the data source.
- `Data Source Real-time Monitoring`: Monitoring the data write volume from a specific log.
- `Field Average Value Monitoring`: Monitoring the average value of a specific field.
- `num Field Maximum Value Monitoring`: Monitoring the maximum value of the num field.

Note:
This is for demonstration only and has no actual business purposes.

Summary

This solution attempts both system monitoring metrics and business monitoring metrics. If you only need to monitor business metrics, you can skip the Promethus-related operations. Additionally, please note:

1. The version of Ckafka does not strictly correspond to the open-source version of Kafka. In this solution, Ckafka 2.4.1 and open-source Filebeat-1.14.1 versions can be successfully debugged.
2. The Promethus Service in TCOP has already been integrated into the Grafana Monitoring Service. However, it does not support custom data sources. This embedded Grafana can only connect to Promethus. You need to use the Independently Grayscale Released Grafana to access ES Data to Grafana.

Index Configuration
Default Index Template Description and Adjustment

Last updated: 2024-10-25 09:29:23

Default Template Description

Index Template is a pre-defined template that is automatically applied when creating a new index. It mainly includes index settings, mapping, and template priority configurations. Tencent Cloud ES provides a default index template when creating a cluster. You can choose ES Cluster Management in the console, click the Kibana button in the operation column of the corresponding cluster, enter the Kibana interface, and select Dev Tools in the left-side Management. Use the command GET _template/default@template to view this template. Below are the default template and some explanations of its configuration, which can be adjusted according to your needs.

```
{
  "default@template": {
    order: 1, // Template priority, the higher the value is, the higher the priority is
    index_patterns: [// Index to which the template is applied
      "*"
    ],
    "settings": {
      "index": {
        max_result_window: 65536, // Maximum query result window. If the result quantity exceeds this value,
error message "Result window is too large" will be displayed, and you will need to increase this value.
        "routing": {
          "allocation": {
            "include": {
              "temperature": "hot"
            }
          }
        },
        refresh_interval: 30s, // Index refresh interval. The indexed document can only be queried after the
interval elapses. If you have high requirement for real-time query, you can properly reduce this value, but a
too small value will compromise the write performance.
        "unassigned": {
          "node_left": {
            "delayed_timeout": "5m"
          }
        },
        "translog": {
          sync_interval: 5s, // Translog flush interval. A too small value will compromise the write
performance.
          "durability": "async"
        },
        number_of_replicas: 1 // Number of replica shards
      }
    },
    "mappings": {
      "_default_": {
        "_all": {
          enabled: false // It is recommended to set this parameter to disabled. The _all field contains all
other fields to form a large string, which will take up a lot of disk space and compromise the write
performance
        },
        dynamic_templates: [ // Dynamic template
          {
            message_full: { // Dynamically map the field named message_full to text and keyword types
              "match": "message_full",
              "mapping": {
                "type": "text",
                "fields": {
```

```
                "keyword": {
                  "type": "keyword",
                  "ignore_above": 2048
                }
              }
            }
          }
        },
        {
          message: { // Dynamically map the field named message to text type
            "match": "message",
            "mapping": {
              "type": "text"
            }
          }
        },
        {
          strings: { // Dynamically map a field of string type to keyword type
            "match_mapping_type": "string",
            "mapping": {
              "type": "keyword"
            }
          }
        }
      ]
    },
    "aliases": {}
  }
}
```

Adjusting Template

In the Kibana interface, use the command `PUT _template/my_template` in **Dev Tools** to define your own index template, and set the template priority `order` value greater than the default template priority to overwrite the default index template configuration.

> **Note**
> The index template is only applied when an index is created; therefore, template adjustment will not affect existing indexes.

Adjust the number of primary shards

In Elasticsearch 5.6.4 and 6.4.3, an index has 5 primary shards by default.
For scenarios with a small data size and a large number of indices, you are recommended to lower the number of primary shards so as to reduce the pressure of index metadata on the heap memory. You can do so by referring to the following template:

```
{
  "index_patterns" : ["*"],
    "order" : 2, // Make sure that the order field value in the template is greater than 1. Order represents
the priority of the template; the higher the value, the higher the priority
    "settings" : {
        "index": {
            "number_of_shards" : 1 // number_of_shards represents the number of shards in the index
        }
    }
}
```

Adjusting Field Type

In the default template, we dynamically map string type fields to keyword types to prevent full-text indexing on all text data. You can modify specific string type fields to text based on business needs, allowing full-text indexing.

```
{
  "index_patterns" : ["*"],
    order : 2, //Make sure that the value of the order field in the template is greater than 1.
    "mappings": {
      "properties": {
        Field Name: {
          "type":  "text" //type indicates the data type of the field
        }
    }
  }
}
```

Other Business Scenarios

For example, if you want the indexed documents to be searchable after 10 seconds and apply to all `search-*` indices, you can create a template as follows:

```
{
    "index_patterns" : ["search-*"],
    "order" : 2, // Make sure that the order field value in the template is greater than 1. Order represents
the priority of the template; the higher the value, the higher the priority
    "settings" : {
        "index": {
            "refresh_interval": "10s"
        }
    }
}
```

Managing Indices with Curator

Last updated: 2024-10-25 09:29:45

Curator is a tool officially released by Elastic for managing Elasticsearch indexes. It can perform many index lifecycle management tasks, such as cleaning up indexes created more than 7 days ago, scheduling daily backups of specific indexes, and migrating indexes from hot nodes to cold nodes, among others. For more operations supported by Curator, please refer to the official documentation's feature list.
Curator offers a command line CLI tool that allows tasks to be executed through parameter configuration. Curator also provides a comprehensive Python API, enabling integration with Tencent Cloud's Serverless Cloud Function (SCF). For example, using Curator to automatically delete expired data in Tencent Cloud Elasticsearch. Tencent Cloud's serverless SCF has Curator templates configured, and users can operate it with simple parameter settings after applying the template. For more ways to use Tencent Cloud's Serverless SCF, please refer to SCF.

Curator Usage Example

The following describes how to configure and run Curator to delete expired indices regularly.

Installing

Purchase a CVM instance in the VPC where your Elasticsearch cluster resides and install the Curator package via pip.

```
pip install elasticsearch-curator
```

Running as command line parameters

The following command will filter the index names matching the logstash-20xx-xx-xx format and those dated 7 days ago, and then delete those indexes.

> **Note**
> The sample code performs a deletion operation to clear your data. Make sure that the above statement has been tested in a non-production environment. You can add the `--dry-run` parameter for testing purposes to avoid actually deleting data.

```
curator_cli --host 10.0.0.2:9200 --http_auth 'user:passwd' delete-indices --filter_list '[{"filtertype":
"pattern", "kind": "prefix", "value": "logstash-"}, {"filtertype": "age", "source": "name", "direction":
"older", "timestring": "%Y.%m.%d", "unit": "days", "unit_count": 7}]'
```

Running as configuration files

If your operations are complex, have too many parameters, or you do not want to use command line parameters, you can place the parameters in configuration files for execution.\nIn the specified config directory, you need to edit the files config.yml and action.yml, and add them to the path you have designated.

```
curator_cli --config PATH (replace PATH with your actual config directory)
```

Scheduled execution

If you need a scheduled run, you can configure the command to a crontab on Linux, or directly use the timer trigger feature of Tencent Cloud SCF mentioned above.

Using the API

For more information on how to use the Python APIs, please see documentation.

Heat-Temperature Separation and Index Lifecycle Management

Last updated: 2024-10-25 09:30:14

Elasticsearch is mainly used for storing and retrieving huge data. Storing all data on SSD drives would be very costly. Heat-Temperature Separation can solve this problem. A Hot-Warm Cluster can include both warm and hot nodes within a single cluster, balancing performance and capacity.

- Hot data with high read-write performance requirements (such as logs within 7 days) can be stored on the hot node in SSD disks.
- Indexes with large storage requirements but low read/write performance demands (e.g., logs older than a month) can be stored on Warm Nodes using SATA disks.

Tencent Cloud ES provides the ability to quickly configure and build Hot-Warm Clusters. Users can specify Warm Node Specifications based on business needs via the Tencent Cloud website to quickly set up a Heat-Temperature Separation Architecture ES cluster.

Create a Hot-Warm Cluster

Creating directly when purchasing the cluster

1. Go to the Tencent Cloud ES Create Cluster page, and fill in the necessary information to create the cluster.
2. Choose **Warm Data Nodes** for the data node deployment method and select the Warm Node Specifications as shown below:
   Search for the required CAM policy as needed, and click to complete policy association.

3. Further set other parameters of the cluster, confirm and pay.

Modify an existing cluster to a Hot-Warm Cluster

In the Cluster Details Page, select the top right **More Operations > Adjust Configuration**. In the Adjust Configuration page, set the **Data Node Deployment Mode** to **Warm Mode**. Specify the Warm Node Specifications and related configurations as required to transform the existing cluster to a Hot-Warm Cluster.

Using a Hot-Warm Cluster

View node roles

Verify Node Warm Attributes, using the following command:

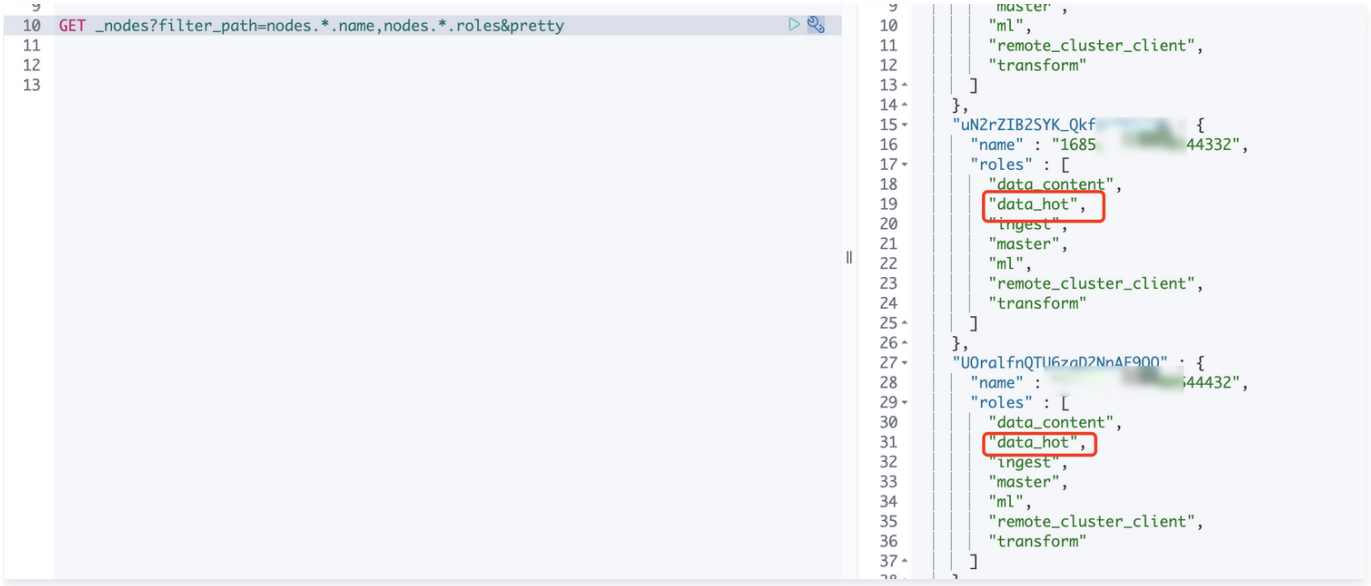
```
GET _cat/nodeattrs?v&h=node,attr,value&s=attr:desc
```

```
node        attr        value
node1   temperature     hot
node2   temperature     hot
node3   temperature     warm
node4   temperature     hot
node5   temperature     warm
...
```

To view node attributes in version 7.14.2, use the following command:

```
GET _nodes?filter_path=nodes.*.name,nodes.*.roles&pretty
```

Search for the required CAM policy as needed, and click to complete policy association.



Specify the warm property of the index

The business party can decide the warm property of the index based on the actual situation.

1. Create an index.

```
PUT hot_warm_test_index
{
   "settings": {
      "number_of_replicas": 1,
      "number_of_shards": 3
   }
}
```

2. View the shard allocation. You can see that shards are evenly distributed on five nodes.

```
GET _cat/shards/hot_warm_test_index?v&h=index,shard,prirep,node&s=node
index           shard prirep node
hot_data_index 1     p      node1
hot_data_index 0     r      node1
hot_data_index 2     r      node2
hot_data_index 2     p      node3
hot_data_index 1     r      node4
hot_data_index 0     p      node5
```

3. Set the index.
   - Set the index to hot index.

```
PUT hot_warm_test_index/_settings
{
 "index.routing.allocation.require.temperature": "hot"
}
//In version 7.14.2, the command to change the index to hot attribute is as follows:
PUT hot_warm_test_index/_settings
{
 "index.routing.allocation.include._tier_preference": "data_hot"
}
```

View the shard allocation. As you can see, all shards are allocated on hot nodes.

```
GET _cat/shards/hot_warm_test_index?v&h=index,shard,prirep,node&s=node
index          shard prirep node
hot_data_index 1     p      node1
hot_data_index 0     r      node1
hot_data_index 0     p      node2
hot_data_index 2     r      node2
hot_data_index 2     p      node4
hot_data_index 1     r      node4
```

- Set the index as a Warm Index.

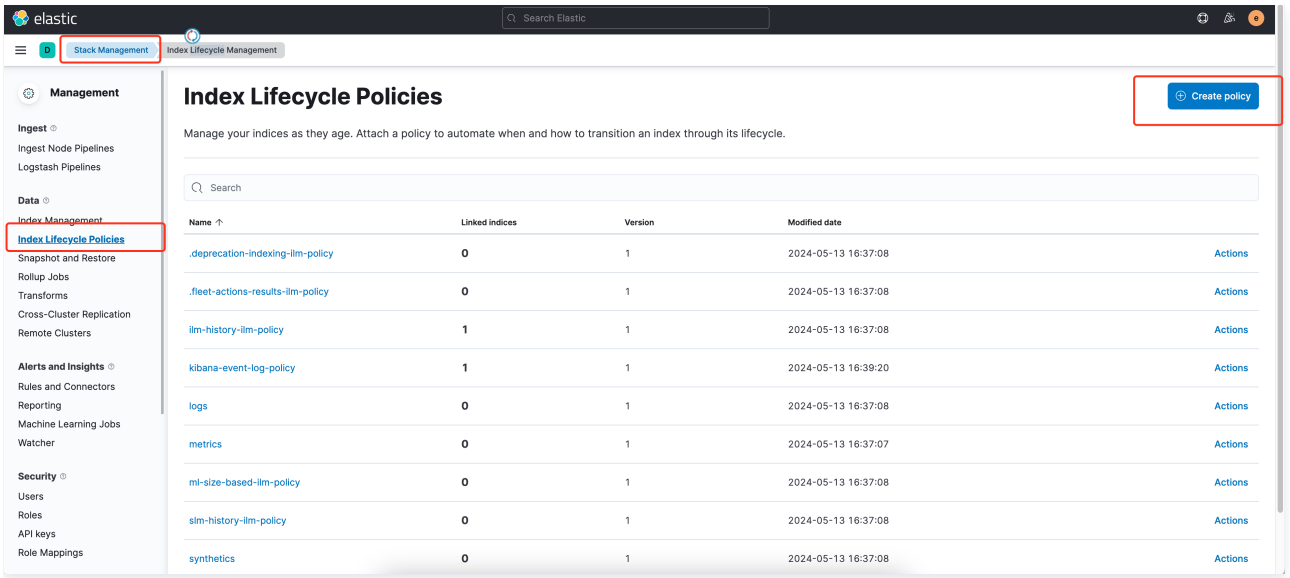
```
PUT hot_warm_test_index/_settings
{
 "index.routing.allocation.require.temperature": "warm"
}
//The command to change the index to a warm attribute in version 7.14.2 is as follows:
PUT hot_warm_test_index/_settings
{
 "index.routing.allocation.include._tier_preference": "data_warm"
}
```

Check shard allocation, shards are allocated to warm nodes.

```
GET _cat/shards/hot_warm_test_index?v&h=index,shard,prirep,node&s=node
index          shard prirep node
hot_data_index 1     p      node3
hot_data_index 0     r      node3
hot_data_index 2     r      node3
hot_data_index 0     p      node5
hot_data_index 2     p      node5
hot_data_index 1     r      node5
```

Index Lifecycle Management

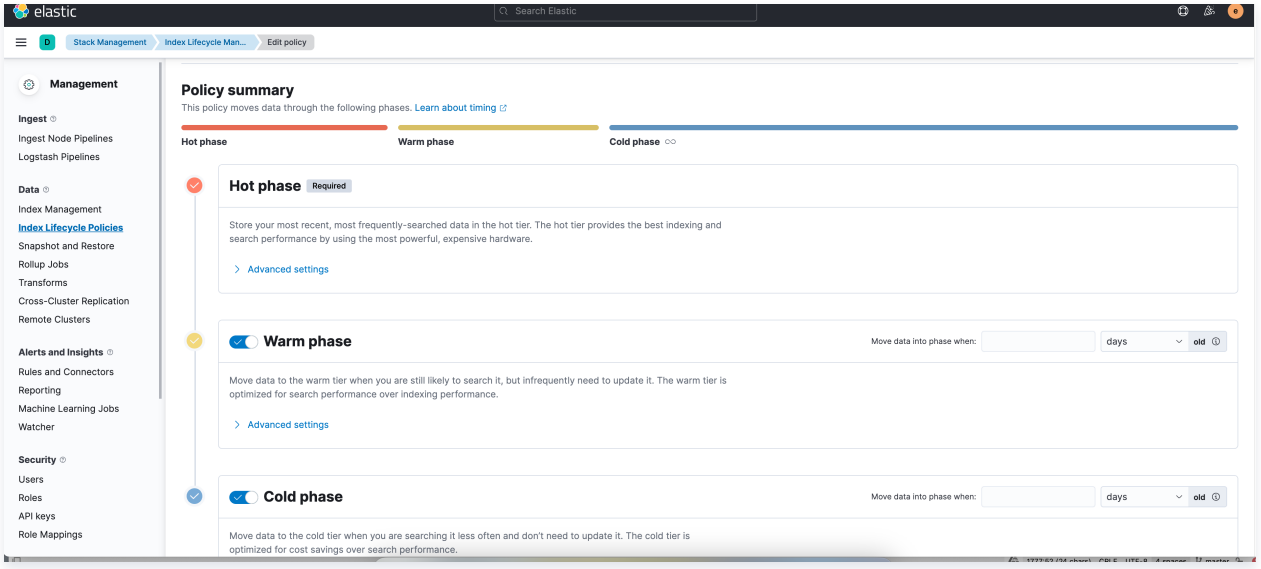
Tencent Cloud currently offers version 6.8.2 of the cluster. This version of Elasticsearch (>=6.6) provides the Index Lifecycle Management feature. Index Lifecycle Management can be configured via API or the Kibana interface. For details, please refer to index-lifecycle-management. The following text will demonstrate how to use Index Lifecycle Management through the Kibana interface, in combination with the Warm and Cold Separation Architecture, to achieve dynamic management of index data.
The position of Index Lifecycle Management in Kibana is shown below (version 7.14.2):



click **Create policy** to enter the configuration interface. The lifecycle of an index includes `Hot phase`, `Warm phase`, and `Cold phase`.

- Hot phase: In this phase, the rollover API can be called based on the number of documents, size, or duration of the index. For details, please refer to indices-rollover-index. As it is not very relevant to this article, it will not be elaborated here.
- Warm phase: When an index is rolled over from the `Hot phase`, it enters the `Warm phase`. In this phase, the index is set to read-only. Users can set the attributes to be used for this index, such as choosing the `temperature: warm` attribute for the Warm and Cold Separation strategy. Additionally, the index can be forceMerged, shrunk, etc. For these operations, please refer to the shrink API and the force merge official documentation.

  Search for the required CAM policy as needed, and click to complete policy association.



- Cold phase: You can set an index to enter the cold phase after it rolls over for a certain period. In this phase, an attribute can also be set. From the Warm and Cold Separation Architecture, it is evident that the warm and cold attributes are extensible. Not only can hot and warm be specified, but also hot, warm, cold, freeze, and more can be added. If you want to use a three-tiered warm and cold separation, you can specify `temperature: cold`. The index freeze operation is also supported. For more details, please refer to the freeze API official documentation.

SQL Support

Last updated: 2024-10-25 09:30:41

Tencent Cloud Elasticsearch supports using SQL instead of DSL as the query language. For those engaged in Product Operation, data analysis, and new ES users, using SQL for queries can reduce their learning curve for getting started with ES.
ES provides two SQL parsers. All open-source versions of ES come pre-installed with the SQL parsing plugin provided by the open-source community. ES 6.4.3 and above (on both Basic Edition and Platinum Edition) supports the native SQL parser of Elasticsearch.

Native SQL Parser

You can use the SQL API for simple queries.

```
POST /_xpack/sql?format=txt
{
    "query": "SELECT * FROM my_index"
}
```

Note:
When using, replace my_index in the above example with the existing index.

For more APIs and usage methods of the native SQL parser, please refer to the official documentation.

Open-Source SQL Parsing Plugins

- v7.5.1 and above:

```
POST /_nlpcn/sql
{
      "sql":"select * from test_index"
}
```

- Other versions:

```
POST /_sql
{
      "sql":"select * from test_index"
}
```

Note:
During use, replace test_index in the above example with an existing index.

Usage: For more APIs and usage methods of SQL plugins, see documentation.

SQL JDBC Access

Access to ES clusters through JDBC is supported in the Platinum Edition of ES 6.4.3 and above. You need to download the JDBC driver first by clicking download, or by pasting the following contents into the Maven pom file to add dependencies and download:

```
<dependency>
  <groupId>org.elasticsearch.plugin</groupId>
  <artifactId>x-pack-sql-jdbc</artifactId>
  <version>6.4.3</version>
</dependency>
```

Sample code for SQL JDBC access:

```
import java.sql.*;
```

```
import java.util.Properties;

public class Main {

    public static void main(String[] args) {
        try {
            Class.forName("org.elasticsearch.xpack.sql.jdbc.jdbc.JdbcDriver");
        } catch (ClassNotFoundException e) {
            e.printStackTrace();
            return;
        }
        String address = "jdbc:es://http://YOUR_ES_VIP:9200";
        Properties properties = new Properties();
        properties.put("user", "elastic");
        properties.put("password", "YOUR_PASS");

        Connection connection = null;
        try {
            connection = DriverManager.getConnection(address, properties);
            Statement statement = connection.createStatement();
            ResultSet results = statement.executeQuery("select FlightNum from kibana_sample_data_flights
limit 10");
            while (results.next()) {
                System.out.println(results.getString(1));
            }
        } catch (Exception e) {
            e.printStackTrace();
        } finally {
            try {
                if (connection != null && !connection.isClosed()) {
                    connection.close();
                }
            } catch (SQLException e) {
                e.printStackTrace();
            }
        }
    }
}
```

Receiving Watcher Alerts via WeCom Bot

Last updated: 2024-10-25 09:31:01

Tencent Cloud's ES Platinum version supports the X-Pack Watcher feature. By adding triggers, operations, and other configurations, specific actions can be executed when conditions are met. For example, when an error log is detected in the index, an alarm can be automatically sent. This article introduces how to configure an Enterprise WeChat robot to receive alarms sent by Watcher.

Note

- The X-Pack Watcher feature is only available in the platinum version.
- Due to Tencent Cloud ES network architecture adjustments, only instances created in June 2020 and later support configuring an Enterprise WeChat robot to receive Watcher alarms.

Background

A Watcher consists of four parts, as follows:

- Trigger: Defines when a Watcher starts executing and must be set when configuring a Watcher. For details on supported triggers, see Schedule Trigger.
- Input: Query conditions executed on the monitored index. When the Watcher is triggered, input loads the data into the execution context. This context can be accessed in subsequent Watcher execution phases. For details, please see Inputs.
- Condition: Conditions that must be met to execute actions.
- Actions: Specific operations executed when conditions occur. For example, the Webhook Action introduced in this article.

Directions

1. Prepare a CVM within the same VPC as the ES cluster that can access the Webhook address (e.g., via the internet).
2. Install Nginx on the CVM. Refer to Nginx Installation for detailed instructions.
3. Configure Nginx proxy forwarding. Replace the Server section in the nginx.conf file with the following configuration.

- The default port for the Nginx service is 80. If you need to change this port, log in to the console security groups to allow traffic through this port.
- <Enterprise WeChat Bot Webhook Address>: Replace with the Webhook address of the Enterprise WeChat bot that will receive alert messages.

```
server {
        listen       80;
        server_name  localhost;
        index index.html index.htm index.php;
        root /usr/local/nginx/html;
        #charset koi8-r;

        #access_log  logs/host.access.log  main;
        location ~ .*\.(php|php5)?$
        {
            fastcgi_pass 127.0.0.1:9000;
            fastcgi_index index.php;
            include fastcgi.conf;
        }
        location ~ .*\.(gif|jpg|jpeg|png|bmp|swf|ico)$
        {
            expires 30d;
        # access_log off;
        }
        location / {
            proxy_pass <Enterprise WeChat robot's wehbook address>;
        }
        location ~ .*\.(js|css)?$
        {
            expires 15d;
```

```
            # access_log off;
        }
        access_log off;

        #error_page  404              /404.html;

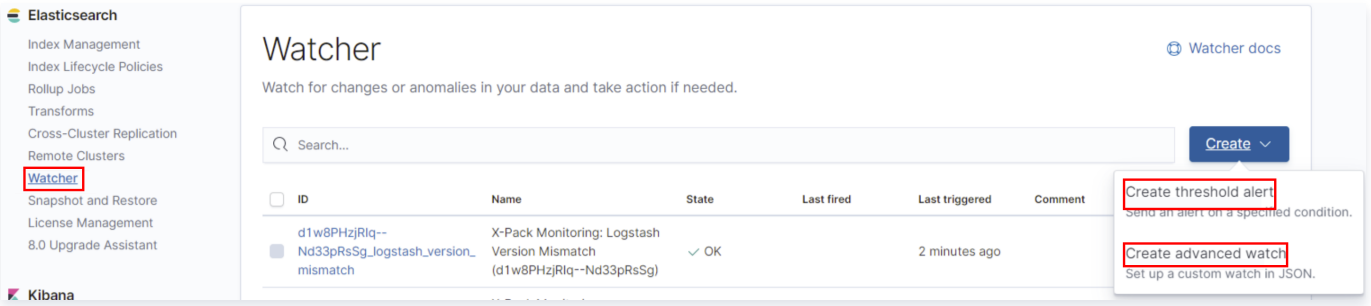
        # redirect server error pages to the static page /50x.html

        error_page   500 502 503 504  /50x.html;
        location = /50x.html {
                root   html;
        }
}
```

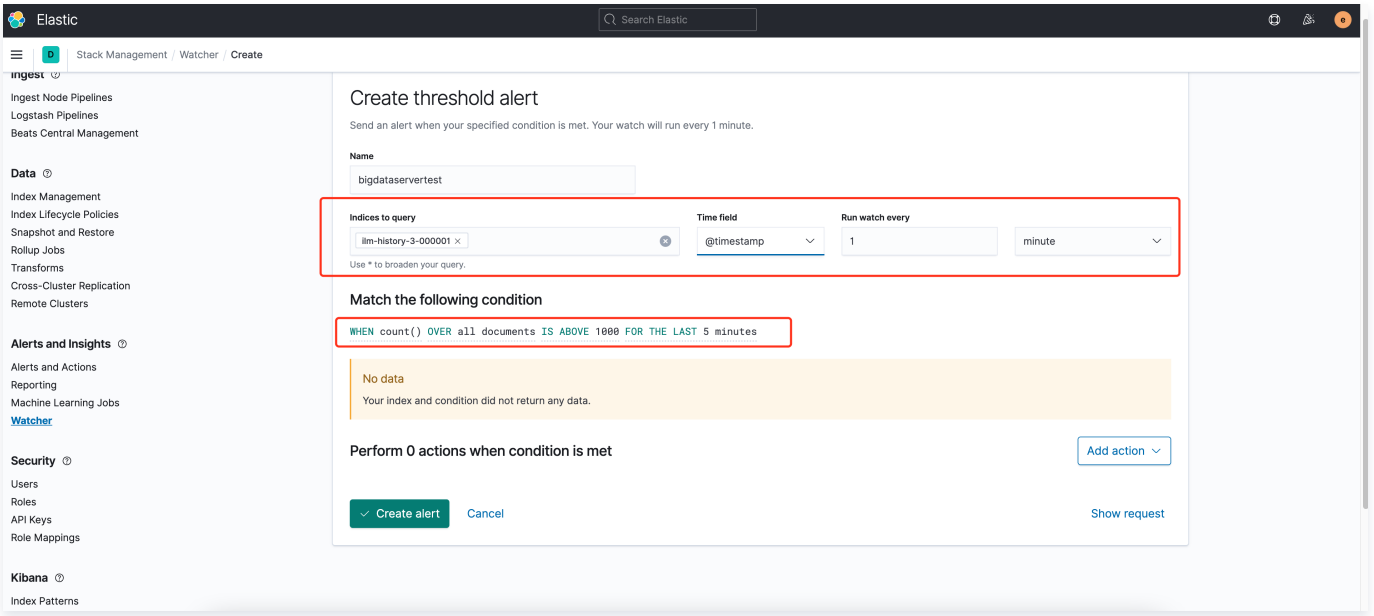
4. Load the modified configuration file and restart Nginx.

```
/usr/local/webserver/nginx/sbin/nginx -s reload
/usr/local/webserver/nginx/sbin/nginx -s reopen
```

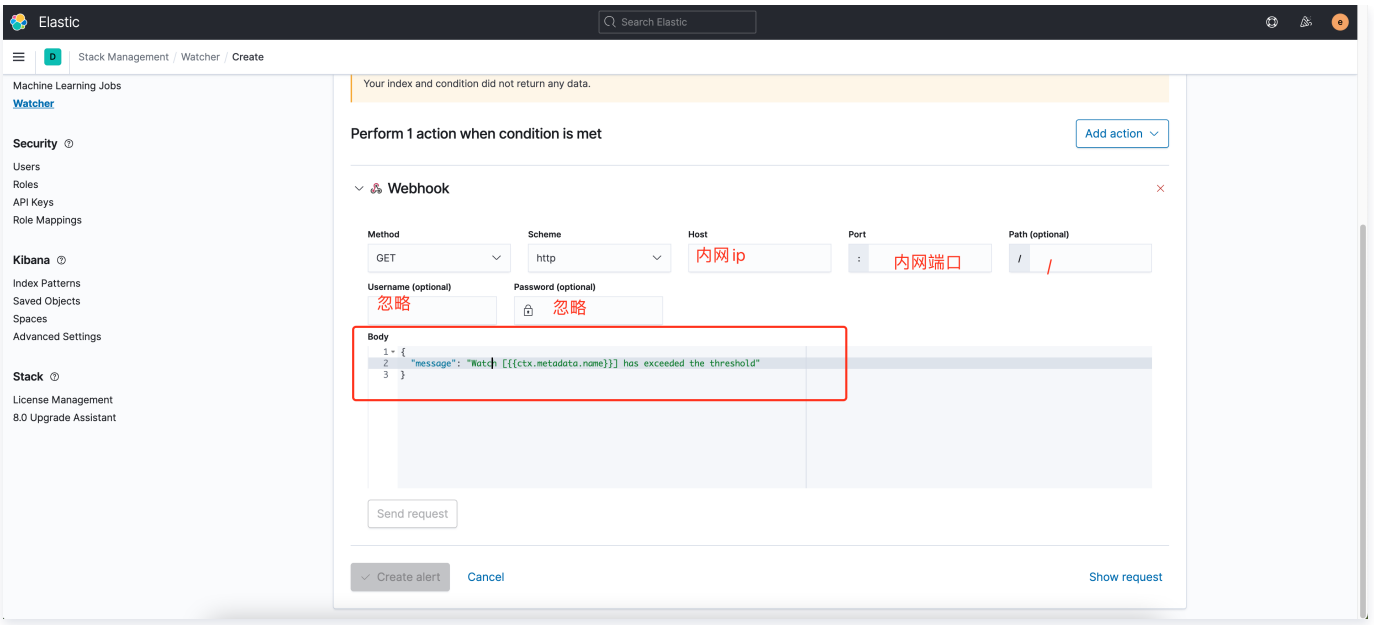
5. Configure the Watcher alert rules.**This step can be graphically performed in the Kibana interface under Management > Watcher options.**



- `Create threshold alert` Configure threshold alerts through the interface. You can monitor and alert for specific conditions of an index, such as CPU utilization, number of documents, etc. You can make more detailed settings in the Condition section below. See the following for reference:



- Click the upper right corner**Add action**, select "Webhook", and configure as follows:



- Click **Send request** to test, then click **Create alert**.
- `Create advanced watch` Set various Watcher parameters through the API. For API details, please refer to PUT Watch.

6. After completing the above steps, you will receive alert information from the robot in the enterprise WeChat group you created.

Best Practices for HTTPS Cluster Access Communication

Last updated: 2024-11-20 17:26:43

HTTPS is a secure version of HTTP that ensures data transmission security through mechanisms like transmission encryption and identity authentication. This article demonstrates how to use clients such as Beats, Logstash, Kibana, and Java Client to access an HTTPS-enabled ES cluster based on the Tencent Cloud ES cluster environment.

HTTPS Cluster Environment Preparation

Create an HTTPS Protocol Cluster

First, we create an HTTPS cluster in the Tencent Cloud ES console by selecting HTTPS protocol on the purchase page. Currently, this feature is supported through an allowlist. You can submit a ticket to apply for access.

ES enables the HTTPS protocol by setting specific parameters in the elasticsearch.yml configuration file, as shown below:

```
xpack.security.http.ssl.enabled: true
xpack.security.http.ssl.keystore.path: certs/ces-certificates.p12
xpack.security.http.ssl.truststore.path: certs/ces-certificates.p12
```

Obtain pem Certificate File

Once the cluster is successfully created, you can export the relevant certificates from the cluster access control module in the ES console access control page.

Search for the required CAM policy as needed, and click to complete policy association.

Tencent Cloud ES provides the following two certificate files:

File Name	Purpose
client-certificates.pem	For Beats, Logstash, and Java Client to connect to the ES cluster
server-certificates.pem	For Kibana to connect to the ES cluster

The configuration methods for connecting clients such as Beats, Logstash, Kibana, and Java to the HTTPS cluster are introduced in detail below.

Beats Output to HTTPS Cluster

CVM Metricbeat Output to ES

1. First, we create a CVM in the Tencent Cloud CVM Console under the same VPC as the ES cluster. After creation, upload the obtained pem authentication file to the CVM. The storage path here is: /usr/local/service/https-certs. Then go to the Tencent Cloud ES Console Beats Management Page and create a Metricbeat.

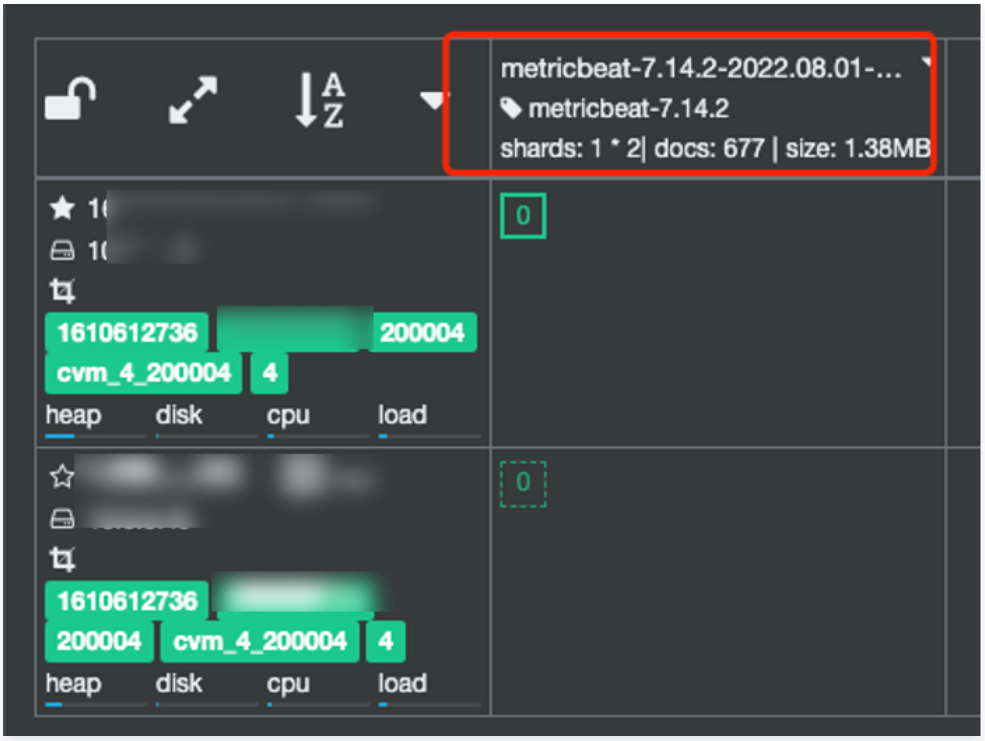
2. The most crucial step is to configure the following in the metricbeat.yml configuration file.

```
output.elasticsearch:
    hosts: ["https://ES-VIP:9200"]
    username: "elastic"
    password: "changeme"
    ssl.certificate_authorities: ["/usr/local/service/https-certs/client-certificates.pem"]
    ssl.verification_mode: certificate
```

Configuration description:

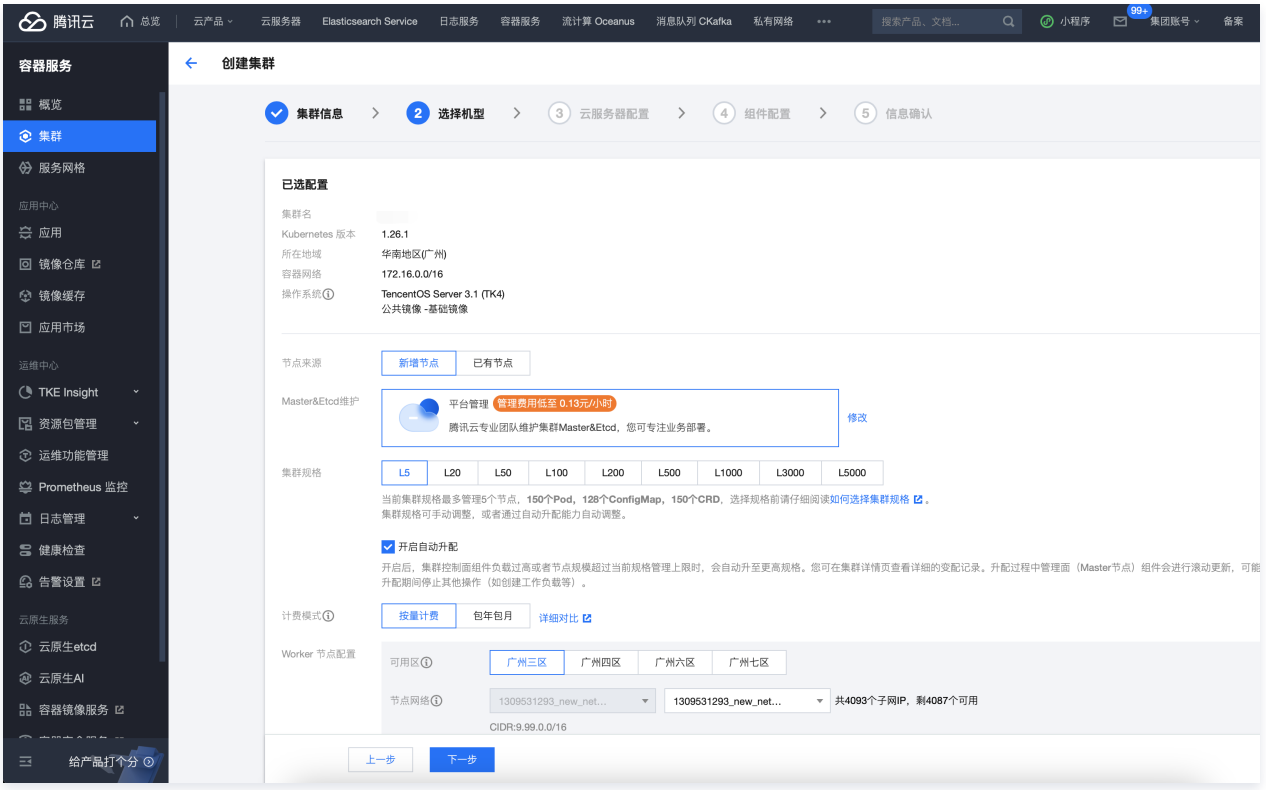
Configuration Item	Description
hosts	VIP of the ES cluster, such as `https://10.0.X.29:9200`, starting with https
username/password	Username and password of the ES cluster
ssl.certificate_authorities	The pem authentication certificate file path required to connect to the HTTPS cluster
ssl.verification_mode	Server certificate authentication mode. There are four modes: full, strict, certificate, and none. Here, we use the certificate mode, which authenticates only the CA certificate, not the hostname information. For more details, refer to the official documentation.

3. After configuration, you can see an automatically created index starting with metricbeat-7.14.20-* in Cerebro or Kibana of the ES cluster. This completes the configuration for Metricbeat in the CVM to connect to the HTTPS ES cluster.



TKE Filebeat Log Collector outputs to ES

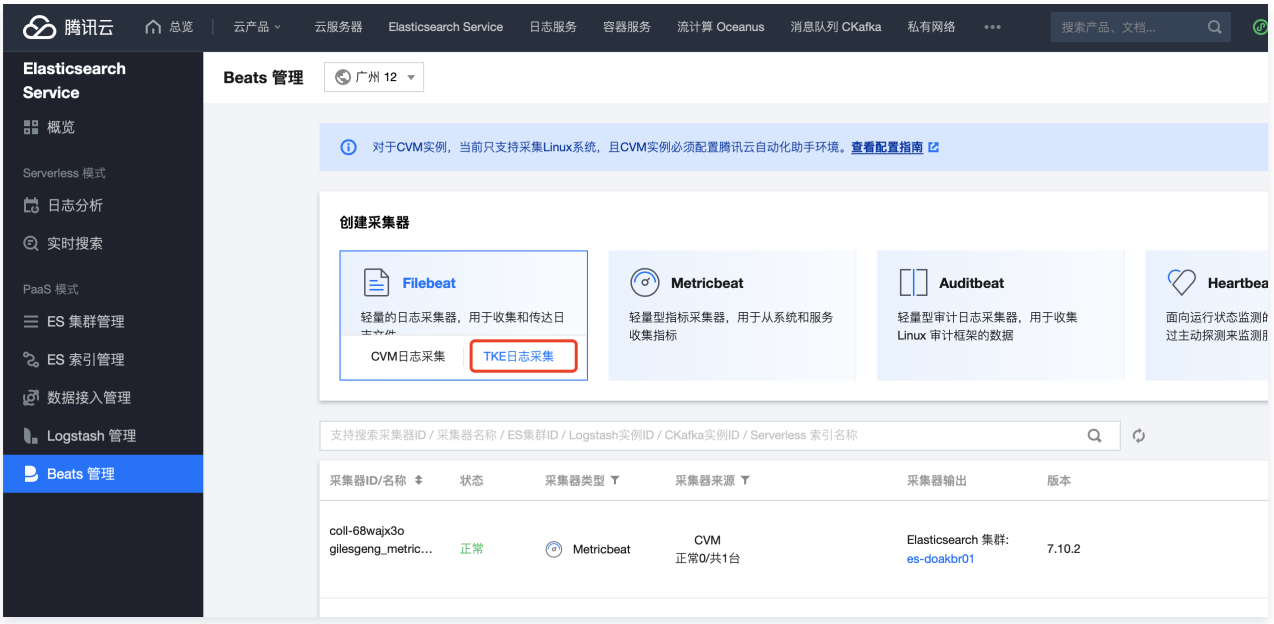
1. The TKE Filebeat log collector outputs to the HTTPS ES Cluster following the same process as the Metricbeat output on CVM. First, we upload the PEM file to the CVM node where the Worker is automatically created when the TKE cluster is created, such as the /var/log/https-certs directory.



2. Location of the client-certificates.PEM file:

```
-rw-r--r-- 1 root root 1338 Aug  1 21:12 client-certificates.pem
[root@VM-3-30-centos https-certs]# pwd
/var/log/https-certs
[root@VM-3-30-centos https-certs]#
```

3. Then, we create a new Filebeat collector in the Tencent Cloud ES console, selecting TKE log collection:

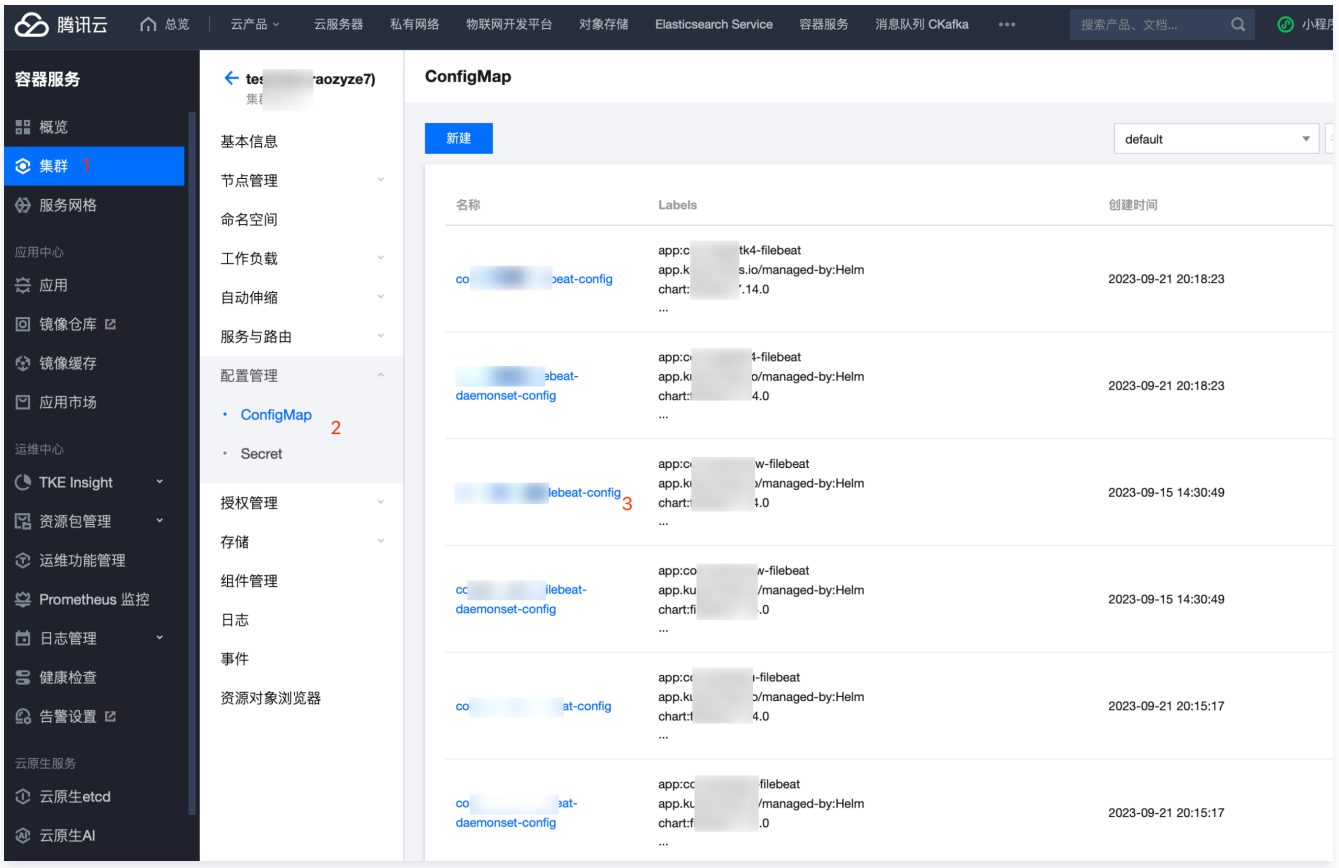


4. Next, configure the basic information of the Filebeat collector, choose the version number, and output to the ES Cluster. As shown in the figure below:

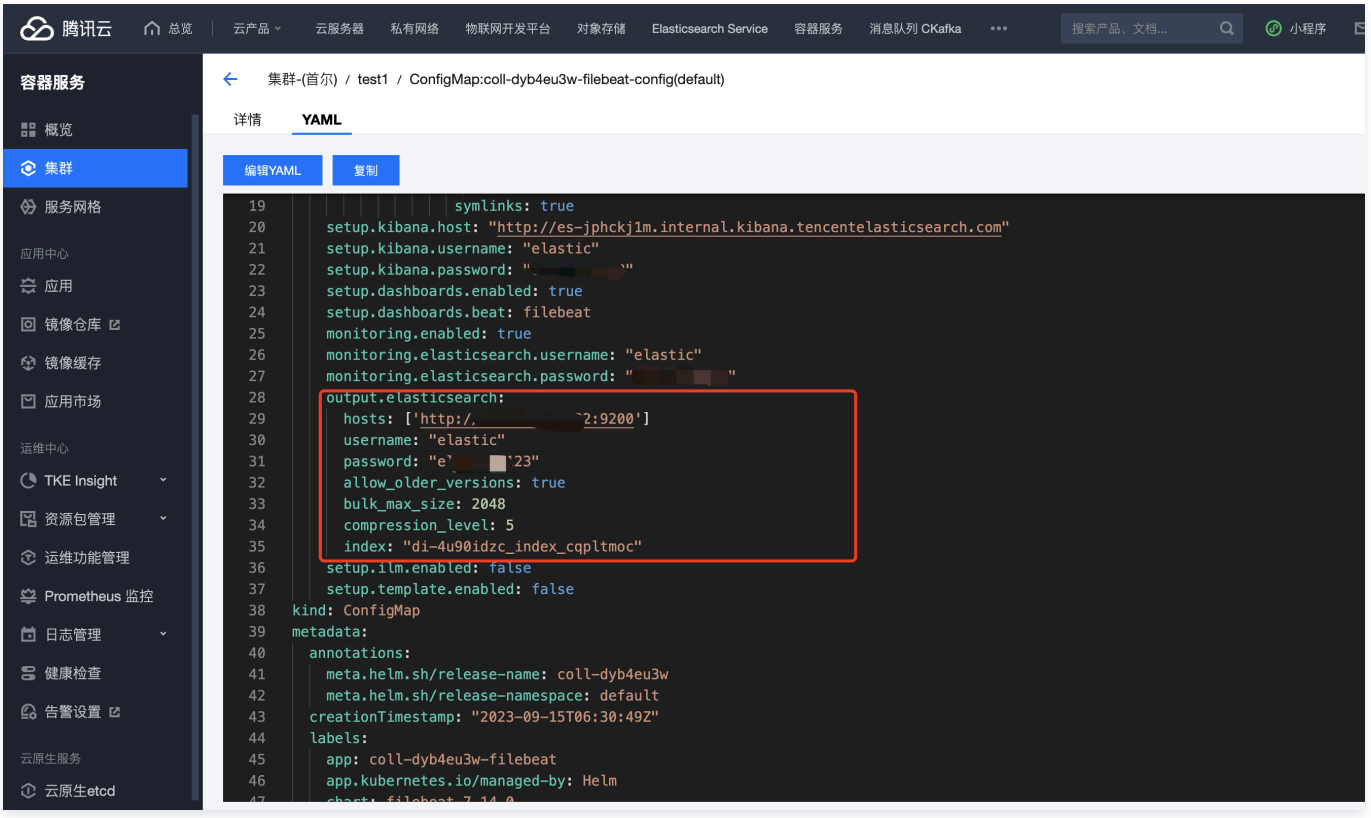
5. Start configuring the collection source, as shown in the figure below:

6. After creating the Filebeat TKE container log collector, go to the TKE cluster's details page, find the ConfigMap in configuration management, and locate the corresponding beats config file:

Search for the required CAM policy as needed, and click to complete policy association.



7. Click **Edit YAML**, then modify the output.elasticsearch configuration item, as shown in the figure below:

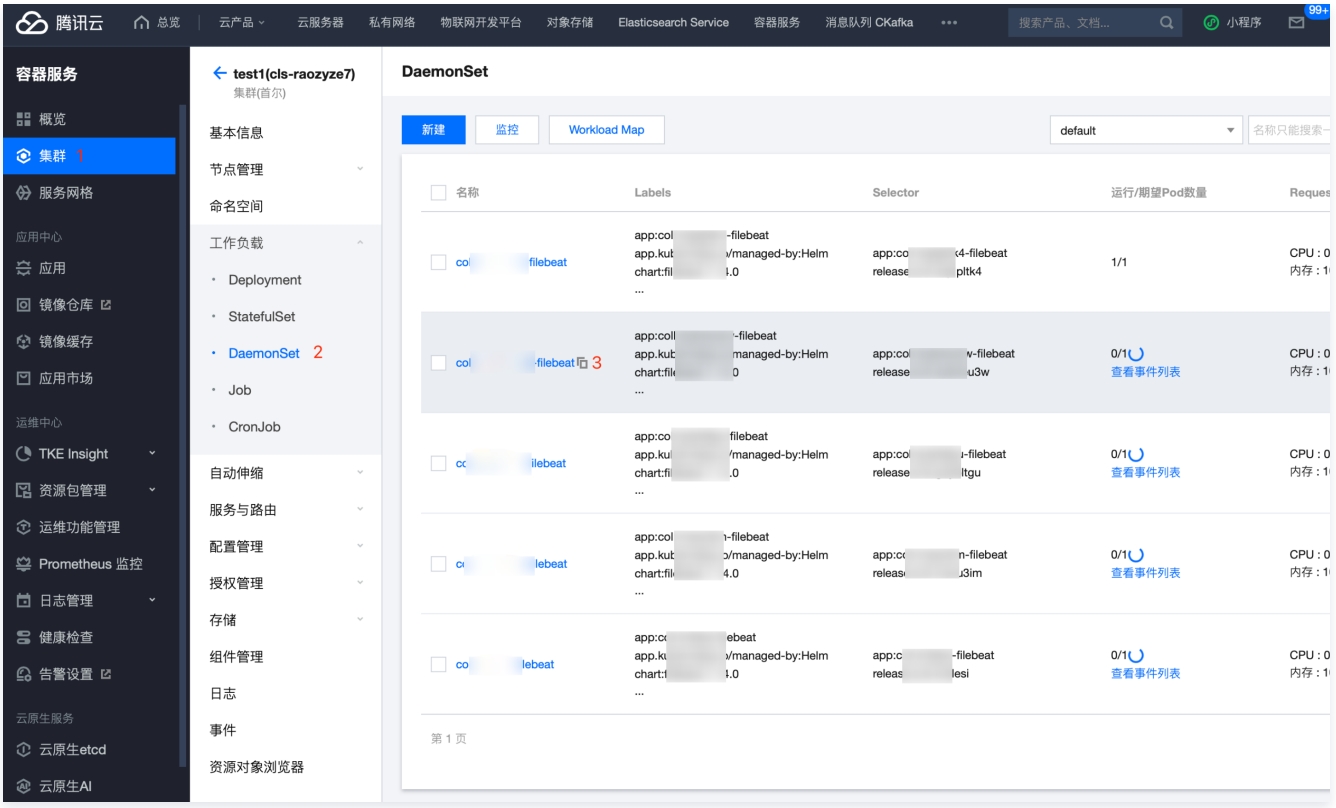


8. The configuration details are as follows, and the explanation of the configuration is the same as for Metricbeat.

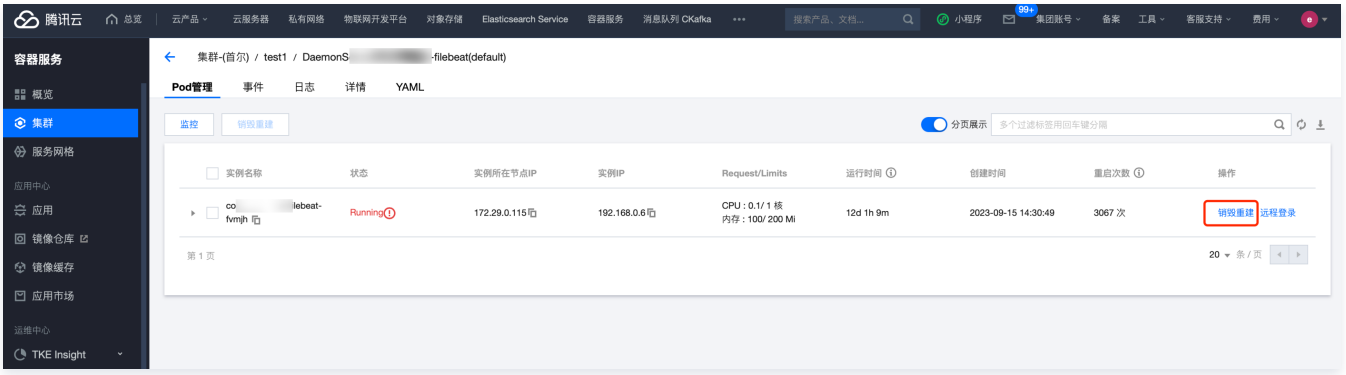
```
output.elasticsearch:
      hosts: ['https://ES-VIP:9200']
```

```
    username: "elastic"
    ssl.certificate_authorities: ["/var/log/https-certs/client-certificates.pem"]
    ssl.verification_mode: "certificate"
    password: "changeme"
    indices:
      - index: "filebeat-tke-%{+yyyy.MM.dd}"
        when.equals:
          tke_collector_target_name: "logs_to_https_es"
```

9. Then, in the workloads of the TKE cluster, find the DaemonSet, then go to the corresponding beats DaemonSet, click in, and destroy and recreate its pod.

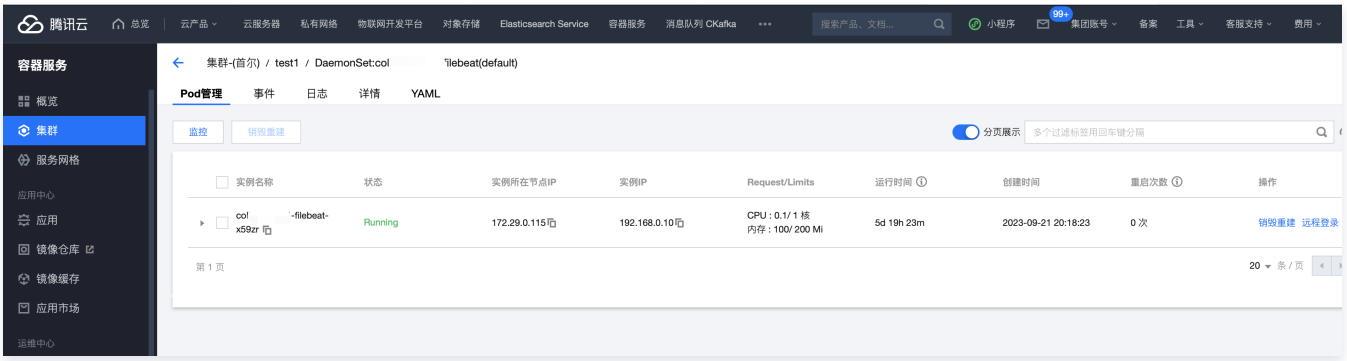


10. Go to the Pod management page, select the corresponding Pod for destruction and recreation.

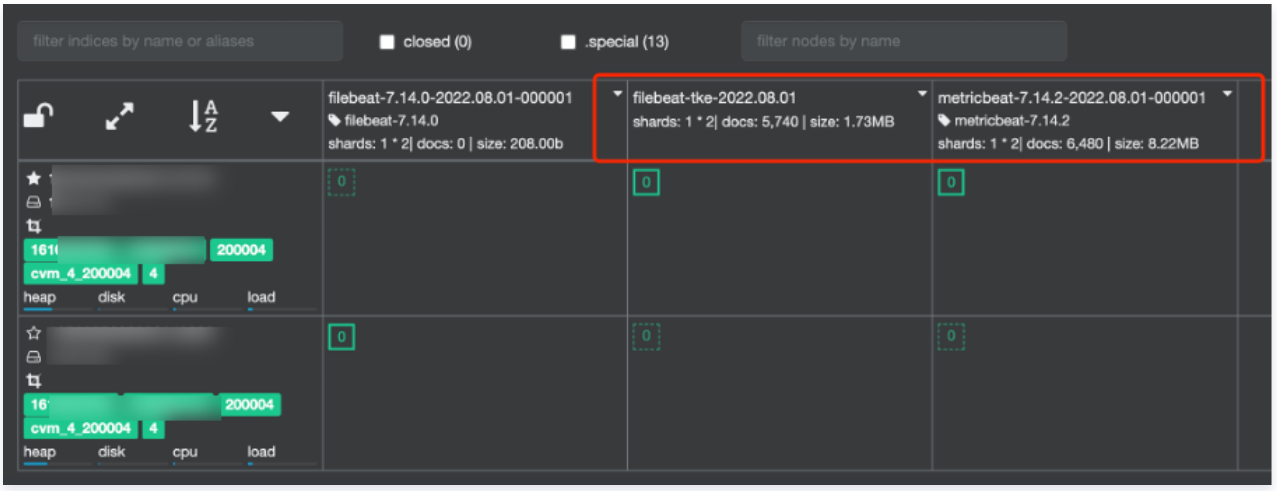


11. After destruction and recreation, the Pod status will become Running, as shown in the figure below.

> **Note**
> If the pod is destroyed and recreated multiple times and remains in the red Running state, there might be an issue with the beats yml configuration. You can check the logs for specific issues or recheck the yml configuration for errors.

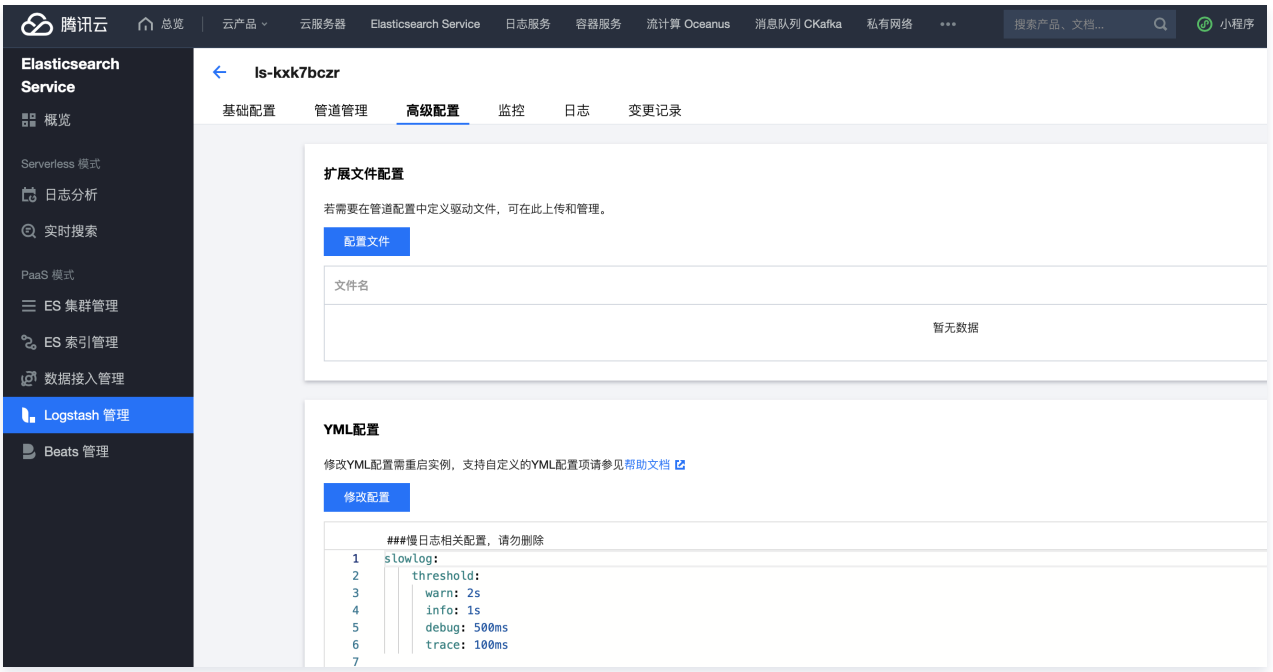


12. Then, in the ES cluster, you will see an automatically created index prefixed with filebeat-tke-*. This indicates that the TKE logs have been successfully output to the HTTPS cluster.



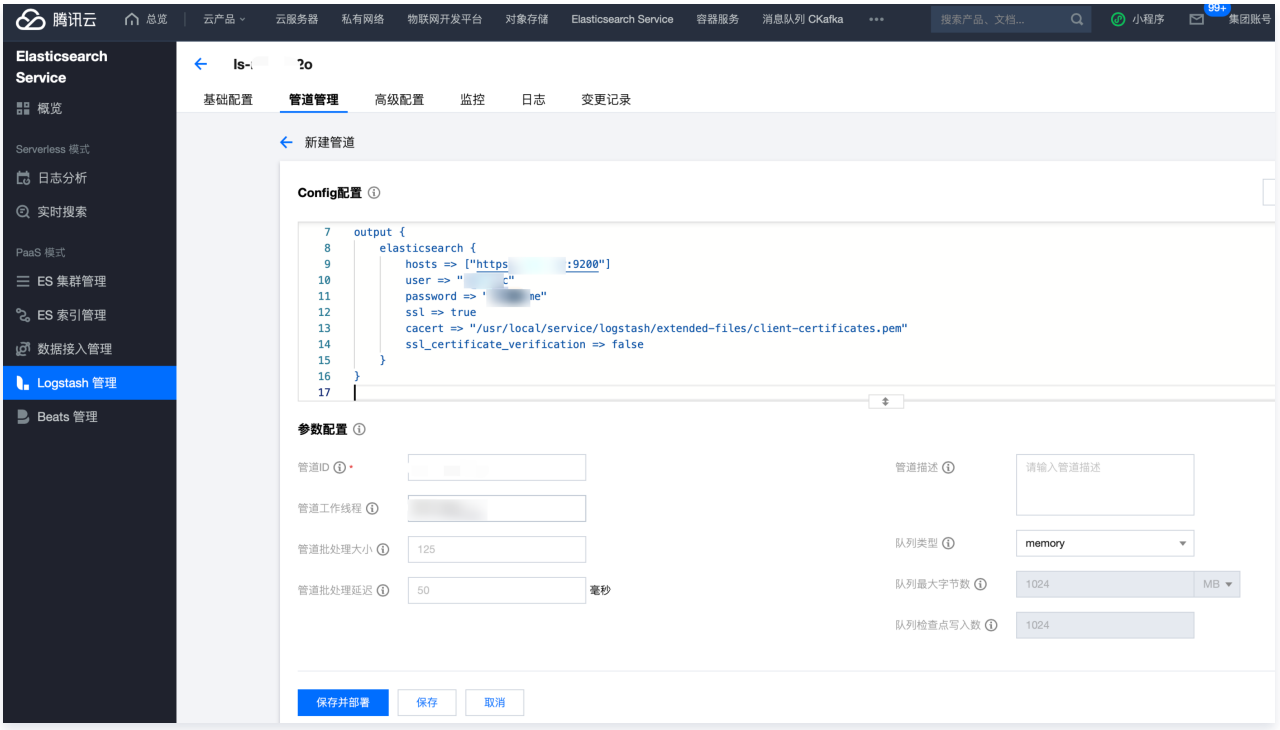
Logstash Output to HTTPS Cluster

1. Tencent Cloud Logstash is a fully managed product. Therefore, we first need to upload the PEM file as an extended file in the Tencent Cloud Logstash console, as shown below.



2. Enter the **Logstash Management** > **Advanced Settings** interface, and click the **Local Upload** button.

3. Then, in the Pipeline Management Tab, click the **Create Pipeline** button to create a new pipeline. In the Config configuration edit box of the pipeline, configure the ES connection information and certificate path.



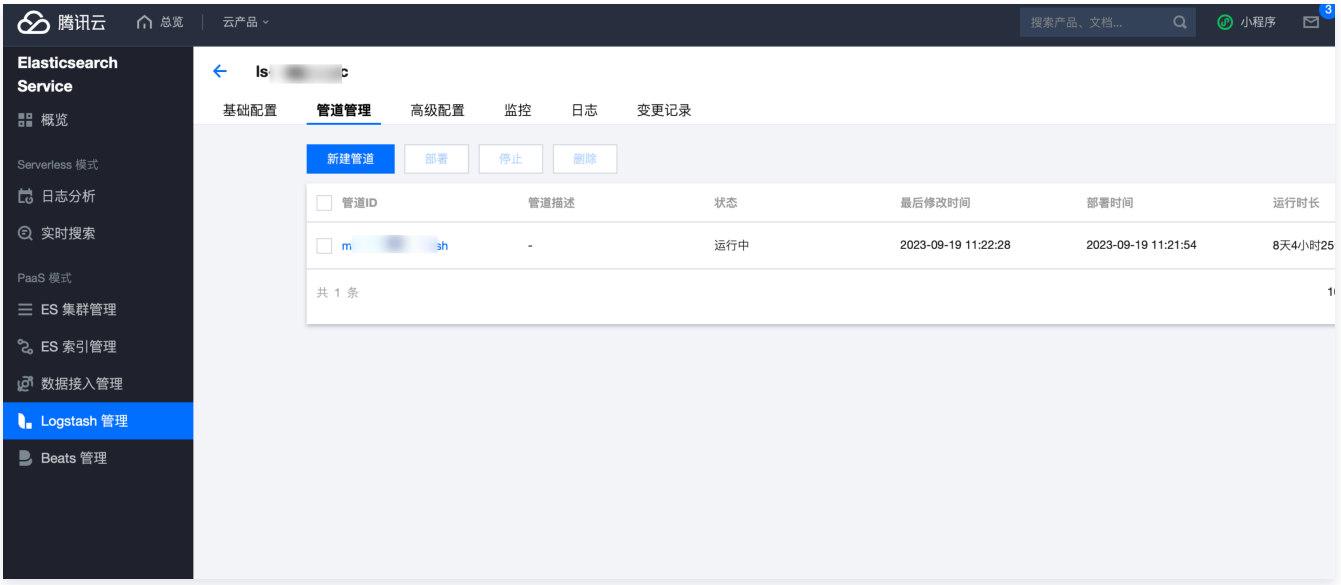
4. The detailed pipeline configuration text information is as follows:

```
output {
    elasticsearch {
        hosts => ["https://ES-VIP:9200"]
        user => "elastic"
        password => "changeme"
```

```
        ssl => true
        cacert => "/usr/local/service/logstash/extended-files/client-certificates.pem"
        ssl_certificate_verification => false
    }
}
```

- The cacert here refers to the path of the PEM file we uploaded via the extended file method. The fixed path is: `/usr/local/service/logstash/extended-files/client-certificates.pem`.
- Other configuration instructions are consistent with Metricbeat.

5. Click **Save** and deploy the pipeline. You can view the newly created pipeline information in the pipeline list, and its status will be Running.



6. At this point, we can go to the HTTPS cluster and see data being written from the input cluster.

Kibana connecting to the HTTPS cluster

1. Tencent Cloud ES cluster comes with built-in Kibana access capability by default, so generally, customers do not need to configure Kibana. Here is the configuration method for connecting a self-built Kibana to the HTTPS cluster. Unlike the client authentication certificate used by Beats, Logstash, etc., Kibana uses server-certificates.pem. The command to generate the certificate on the Tencent Cloud ES side in CVM is as follows:

```
openssl pkcs12 -in ces-certificates.p12 -password pass:xxxxxx -nokeys -cacerts -out server-certificates.pem
```

2. Since we have already obtained the pem certificate file, we will copy the server-certificates.pem file to the following path on the Kibana node: `/usr/local/service/https-certs`. Then modify the kibana.yml configuration file as follows:

```
elasticsearch.hosts: ["https://ES-VIP:9200"]
elasticsearch.username: "elastic"
elasticsearch.password: "changeme"
elasticsearch.ssl.verificationMode: certificate
elasticsearch.ssl.certificateAuthorities: ["/usr/local/service/https-certs/server-certificates.pem"]
xpack.encryptedSavedObjects.encryptionKey: "dfed624ca4014135f61804440536xxxx"
xpack.fleet.registryUrl: "https://epr.elastic.co"
```

3. Description of Configuration Items:

- elasticsearch.ssl.certificateAuthorities: The path to store the pem file.
- elasticsearch.ssl.verificationMode: Certificate verification mode, 'certificate' adopts the mode of only verifying the CA certificate, not the hostname.
- xpack.fleet.registryUrl: (Optional configuration) The public network repository that Fleet integration module needs to access. Kibana node needs to have public network access capability here.

- xpack.encryptedSavedObjects.encryptionKey: The value can be obtained by executing the `bin/kibana-encryption-keys generate` command on Kibana.

```
[root@VM-48-212-centos /usr/local/service/kibana]# bin/kibana-encryption-keys generate
## Kibana Encryption Key Generation Utility

The 'generate' command guides you through the process of setting encryption keys for:

xpack.encryptedSavedObjects.encryptionKey
    Used to encrypt stored objects such as dashboards and visualizations
    https://www.elastic.co/guide/en/kibana/current/xpack-security-secure-saved-objects.html#xpack-security-secure-saved-o

xpack.reporting.encryptionKey
    Used to encrypt saved reports
    https://www.elastic.co/guide/en/kibana/current/reporting-settings-kb.html#general-reporting-settings

xpack.security.encryptionKey
    Used to encrypt session information
    https://www.elastic.co/guide/en/kibana/current/security-settings-kb.html#security-session-and-cookie-settings


Already defined settings are ignored and can be regenerated using the --force flag.  Check the documentation links for in
.
Definitions should be set in the kibana.yml used configure Kibana.

Settings:
xpack.encryptedSavedObjects.encryptionKey: 1fb56afd8128f4da65ee056b839c7b9f
xpack.security.encryptionKey: 16027cc53a6f269857251b1d4720cea5

[root@VM-48-212-centos /usr/local/service/kibana]#
```

4. After restarting Kibana, you can normally access the HTTPS ES cluster.

Java Client connecting to the HTTPS cluster

1. This document demonstrates the configuration methods for accessing an HTTPS cluster in a Spring project. First, you need to add the following Maven dependencies:

```
        <dependency>
            <groupId>org.elasticsearch.client</groupId>
            <artifactId>elasticsearch-rest-client</artifactId>
            <version>7.10.1</version>
        </dependency>
        <dependency>
            <groupId>org.elasticsearch.client</groupId>
            <artifactId>elasticsearch-rest-high-level-client</artifactId>
            <version>7.10.1</version>
        </dependency>
        <dependency>
            <groupId>org.elasticsearch</groupId>
            <artifactId>elasticsearch</artifactId>
            <version>7.10.1</version>
        </dependency>
```

2. Next, we configure the Elasticsearch connection in the ElasticsearchConfig custom definition class.

```
@Configuration
public class ElasticsearchConfig {
    @Value("${es.username}")
    private String userName;
    @Value("${es.password}")
    private String password;
    @Value("${es.scheme}")
    private String scheme;
    @Value("${es.domain}")
    private String domain;
    @Value("${es.https.cert}")
    private String certFile;
    @Value("${es.port}")
    private int port;
    private RestHighLevelClient client;
```

```
    @Bean
    public RestHighLevelClient EsConnectInit() throws CertificateException, IOException, KeyStoreException,
NoSuchAlgorithmException, KeyManagementException {
        final CredentialsProvider credentialsProvider = new BasicCredentialsProvider();
        credentialsProvider.setCredentials(AuthScope.ANY,new
UsernamePasswordCredentials(userName,password));
        RestClientBuilder builder;
        if (scheme.equals("https")) {
            Path caCertificatePath = Paths.get(certFile);
            System.out.println(certFile);
            CertificateFactory factory = CertificateFactory.getInstance("X.509");
            Certificate trustedCa;
            try(InputStream is = Files.newInputStream(caCertificatePath)) {
                trustedCa = factory.generateCertificate(is);
            }
            KeyStore trustStore = KeyStore.getInstance("pkcs12");
            trustStore.load(null,null);
            trustStore.setCertificateEntry("ca",trustedCa);
            SSLContextBuilder sslContextBuilder = SSLContexts.custom().loadTrustMaterial(trustStore, new
TrustStrategy() {
                @Override
                public boolean isTrusted(X509Certificate[] x509Certificates, String s) throws
CertificateException {
                    return true;
                }
            });
            final SSLContext sslContext = sslContextBuilder.build();
            final HostnameVerifier hostnameVerifier = new HostnameVerifier() {
                public boolean verify(String hostname, SSLSession session) {
                    return true;
                }
            };
            builder = RestClient.builder(new HttpHost(domain, port, scheme)).setRequestConfigCallback(new
RestClientBuilder.RequestConfigCallback() {
                @Override
                public RequestConfig.Builder customizeRequestConfig(RequestConfig.Builder
requestConfigBuilder) {
                    requestConfigBuilder.setConnectTimeout(-1);
                    requestConfigBuilder.setSocketTimeout(-1);
                    requestConfigBuilder.setConnectionRequestTimeout(-1);
                    return requestConfigBuilder;
                }
            }).setHttpClientConfigCallback(new RestClientBuilder.HttpClientConfigCallback() {
                @Override
                public HttpAsyncClientBuilder customizeHttpClient(HttpAsyncClientBuilder httpClientBuilder)
{
                    httpClientBuilder.disableAuthCaching();
                    httpClientBuilder.setSSLHostnameVerifier(hostnameVerifier);
                    return httpClientBuilder.setSSLContext(sslContext)
                            .setDefaultCredentialsProvider(credentialsProvider);

                }
            });
        } else {
             builder = RestClient.builder(new HttpHost(domain, port, scheme))
                    .setRequestConfigCallback(new RestClientBuilder.RequestConfigCallback() {
                        @Override
                        public RequestConfig.Builder customizeRequestConfig(RequestConfig.Builder
requestConfigBuilder) {
                            requestConfigBuilder.setConnectTimeout(-1);
                            requestConfigBuilder.setSocketTimeout(-1);
                            requestConfigBuilder.setConnectionRequestTimeout(-1);
```

```
                            return requestConfigBuilder;
                        }
                    }).setHttpClientConfigCallback(new RestClientBuilder.HttpClientConfigCallback() {
                        @Override
                        public HttpAsyncClientBuilder customizeHttpClient(HttpAsyncClientBuilder
httpClientBuilder) {                            httpClientBuilder.disableAuthCaching();
                            return httpClientBuilder.setDefaultCredentialsProvider(credentialsProvider);
                        }
                    });
        }
        client = new RestHighLevelClient(builder);
        return client;
    }
}
```

3. The configuration information of Elasticsearch is as follows:

```
es.scheme=https # Protocol https or http
es.port=9200 # Port of the ES cluster VIP
es.domain=9.10.1.X  # VIP of the ES cluster
es.username=elastic # ES cluster username
es.password=changeme # ES cluster password
es.https.cert=/usr/local/services/certs/client-certificates.pem # PEM authentication certificate file path
required for connecting to an HTTPS cluster
```

With the above configuration, you can successfully connect to the Elasticsearch cluster with HTTPS protocol enabled in a Spring project.

Practice Tutorial for HTTPS Cluster Access and Communication

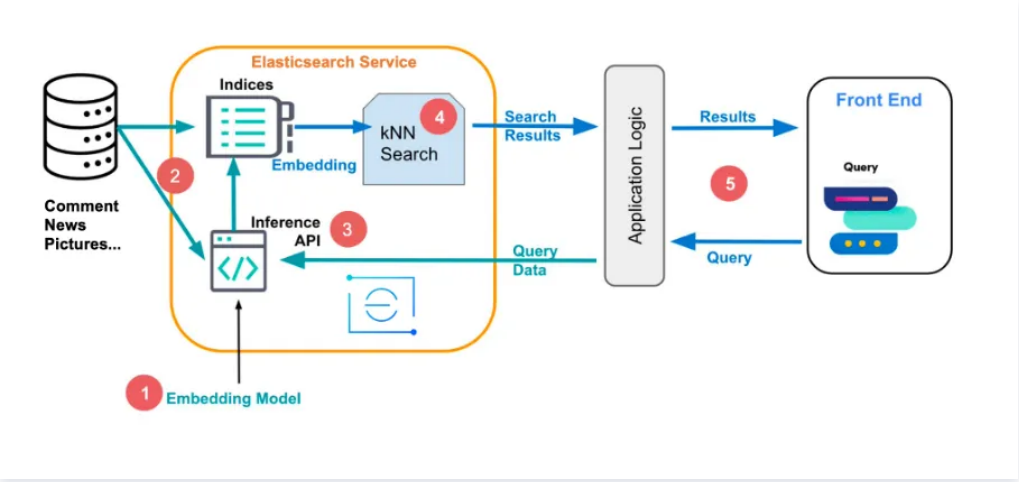
Last updated: 2024-11-20 17:27:56

Foreword

Elasticsearch (ES) is a distributed search engine based on Lucene. It can quickly store, search, and analyze large amounts of data. ES is widely used in various fields such as enterprise search, log analysis, security analysis, business intelligence, and vector retrieval due to its high performance, scalability, and rich features. Tencent Cloud Big Data ES is a fully managed cloud service for the Elastic Stack, integrated with X-Pack features and optimized with Tencent's proprietary enhancements. The newly released ES 8.8.1 version integrates the HNSW algorithm to improve the computation speed for vector retrieval. After our optimizations, it supports billion-scale vector retrieval. This article aims to introduce how to use ES 8.8.1, which combines text search, vector retrieval, and AI capabilities, to bring you a cutting-edge search and analysis experience.

Overall Architecture

ES 8.8.1 provides an end-to-end search and analysis platform from natural language processing to vectorization to vector search, and it can integrate with large models. Compared to ES 7, ES 8.8.1 introduces the HNSW algorithm to create a graph index for vectors and integrates with Lucene's functionalities (inverted index, BKD trees, cache, etc.) to optimize retrieval performance. Therefore, ES 8.8.1's vector search can be combined with any query supported by ES, and it is compatible with aggregation, document-level security, field-level security, index sorting, and provides a good performance experience.



The overall process is summarized as follows:
Vectorized Data: Users can upload pre-trained big data models, such as Word Embeddings or deep learning models (like BERT), allowing ES to perform real-time reading and writing of Embeddings. Businesses can also directly report vector data to ES.
Indexing Vector Data: Store vectorized data in ES. During the indexing process, vector data is distributedly stored across multiple nodes to enhance data reliability and scalability.
Initiating Vector Search Requests: Vector search requests can be initiated using ES's search API. The request must specify the vector to be searched and other relevant parameters, such as the model to use, similarity algorithms, and the number of results to return.
Executing Vector Searches: ES accelerates vector search by using an inverted index, quickly locating documents containing specific vectors. During the search, ES identifies similar vectors based on the query vector's characteristics through the inverted index. Upon finding matching documents, ES calculates the similarity between the query vector and the document vectors. Common similarity calculation methods include cosine similarity, Euclidean distance, etc.
Returning Search Results: Based on the similarity calculation results, ES returns documents or results most similar to the query vector. The number and order of results returned can be customized. Moreover, after retrieval, it's possible to feed the top results into large language models (such as Mixer, GPT) for dialog-style result integration, ultimately delivering conversational search experiences to users.

Operational Practice

1. Apply for ES 8.8.1 cluster on Tencent Cloud ES.

2. Upload and deploy a custom NLP model in ES (optional).

   If you need to convert raw data into vectors in advance, you can perform inference tasks using big data models, segment and extract information from the text, and convert it into vectors. The biggest difference of Tencent Cloud ES 8.8.1 is that it supports users uploading pre-trained big data models locally without needing to restart the cluster, such as transformer models downloaded from HuggingFace, and manage and test the models on the Kibana interface. Finally, by integrating different Processors into the pipeline, data can be processed flexibly. For more details, see How to deploy local transformer models to Elasticsearch .
3. Create a Vector Index Structure in ES.

   Command examples and some parameter descriptions are as follows:

```
PUT my_vector_index
{
"mappings": {
   "properties": {
        "my_vector_field": {
           "type": "dense_vector",
            "dims": 768,              // Supports up to 2048 dimensions
            "element_type": "float",  // Supports float, byte
            "index": true,
            "similarity": "cosine",   // Supports cosine, dot_product, l2_norm
            "index_options": {        // Advanced hnsw parameter configuration
                        "type": "hnsw",
                        "m": 16,
                        "ef_construction": 100
            }
         }
       }
    }
}
```

4. Import vector data.

   If your data has already been vectorized, you can directly write it into ES using the following ES Bulk/Index API.

```
POST my_vector_index/_doc
{ "my_vector_field": [54, 10, -2],  "ip": "1.1.1.1", "name": "john" }
```

   If your data needs to be vectorized, you can specify an ES Ingest Pipeline configured with a previously uploaded NLP Model to perform inference tasks during real-time data entry. Models can be added, deleted, or updated as needed.
5. Perform vector retrieval

   The vector retrieval feature provided by ES 8.8 supports standalone vector retrieval and can be combined with any query to achieve efficient pre-filtering and hybrid search. It is compatible with aggregation, document-level security, field-level security, index sorting, and more. After vector search execution, you can still perform ES aggregation operations.

```
POST my_vector_index/_search
```

```
{
  "query": {                    // Optional, hybrid multi-scoring retrieval
      "match": {
        "name":  "john"
        }
     },
  "knn": {
    "field": "my_vector_field",
    "query_vector": [54, 10, -2],
    "k": 10,
    "num_candidates": 100,
    "query_vector_builder": {   // Optional, invoke the model for embedding
    "text_embedding": {
      "model_id": "sentence-transformers__msmarco-minilm-l-12-v3",
      "model_text": "How is the weather in Jamaica?"
      }
    },
    "filter": {                 // Optional, pre-filtering
      "term": {
        "file-type": "png"
      }
    }
}
"size": 100
}
```

The data in ES is scattered across multiple machine nodes in the form of shards. The vector search process is as follows:

1. First, search num_candidates approximate nearest neighbor candidate vectors on each shard.
2. Calculate the similarity between these candidate vectors and the query vector, and select k most similar results from each shard.
3. The Coordinator will merge the results from each shard to return the top size global final results to the users.

Therefore, users can increase num_candidates to improve the accuracy of the query results. Correspondingly, the query time consumption will also increase, trading time for accuracy. If your query has not been vectorized, you can specify a pre-uploaded large model for vectorization in the query statement.

Optimization Practices

During the process of supporting the access of a client's large online search service, the business continuously imported over 100 million vectors and conducted final performance stress testing. ES 8.8 easily handled up to one billion vectors and achieved an average query result response time in milliseconds. Below, we will specifically introduce the optimization practices for accessing this business.

1. Using segment merging significantly improves vector retrieval performance

   ES's write model uses a storage structure similar to LSM-Tree. The data written in real-time is first placed in Lucene's memory buffer and relies on writing to the translog to ensure data reliability. When it accumulates to a certain level, it is written in batches to a new segment. ES applies the same processing method to vectors, enabling real-time insertion and retrieval. However, this method generates many small segments, and vectors across different segments cannot be connected, resulting in low query efficiency. ES slowly and continuously merges underlying segments during cluster operation. We also recommend that businesses can schedule some forcemerge tasks during off-peak periods to reduce the number of underlying segments. Actual tests show it can improve vector retrieval performance by at least 3 to 5 times.

   The configuration used for the stress test was: 6 high IO SSD cloud data disks, 16 cores, 64GB, 100 million vectors, 96 dimensions. The final stress test results are as follows:

Was segment merge optimization done?	Performance Before Optimization (30 segments)	Performance After Optimization (2 segments)
Query QPS	622	3112
Avg query latency	307 ms	81 ms

   It is important to note that forcemerge requires a full rebuild of the graph index, which has a high resource occupancy. It is recommended to perform this during business off-peak periods. The ES cluster supports unlimited horizontal scaling. If the business performance is not satisfactory, it can be improved by adding more nodes and increasing the shard number.

2. Using ES 8.8, increase the merge thread pool size to improve forcemerge efficiency.

   Compared to previous versions, ES 8.8 has seen significant improvements in Lucene's merge of HNSW graphs, which aims to reuse the largest existing HNSW graph whenever possible. Thus, Lucene no longer starts from an empty graph but uses all previous work to build the largest existing segment. This improvement is substantial when merging larger segments, reducing segment merge time by 40% in benchmark tests. Tencent Cloud's ES self-developed kernel adds a new merge policy, automatically merging and cleaning segments that exceed the threshold during cluster idle time. This reduces storage costs and improves query performance without affecting overall cluster performance. If users need to perform forcemerge, they can configure the forcemerge thread pool size based on the machine's cores to speed up the forcemerge process.

```
thread_pool.force_merge.size: 8          // Initial default value is 2, which is relatively small
```

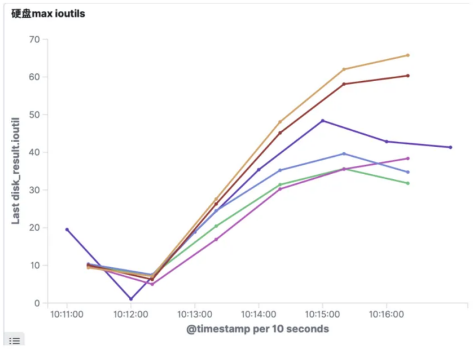
   The Cloud ES console also allows users to set scheduled tasks for forcemerge.

3. Allocate enough RAM and high IO disks when possible

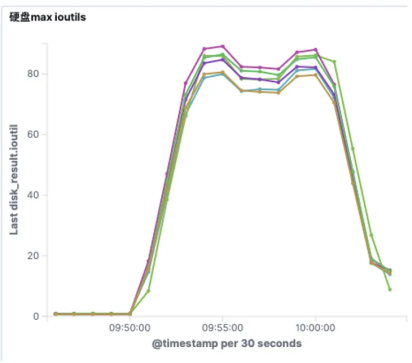
   ES stores vector data on the disk and uses inverted indexing to accelerate vector queries and matches, allowing the dataset to exceed the total available RAM on the local host. However, as the proportion of HNSW data that can fit into the page cache decreases, performance will decline. Performance-focused users should adjust RAM size according to the dataset size and opt for high IO hard drives, such as SSDs, to maintain good query performance.

4. Shard number, ES query parameters optimization.

   During performance testing, users encountered issues such as low stress test load and uneven load distribution. We resolved these issues by optimizing shard number, distribution, and other ES query parameters.



优化前：负载不均衡且资源利用率上不去



优化后：负载均衡且资源接近用满

5. Recall rate testing.

We calculated the recall rate based on different ef_construction parameters compared to brute force computation. The results are as follows.

efConstruction	100	200	500
Top 100 recall rate	>=96%	=100%	=100%
Top 200 recall rate	>=94.5%	>99.9%	=100%
Top 500 recall rate	>=90%	>=99.2%	>99%

Summary

The recently launched Tencent Cloud Big Data ES version 8.8.1 offers powerful cloud-based AI enhancements and vector retrieval capabilities. It supports implementing natural language processing, vector search, and integration with large models on an end-to-end search and analysis platform. The average response latency for billion-level vector retrieval is controlled at the millisecond level, helping customers achieve advanced AI-driven search capabilities and bringing a new cutting-edge experience to search and analysis.

Tencent Cloud ES RAG Practice Tutorial: Implement Intelligent Question Answering on ES Documentation with Just Hundreds of Lines of Code

Last updated: 2024-11-20 17:28:23

Introduction

The rapid development of information technology has led to an explosive growth of massive data. On one hand, increasing data can bring convenience to our lives; on the other hand, software development faces enormous challenges—different structured data like images, audio, and videos are increasingly appearing, posing immense challenges to search analysis. Traditional keyword searches limit search results to the input keywords, resulting in poor user experience.
Thus, the emergence of vector retrieval provides us with a new thought process. Tencent Cloud VectorDB stores unstructured, semi-structured, and even structured data in vector form, enabling operations like similarity search, clustering, and dimensionality reduction, combined with machine learning models, to offer users a more intelligent search service.

However, integrating **text search** + **vector retrieval** + **AI capabilities** into one, and being mature, stable, and trustworthy, cloud search engines like this are rare in the industry. Tencent Cloud Big Data ES has recently launched its ES 8.8.1 version, which boasts the following advantages:

- It provides powerful cloud AI enhancement and vector retrieval capability;
- It supports the implementation of NLP, vector search, and integration with large models in an end-to-end search and analysis platform;
- Billions-level vector retrieval with average response latency controlled at the millisecond level, helping participants achieve AI-driven advanced search capability, bringing a cutting-edge experience to search and analysis.

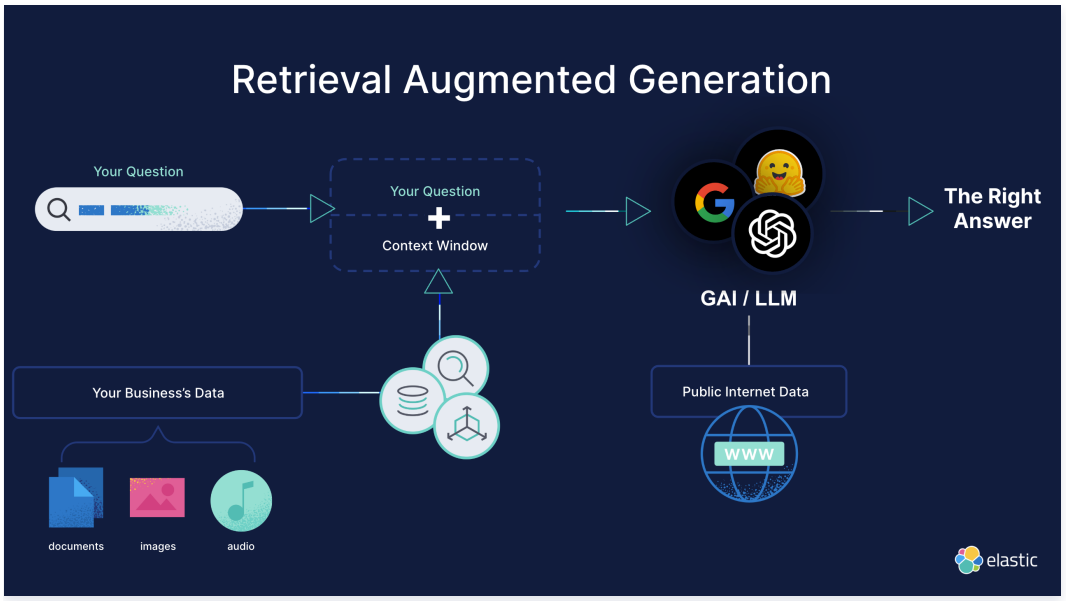
This article will demonstrate how to achieve AI Q&A with just a few hundred lines of code by integrating Tencent Cloud ES and ChatGPT.

RAG Introduction

Before we delve into the formal introduction, let's first explain RAG. With the continuous development of search engine technology, our query demands have also been increasing. Traditional keyword search can no longer meet users' requirements for query accuracy and efficiency. Therefore, we introduced semantic search technology. By using advanced NLP technology, semantic search can better understand users' query intents and return more relevant search results. And with the continuous development of machine learning technology, especially the popularity of generative large models like ChatGPT, a new technical direction has emerged — RAG.
However, truly understanding what RAG is not easy, and implementing RAG is even more challenging. The current situation is that most of the time users simply understand implementing RAG as adding a Tencent Cloud VectorDB in the enterprise. But RAG is a complex concept; it is not just a Tencent Cloud VectorDB. Implementing RAG requires an in-depth understanding of business scenarios, a large amount of data processing, and algorithm optimization. The understanding and feedback of user behavior are also critical to achieving the final effect. Therefore, we need more of a solution rather than just a high-performance Tencent Cloud VectorDB.

What is RAG?

RAG is the abbreviation for Retrieval Augmented Generation, meaning retrieval-enhanced generation. It is a method that uses large language models (LLMs) and search engines like Elasticsearch to retrieve relevant information from vast amounts of text data and then generate new text based on this information. RAG can be used for various applications, such as knowledge Q&A, text summaries, dialogue generation, etc. The final effect of RAG is critically dependent on accurately understanding the user's issues, providing accurate and abundant context data, excellent generative large models, and appropriate prompts.



Why implement RAG?

For example, on the official documentation, the current text retrieval method presents the following issues:

- **Vocabulary Mismatch (lexical mismatch):** When the documentation and the inquiry use different words to express the same or similar meaning, traditional search methods may not find relevant documents. For instance, in the context of the ELK Technology Stack, "logstash" and "ls" mean the same thing in context, but they are different words lexically.

- **Semantic Mismatch (semantic mismatch):** When the documentation and the inquiry use the same or similar words to express different or uncorrelated meanings, traditional search methods may find uncorrelated documents. For example, "Elastic" can refer to elasticity or a technology company, but they are different in semantics.
- **Language mismatch:** When documents and queries use different languages, traditional search methods may fail to find relevant documents. For example, "Serverless architecture" and "无服务器架构" refer to the same architecture, but they are different in

language.

Therefore, we need to solve the above issues through semantic search to find the corresponding documents. However, the actual needs of customers may be more complex. The customer's questions might be:

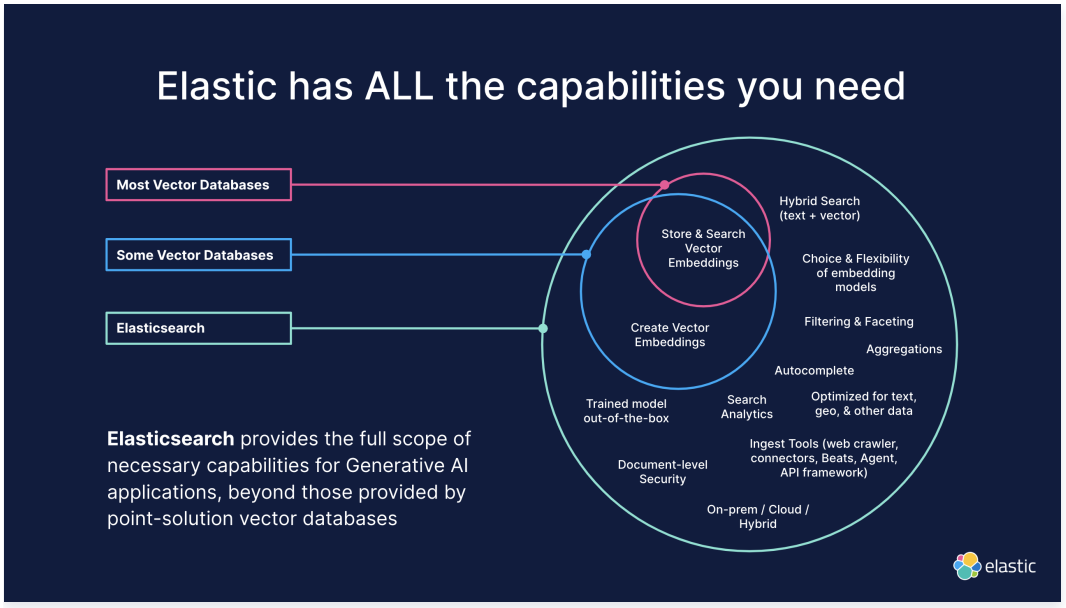
- How is the ES Serverless service charged? Please provide a detailed example.
- How to terminate an ls instance? Give me the specific procedures.

In such scenarios that require learning documentation and providing guidance, relying solely on full-text search, semantic search, or hybrid search may not achieve satisfactory results. To reach this goal, we need to combine large models and enhance retrieval generation (RAG) based on enterprise private data to improve efficiency and accuracy while providing guiding procedures.

Using Tencent Cloud ES for RAG

Tencent Cloud ES (ES) is a fully managed cloud massive data retrieval and analysis service, featuring a high-performance proprietary kernel and integrating X-Pack. ES supports easy cluster management through autonomous indexing, compute-storage separation, cluster inspection, and also supports zero maintenance, automatic elasticity, and on-demand usage in Serverless mode. With ES, you can efficiently build services for information retrieval, log analysis, and operations monitoring. Its unique vector retrieval can also help you build AI deep applications based on semantics and images.

The recently launched Tencent Cloud ES version 8.8.1 includes a comprehensive engine with all the features needed to implement RAG, from vector storage and search to vector generation and hybrid search ranking. It even includes a proprietary vector model:



To demonstrate the complete process of using Elasticsearch for RAG, we will create a case study using Tencent Cloud ES and ChatGPT, implementing it within Tencent Cloud ES's official documentation. The steps are as follows:

- Create an ES cluster and configure the relevant parameters and plugins.
- Collect and understand our data. In this example, the Tencent Cloud ES help documentation will serve as our knowledge base to study how to enhance semantic search by combining large models.
- Process the data so it can be used for semantic search. Use the BAAI/bge-base-zh model to convert text into vectors and store both vectors and text in ES.
- Use the API provided by ES or Kibana tool to conduct a hybrid search of vector + text.
- Use the API or SDK provided by large models to conduct RAG dialogue generation.

Below, we will describe the specific operations for each step in detail.

Creating ES cluster

Apply for ES 8.8.1 cluster on Tencent Cloud ES.

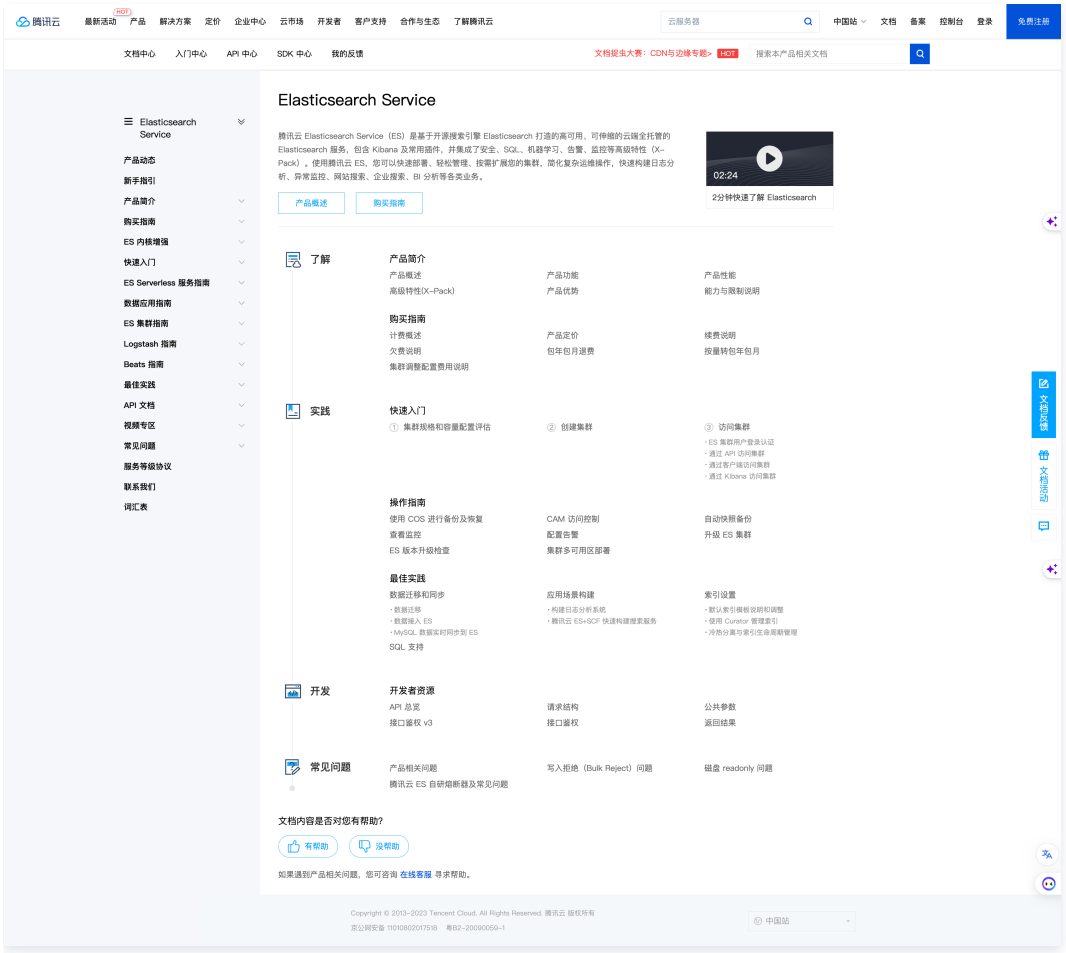
Note:
Since deploying an embedding model requires adequate memory, we should ensure the selected memory is sufficient. To implement the contents in this example, it is recommended to choose data nodes with 8G and above memory configuration.

Collect and understand our data

To implement RAG in specific scenarios, the first question we face is how to collect the data that may be used in these scenarios. Unfortunately, most of the time, this data is not well-managed structured data, such as the latest regulatory documentation, internal employee handbooks, sensor data, logs, and promotional records. This data may be stored in Wikis, network disks, emails, logs, or scattered across multiple different databases. How to collect and understand this data, and how to convert it into fields with different search weights, poses significant challenges, which are left to the developers to handle.

Using the Tencent Cloud ES documentation as an example, the documents are stored in Markdown format on GitHub and are updated regularly. To implement RAG on them, you need to contact the relevant business team to obtain data synchronization permissions and perform periodic updates. Additionally, this is a typical semi-structured data case; although there are HTML and Markdown tags, the content and format of each article are inconsistent. Extracting the appropriate "context window" requires effectively reorganizing the data in the search engine.

Search for the required CAM policy as needed, and click to complete policy association.

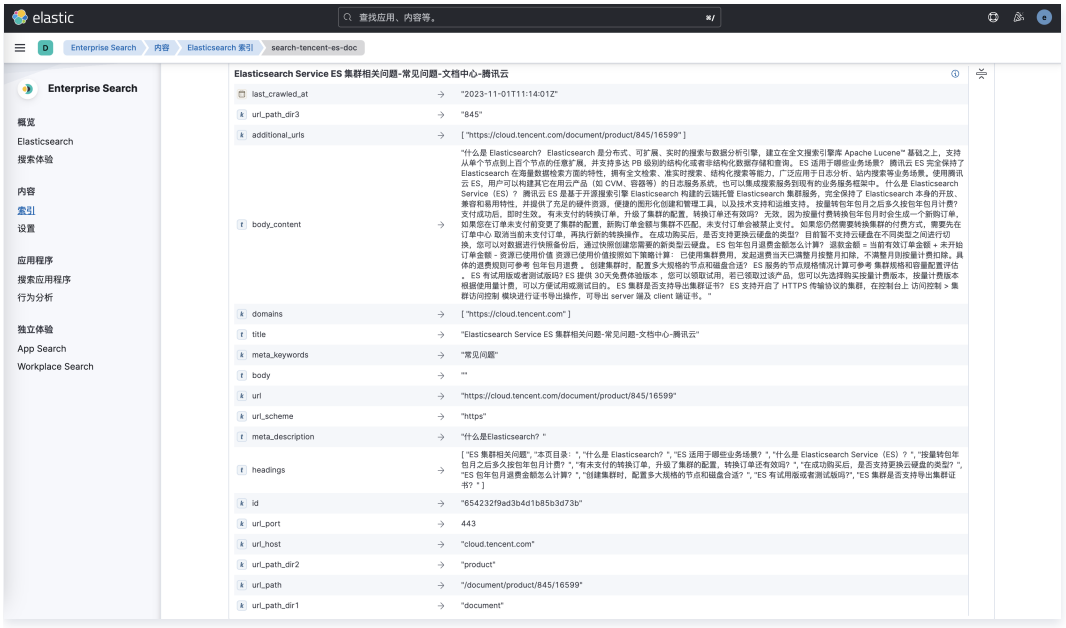


To address these issues, the enterprise search feature provided by ES significantly reduces our workload. From communication, data collection, and information extraction, to data understanding.

To minimize reading time, we have created a short video for you to quickly understand how we use crawlers to capture page data and how the captured data is processed and cleaned through a pipeline. Please refer to the video demonstration below:
Watch video

> **Note:**
> If you need to use the enterprise search feature for data ingestion, please contact us for the relevant operations.

After completing data collection and simple processing, we can directly analyze our scenario data in Kibana:



It can be seen that the core content is included in three fields: `body_content`, `headings`, and `title`. To support RAG with semantic search, vectors need to be generated for these three fields while retaining the original text fields for page display references and sending to large models. Specifically, the `headings` field (automatically generated by a crawler) contains the titles of all sections in the document. This effective summary text is particularly suitable as a vector representation of the text content without chunking and segment storage. Here is an example:

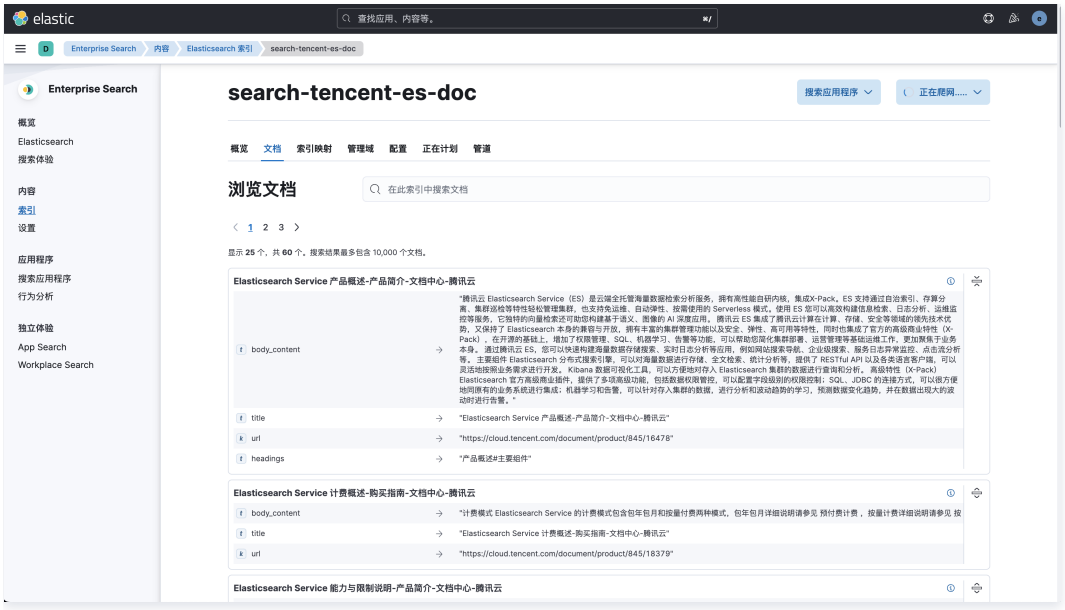
```
"headings":"Cluster specifications and capacity configuration assessment#Table of contents:#Storage capacity assessment#Computing resource assessment#Instance type selection and testing#Shard number assessment"
```

The remaining fields, such as `url`, can be used to link to the source document. All other fields can be deleted. Since the type is an array, it needs to be converted to text for vector conversion. Therefore, after the following data pipeline:

```
PUT _ingest/pipeline/search-tencent-es-doc@custom
{
  "version": 1,
  "description": "Enterprise Search customizable ingest pipeline for the 'search-tencent-es-doc' index",
  "processors": [
    {
      "rename": {
        "field": "doc_content",
        "target_field": "body_content",
        "ignore_missing": true
      }
    },
    {
      "join": {
        "field": "headings",
        "separator": "#"
      }
    },
    {
```

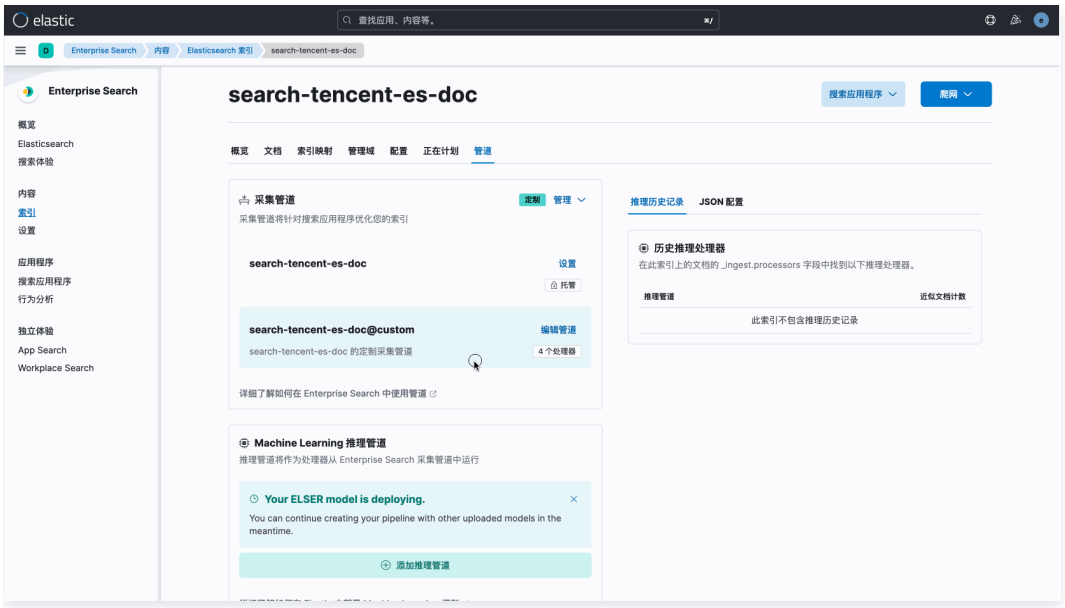
```
      "remove": {
        "keep": [
          "headings",
          "body_content",
          "url",
          "title"
        ],
        "ignore_missing": true,
        "description":"Remove unnecessary fields"
      }
    }
  ]
}
```

Data after cleaning:

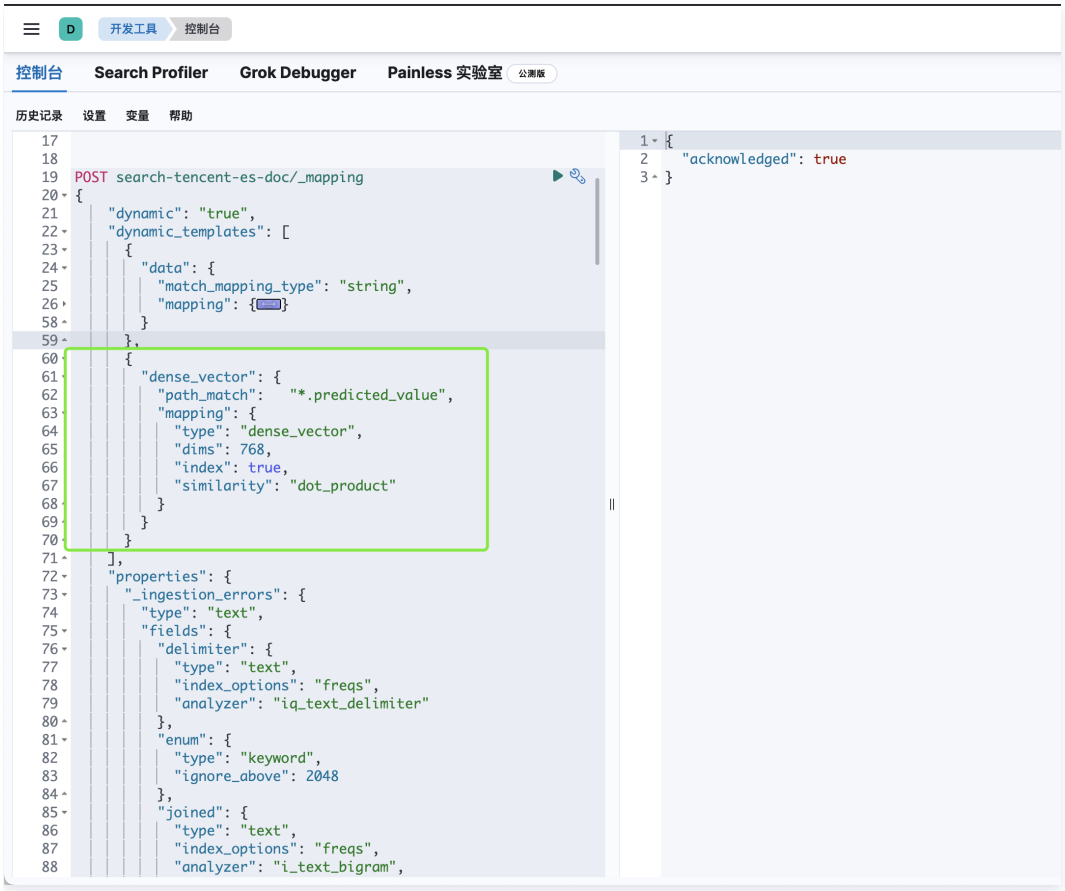


Process the data to enable Semantic Search

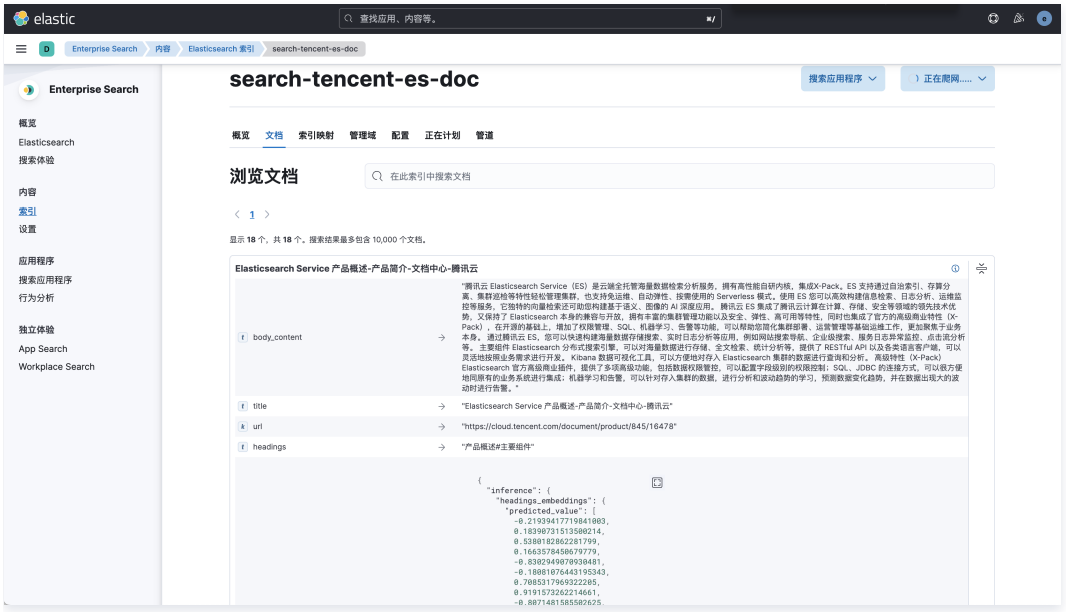
In a RAG system, users ask questions through natural language conversation. Therefore, the engine that retrieves local data must support Semantic Search. Typically, we can achieve Semantic Search through Sparse Vector Retrieval and Dense Vector Retrieval. Two methods are supported on ES, and the ELSER proprietary model is provided for Sparse Vector Retrieval. However, in this example, we will use Dense Vector Retrieval for demonstration. For this, let's first add a Machine Learning Inference Pipeline for vector generation:



Here, the model we use is BAAI/bge-base-zh (in fact, we can deploy multiple embedding models in ES and achieve different effects by debugging and replacing models in different scenarios). This model converts inputs into 768-dimensional vectors. Therefore, we need to define a dense vector field to store data generated by this model:



The final data containing vectors is:



Conduct hybrid search using Vector Retrieval + Text Search in ES

In some common tutorials, you need to combine pre-packaged modules and tools like Langchain to quickly conduct Semantic Search. However, with Tencent Cloud ES, we can achieve this with just a few lines of code.

- **Note:**
  So far, we haven't written any code. All data collection, cleaning, and vector generation can be accomplished using the UI tools provided.
- **For a programmer skilled in using various tools on Elasticsearch, the entire process above might take no more than 30 minutes.** This allows us to quickly move into the stages of vector model and large model debugging.

In the hybrid search phase, thanks to Elasticsearch effectively integrating vector retrieval features with its original full-text search, the whole process requires minimal time. From a code perspective, testing can be done by writing a simple function:

```
from elasticsearch import Elasticsearch


# Search ElasticSearch index and return
def search(es, index_name, embedding_model, query_text):

    source_fields = ["body_content", "url", "title"]
    query = {
        "bool": {
            "must": [{
                "match": {
                    "title": {
                        "query": query_text,
                        "boost": 1
                    }
                }
            }]
        }
    }
    knn = [{
        "field": "ml.inference.headings_embeddings.predicted_value",
        "query_vector_builder": {
            "text_embedding": {
                "model_id": embedding_model,
                "model_text": query_text
            }
```

```
        },
        "k": 5,
        "num_candidates": 10,
        "boost": 24
    }, {
        "field": "ml.inference.body_content_embeddings.predicted_value",
        "query_vector_builder": {
            "text_embedding": {
                "model_id": embedding_model,
                "model_text": query_text
            }
        },
        "k": 5,
        "num_candidates": 10,
        "boost": 24
    },
    ]
    resp = es.search(
        index= index_name,
        fields=source_fields,
        query=query,
        knn=knn,
        size=3,
        source=False)

    return resp
```

Regarding the above code, the key points to focus on are:

1. We usually need to combine mixed search methods to get the best results.
2. Meanwhile, when performing vector search, we can conduct vector retrieval on multiple fields.
3. By passing `embedding_model` as a variable, we can choose different model combinations during debugging to observe the performance of semantic retrieval and see which documents will be recalled.
4. Because we need to provide the retrieved content to the large model for understanding, limited by the model's token restrictions, we usually only recall 1~3 documents.**In this scenario, precision is more important than recall rate!**

Through semantic search, compared to the original full-text search, when searching for "How to destroy ls nodes," we can find information that was not previously available:

```
es = Elasticsearch(hosts=["https://es-7cu6zx9m.public.tencentelasticsearch.com:9200"],
                       http_auth=(username, password))
print(search(es,"moka-ai__m3e-base","How to destroy ls nodes?"))
>>>>>>>>>>>>>
{'took': 79, 'timed_out': False, '_shards': {'total': 2, 'successful': 2, 'skipped': 0, 'failed': 0}, 'hits':
{'total': {'value': 24, 'relation': 'eq'}, 'max_score': 47.651337,
'hits': [{'_index': 'search-tencent-es-doc', '_id': '6542334ead3b4de80cb57864', '_score': 47.651337,
'fields': {
'body_content': ['Operation scenario: When the Logstash instance cannot meet your needs and requires a
return, you can destroy the instance in the ES console to avoid incurring ongoing service costs. If the
instance configuration does not meet your needs, you can also adjust the instance configuration to a suitable
specification, as detailed in Instance Scaling. Refund instructions for different billing modes: For pre-paid
yearly and monthly subscription instances, if the instance has not yet expired but needs to be destroyed
early, refer to Prepaid Refund. For post-paid pay-as-you-go instances, which are billed based on usage, the
instance can be destroyed at any time, and no further charges will be incurred after destruction. Note: Data
cannot be recovered after an instance is destroyed, so please proceed with caution. Steps: 1. Log in to the
ES console, click Logstash Management in the left navigation bar to enter the Logstash instance list. \ufeff
2. In the instance list, select the instance to be destroyed, choose Action > More > Destroy to destroy it;
or click the instance ID/Name to enter the instance basic information page and select More Actions > Destroy
in the upper right corner to destroy it. \ufeff 3. On the destroy instance page, click Confirm. The system
will empty the instance data and reclaim resources. Once the data is cleared, it cannot be recovered. For
details on the refund method for yearly and monthly subscriptions, refer to Prepaid Refund.'],
```

```
'title': ['Destroy Instance in ES - Logstash Guide - Document Center - Tencent Cloud'],
'url': ['https://cloud.tencent.com/document/product/845/55114']}},
```

If you search for "无服务器ES", you can also find related results, without the language barrier preventing the recognition of the relationship between "Serverless" and "无服务器".

Using large models for RAG-based dialogue generation

However, as we've mentioned before, the client's actual needs may be more complex. The client's question might be:

- How is serverless ES charged? Please give me a detailed example.
- How to terminate an ls instance? Give me the specific procedures.

We need to pass the retrieved documents to the large model for generating guiding content. Due to the token limitations of large models, it is impossible to pass unlimited recalled content to the large model for understanding and learning. Additionally, recalling uncorrelated documents might lead to hallucinations. Setting up a well-crafted prompt according to the scenario is also crucial for the large model to provide answers according to our expectations. Thus, its implementation is as follows:

```
def truncate_text(text, max_tokens):
    tokens = text.split()
    if len(tokens) <= max_tokens:
        return text
    return ' '.join(tokens[:max_tokens])
# Generate a response from ChatGPT based on the given prompt
def chat_gpt(prompt, model="gpt-3.5-turbo", max_tokens=1024, max_context_tokens=4000, safety_margin=5):
    # Truncate the prompt content to fit within the model's context length
    truncated_prompt = truncate_text(prompt, max_context_tokens - max_tokens - safety_margin)
    print(truncated_prompt)
    response = openai.ChatCompletion.create(engine="gpt-35-turbo",
                                            messages=[{"role": "system",
                                                       "content": "You are now an expert in Tencent Cloud
ES."},
                                                      {"role": "user", "content": truncated_prompt}])
    return response["choices"][0]["message"]["content"]

negResponse = "Based on the retrieved documents, I cannot answer this question."
resp = search(es,embedding_model,query)
retrival_result = []
for hit in resp['hits']['hits']:
    #Avoid hallucinations
    if hit['score'] > 30:
        retrival_result.append(hit['fields']['body_content'][0])
# Combining the recall content, you can also choose only top 1
combine_result = ";".join(retrival_results)
print(combine_result)
prompt = f"Answer this question: {query}\n If the provided document does not contain the answer, please
reply: '{negResponse}'\n When answering, refer to the help documents from Tencent Cloud ES: {combine_result}
"
answer = chat_gpt(prompt)
```

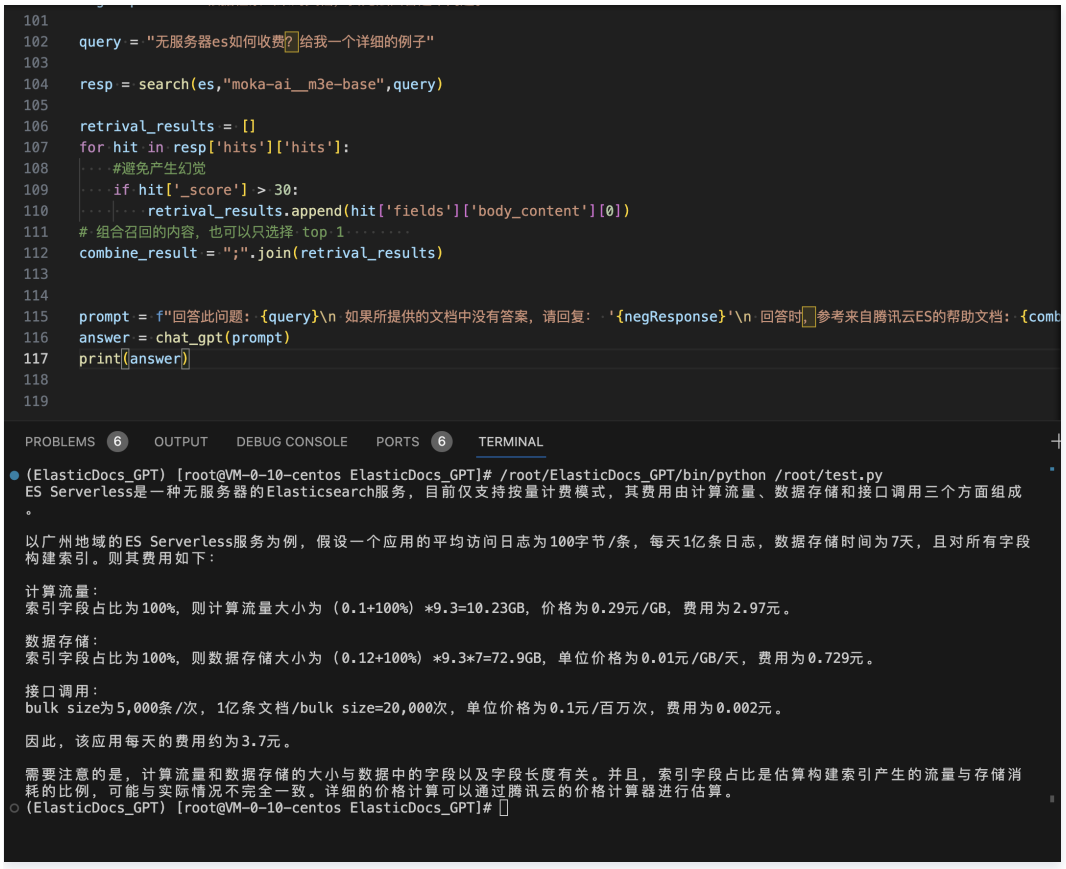
Note:

Regarding the above code, the key points to focus on are as follows:

- Due to the limitations of large models, the `truncate_text()` function and parameters such as `max_context_tokens` are defined to avoid failures due to exceeding tokens.
- Limit the context information for the large model. Prevent irrelevant documents from misleading the large model's understanding by using `if hit['score']>30` .
- Set reasonable prompts to let the large model understand its role and how the role should handle the problem:
  - `"role":"system","content":"You are now an expert in Tencent Cloud ES."`

```
"role":"user","content":"Answer this question: {query}\n If the provided document does not contain the answer, please reply: '{negResponse}'\n When answering, refer to the help documents from Tencent Cloud ES: {combine_result} "
```

By simply running the above code, we can get the following result:



In the above example, we asked, "How is serverless ES charged? Give me a detailed example," and received a comprehensive, model-generated example. Successfully achieved RAG! By integrating simple Python frameworks like streamList, we can easily build an intelligent assistant program:

Summary

In this article, we introduce how to quickly build a search-enhanced generative AI assistant using Tencent Cloud ES and ChatGPT. By leveraging Tencent Cloud ES features like data collection, data cleansing, data analysis, along with vector generation, vector storage, vector retrieval, and combining them with full-text search capabilities, we can rapidly integrate with large models, creating an AI assistant with a user interface in just a few lines of code.
It's not just a recreational app development endeavor; for larger Enterprise Production Environments, Tencent Cloud ES can deliver higher efficiency and more stable performance. Years of development and testing have honed Tencent Cloud ES into a reliable platform.
In the future, we will continue to optimize Tencent Cloud ES to meet more demands of Enterprise Production Environments and provide a better user experience. We strive to bring more innovation and convenience to AI assistant development and applications.

If you are interested in these capabilities, scan the QR code below to join the Tencent Cloud ES AI Enhancement and Vector Retrieval Communication Group!

Bring enterprise knowledge within reach and enjoy RAG application practices based on Tencent Cloud ES

Last updated: 2024-10-25 09:35:01

Lexiang leverages Tencent Cloud ES's one-stop text retrieval and vector retrieval hybrid search capabilities to create an intelligent Q&A module for enterprise-level knowledge management. Through precise information retrieval, strict permission management, and comprehensive knowledge management, it helps internal employees and external partners efficiently mine the knowledge treasures of enterprises and organizations.
This article introduces the practice of Lexiang's one-stop RAG solution based on Tencent Cloud ES to realize the Lexiang intelligent Q&A feature.

Lexiang's Business Background and Challenges

Lexiang is an intelligent organizational learning collaboration platform that integrates knowledge management, corporate training, and cultural construction. It features knowledge bases, Q&A, classrooms, examinations, and more, aiming to help enterprises establish efficient and flexible knowledge sharing and mobility mechanisms to enhance organizational capability.
With the development of large-scale model technology, Lexiang sees unprecedented opportunities to improve the efficiency and accuracy of information acquisition, driving innovation in business models and service modes.
However, Lexiang also faces new challenges, especially in solving the "illusion" problem with large models, ensuring personalized responses while guaranteeing data security and privacy protection. Corporate "private knowledge" cannot be directly used for the pre-training and fine-tuning of large models, which poses higher demands on knowledge management.
How to use the "new bottle" of large models to encapsulate the "old wine" of enterprise knowledge management, maximizing customer value with one hand and commercializing product experience with the other, has become a problem that the Lexiang team must solve.

What is RAG?

With the development of search engine technology, traditional keyword searches can no longer meet user needs. RAG (Retrieval-Augmented Generation) combines vector retrieval and text generation; it searches for relevant documents through a retrieval model and uses them as contextual information for the generation model to produce accurate answers. This method combines external knowledge with the generation model, enhancing the smoothness and accuracy of the content, and is widely used in scenarios such as knowledge Q&A, text summary, and dialogue generation.

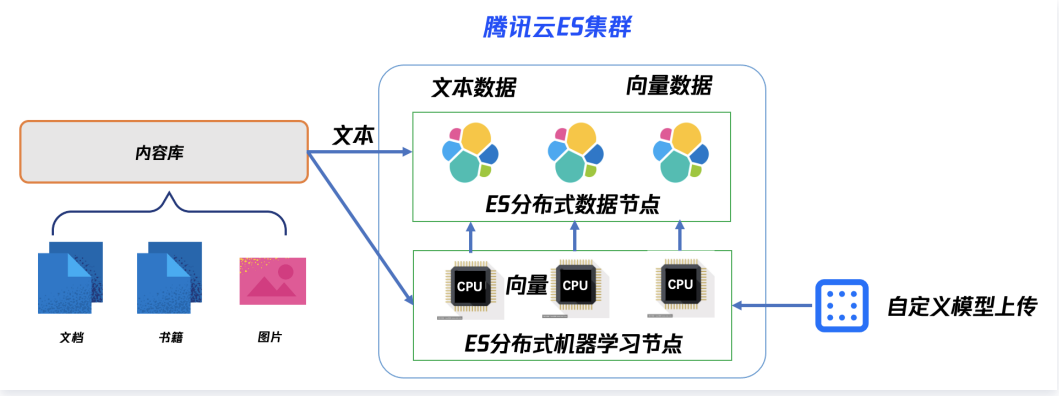
Tencent Cloud ES One-Stop RAG Solution

One-stop vector retrieval capability

Based on the capabilities provided by Tencent Cloud ES, we can directly complete the full process operations of **Vector Generation** > **Vector Indexing/Storage** > **Vector Retrieval**. The specific capabilities are as follows:

- Built-in, ready-to-use ELSER semantic model.
- Support for uploading self-defined models or directly uploading models from open-source communities like Hugging Face.
- Support for proprietary machine learning nodes used for vector reasoning (optional, can also be mixed with data nodes).
- Support for vector indexing (HNSW Algorithm), vector storage, and KNN retrieval.

Search for the required CAM policy as needed, and click to complete policy association.



Unique Hybrid Search Capability

Despite the ability of vector search to perform semantic queries, it may return irrelevant results when dealing with specific strings like "8XLARGE64". By combining the advantages of vector and text search, hybrid search improves the recall rate through simultaneous execution of text and vector searches and merging the results. Tencent Cloud ES supports one-stop hybrid search, allowing text and vector searches to be performed simultaneously in a single query and automatically sorting and merging the results.
Search for the required CAM policy as needed, and click to complete policy association.

Reciprocal Sorting Fusion (RRF)

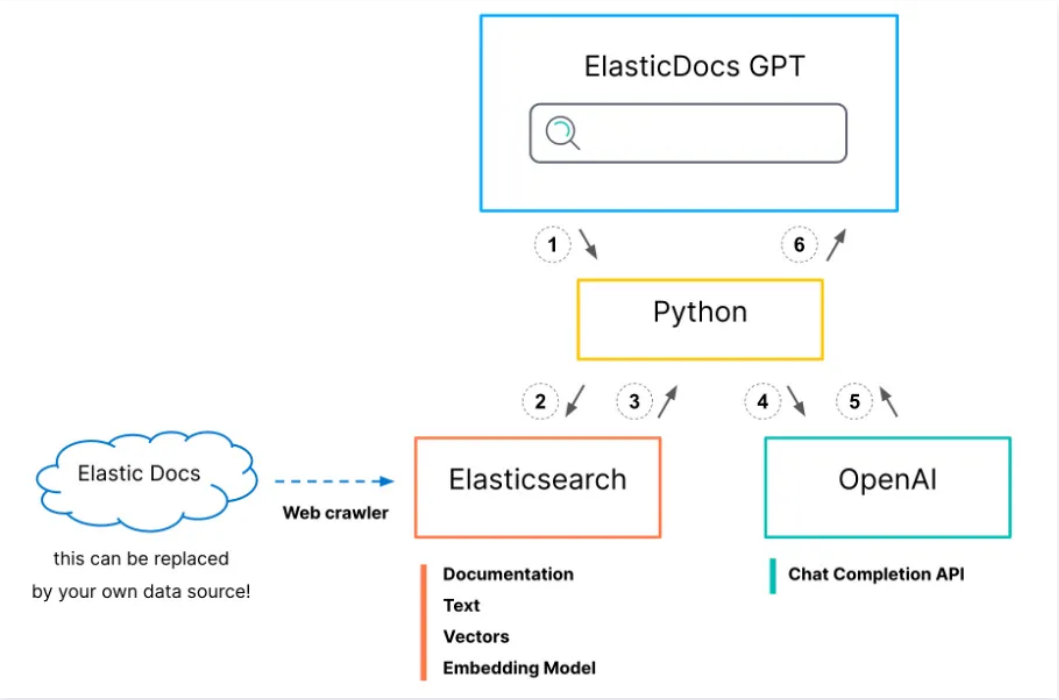
In multi-channel recall, different system scoring mechanisms require normalized scores for fair comparison and fusion. Tencent Cloud ES's built-in Reciprocal Sorting Fusion (RRF) algorithm assigns weights to rankings and calculates the sum of the reciprocals of each system's rankings to generate the final merged ranking list. The advantages of RRF include:

- **Simplicity:**No need for complex normalization, just know the rank of each document in each system.
- **Robustness:**Less sensitive to different scoring scales and distributions.
- **Fairness:**Ensures fairness in the merging process by assigning weights using the same formula.
- **Easy to Implement:**Relatively simple to implement, no complex parameter tuning or training required.
- **Adaptability:**Can adapt to new systems or data, not reliant on specific scoring mechanisms.

Integration with LLM Large Model

Tencent Cloud ES can seamlessly integrate with third-party pipelines such as Langchain and LLM Large Models to help build generative applications.

Search for the required CAM policy as needed, and click to complete policy association.

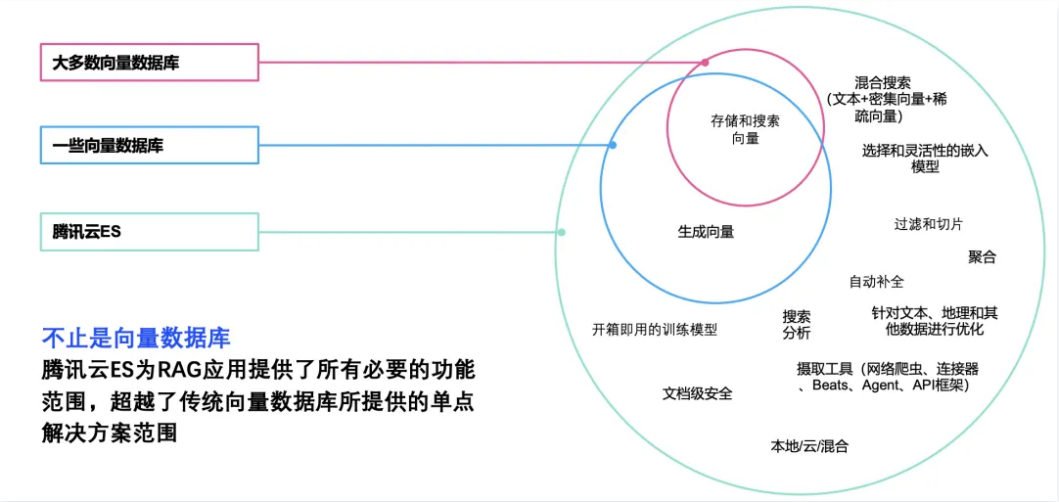


Advantages of Tencent Cloud ES RAG Scheme

Tencent Cloud ES offers a one-stop RAG solution, providing all features within the RAG application scope, surpassing the single-point solutions provided by traditional Tencent Cloud VectorDB. The details are as follows:

- Supports direct upload from Definition embedding models or embedding models from hugging face, completing model management, deployment, and data vectorization within ES.
- Supports hybrid search capabilities of text + vector to flexibly meet various needs.
- Supports scoring and ranking by setting the weight of text, vector, and even field-level term weight. You can also directly use the built-in RRF capability to achieve ranking fusion. Later, the LTR model can be introduced to re-rank coarse-ranked results, meeting the needs of different business scenarios.
- In some permission-sensitive scenarios, ES supports role-based document and field-level permission control for precise access management.

Search for the required CAM policy as needed, and click to complete policy association.



Moreover, RAG applications typically serve online business scenarios, and stability is crucial. Tencent Cloud ES supports multiple replicas setting, multiple availability zones deployment, CCR cross-cluster replication, automatic data backup, and AS capabilities to ensure the stability of online businesses. Additionally, intelligent patrol, multi-dimensional stereoscopic monitoring and alerting, and audit log capabilities help enterprises reduce operation and management costs, achieving cost reduction and efficiency improvement.

Tencent Cloud ES high-accuracy high-performance self-developed kernel

Tencent Cloud ES not only provides a more comprehensive RAG solution compared to the traditional Tencent Cloud VectorDB but also has unique advantages in performance. The Tencent Cloud ES RAG solution significantly enhances the efficiency and precision of vector retrieval, leveraging the powerful performance of the Zixiao GPU and the deeply optimized self-developed kernel.

Self-developed kernel optimization

At the kernel level, Tencent Cloud ES has deeply optimized typical vector scenarios, such as sharded architecture optimization, query parallelization, Lucene query cache lock transformation, etc. The average response latency for 1 billion-scale vector retrieval is controlled at the millisecond level, and the overall query performance is improved by 3 to 10 times:

- **Hybrid Search:** Real-time joint retrieval and multi-channel recall based on vector retrieval and BM25 algorithm, with result fusion significantly improving the recall rate while maintaining high performance.
- **Scalar Quantization of Vectors:** Converting float8 to int8 representation, significantly reducing CPU and heap memory resource consumption and search latency while ensuring that the recall rate is not affected.
- **Adaptive Replica Policy:** ES distributed systems involve many network calls. Cross-region/AZ calls or slow node shard replica calls can lead to high query latency. Tencent Cloud ES adopts an improved local adaptive replica selection policy. The core principle is to calculate the average query response time, query queue, and query success rate between the Coordinator and data shard replicas, continuously adjusting to select the replica with the lowest latency for queries. This significantly reduces search latency and ensures CLB.
- **Query Pruning:** The ES query model splits a query request into shard-level sub-requests forwarded to each shard for parallel execution, finally merging results at the Coordinator. Each shard contains multiple segments. Tencent Cloud ES pre-prunes and merges segments based on dimensions such as columnar storage, numerical indexes, and terms, reducing random IO and optimizing query performance.
- **Query Parallelization:** By leveraging idle CPU resources, a single shard-level request in ES is split into 3-5 sub-requests for parallel processing across segments or docs. Each thread processes a portion of the docs or segments. The results from each thread are merged at the data node and returned to the Coordinator, which then merges the results from all shards and returns them to the client, achieving a multiplicative performance boost.
- **Query Cache Optimization:** Using the CBO strategy, query cache operations that could cause latency spikes of 10x are avoided. The LRU cache performance is more than doubled through fine-grained read-write locks. These improvements have been submitted to and recognized by the Elasticsearch and Lucene communities.

Zixiao GPU Support

Tencent Cloud ES is the world's only ES service that supports GPUs. It can leverage Tencent's self-developed "core" technology, Zixiao, which combines hardware and software to fully utilize the performance benefits of GPUs and improve the efficiency of vector generation and retrieval. Zixiao V1 features high energy efficiency, high throughput, and high bandwidth. Its design specifications are comparable to the NVIDIA A10, with memory bandwidth 30% higher than the A10, and overall performance up to 50% – 100% better. For common small to medium models, Zixiao typically offers over 100% performance improvement compared to the NVIDIA T4, and a performance edge of over 20% compared to the NVIDIA A10.

Performance Comparison of Tencent Cloud ES and Milvus

We conducted detailed performance tests using the open-source tool ann-benchmark. By retrieving various dimensional datasets and achieving a recall rate of 99%, we obtained the Top 10 most similar documents. We then compared the QPS data of vector retrieval between Milvus and Tencent Cloud ES.

Public Datasets	Dimension	Configuration	Recall Rate	QPS	
				Milvus	Tencent Cloud ES
sift-128-euclidean	128	3 nodes, 4C8G, HNSW(m=16, efConstruction=200)	99%	3653	19479
gist-960-euclidean	960			480	4136

It can be seen that whether in high-dimensional or low-dimensional vector retrieval scenarios, Tencent Cloud ES has a performance advantage of nearly 5-8 times, helping business development with lower costs and better performance, achieving cost reduction and efficiency improvement.

Lexiang's RAG Practice Based on Tencent Cloud ES

The core goal of the LeXiang intelligent search module is to help internal employees and external partners efficiently mine the knowledge deposits within the enterprise. Although the interaction with intelligent search is not complex, achieving good search results is not as easy. The first issue to address is the "illusion" problem of large models, to be precise, reducing the "illusion" to the minimum. For enterprise customers, honestly saying "I don't know" has a much smaller impact than a clearly incorrect answer. Furthermore, the "private knowledge" of these enterprises obviously cannot be directly used in large model pre-training (Pre Trained) and fine-tuning (Fine Tuning). For such problems, LeXiang adopts the RAG (Retrieval Augmented Generation) approach.
Search for the required CAM policy as needed, and click to complete policy association.

In the process of implementing RAG, Lexiang used the hybrid search of vector retrieval and precise text matching provided by Tencent Cloud ES, combined with its built-in Reciprocal Sorting Fusion (RRF) algorithm, to achieve multi-path recall, hybrid search, and hybrid sorting, ultimately achieving high recall rate and high accuracy rate in data retrieval.

Practice Results

By implementing Tencent Cloud ES's RAG solution, Lexiang's intelligent search module achieved the following results:

- **High Recall Rate and Accuracy:** By using hybrid search technology, Lexiang effectively enhanced the recall rate and accuracy of vector retrieval, ensuring more relevant and precise user query results.
- **Improved User Satisfaction:** The intelligent search module reduced the "hallucination" phenomenon of large models, increasing the accuracy and reliability of answers, significantly boosting user satisfaction.
- **Enhanced Business Efficiency:** Internal employees and external partners can quickly access the needed knowledge, improving work efficiency and collaboration.
- **Data Security:** Through strict permission management, Lexiang ensured the security and privacy protection of enterprise data, enhancing user trust in the system.

Summary

Tencent Cloud ES's one-stop RAG solution performed excellently in the field of intelligent retrieval, helping enterprises reduce costs and improve efficiency, achieving continuous growth in business value.
Through cutting-edge large model technology, Lexiang transformed years of accumulated enterprise data into valuable knowledge, aiding enterprises in building an intelligent learning community.
In the future, Tencent Cloud ES will continue to delve deeper into the field of intelligent retrieval, consistently improving in cost, performance, and stability, helping customers achieve greater business value. Thank you all for your support, and we welcome you to continue following Tencent Cloud ES and Lexiang!

Tencent Cloud ES Serverless x TKE: Achieve an Observable Log Analysis System in Minutes with Low Threshold

Last updated: 2024-10-25 09:36:51

Introduction

As a cloud-native technology enthusiast, I am active in various developer communities. Recently, Tencent Cloud Elasticsearch Serverless has deeply attracted me with its automatic elasticity, maintenance-free features, rich product capabilities, and the ability to achieve TKE's log collection and observable analysis in minutes. To let everyone freely experience this out-of-the-box product, I have managed to secure free experience vouchers, a 1-yuan purchase of a resource package, and developer courses. For detailed acquisition methods, please refer to the relevant links at the end of the article.

Cloud-native and Containers

With the rapid development of technologies like containers, Serverless, and microservices, cloud-native has gradually built a prosperous technology system. Today, cloud-native is continuously activating application building paradigms with its advantages of cost reduction, efficiency improvement, enhanced continuous delivery capabilities, and ease of development. It is causing transformations in enterprise system architecture, production methods, and business models, making cloud-native the shortest path to digital transformation for enterprises.
Tencent Cloud TKE (TKE) is a highly scalable high-performance container management service based on native Kubernetes, providing a container-centric platform. This article will build a TKE log analysis system based on Tencent Cloud ES Serverless service, achieving deployment in minutes and helping enterprises to construct a lightweight, observable analysis system while also realizing cost reduction and efficiency improvement.

Introduction to Tencent Cloud ES Serverless

I have carefully studied the Tencent Cloud<ES Serverless: From Beginner to Expert in One-stop Log Analysis> course. Simply put, the product's design philosophy, capabilities, and advantages are as follows:

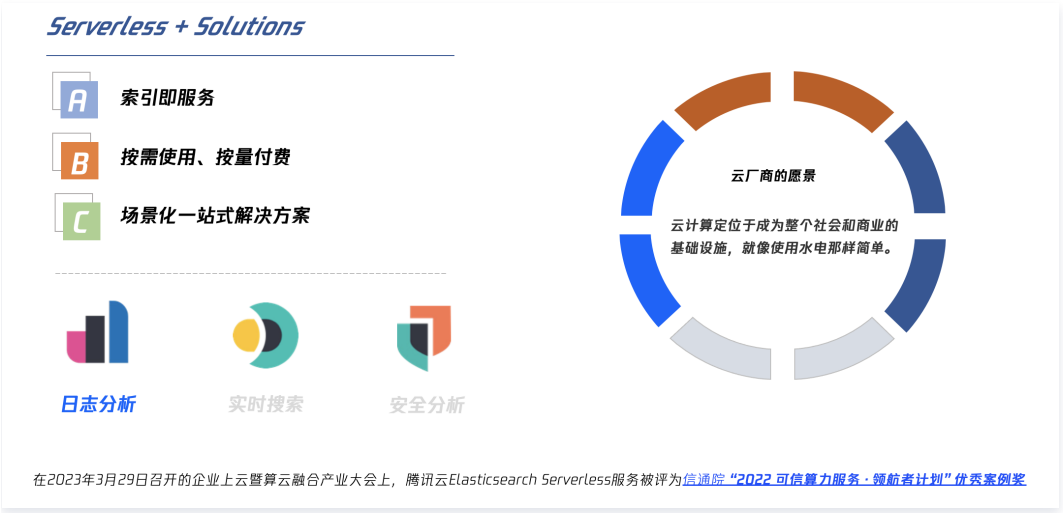
Design Philosophy

The goal of major cloud providers has always been to make computing power as accessible as tap water, available on-demand. The introduction of the Serverless concept has greatly influenced the development of cloud computing in recent years. In the early stages of cloud computing, the focus was on rapid and smooth migration to the cloud, for example, replacing IDC resources with CVM and CBS. Now, customer needs have evolved from merely migrating to the cloud to making effective use of the cloud. This means reducing operation and management costs while focusing more on business, improving efficiency, and achieving cost reduction and efficiency gains.
To help enterprises better focus on their business, the ES Serverless service consistently upholds the following goals in its design and development:

1. **Index-as-a-Service**: No cluster concept, allowing indexes to be created and used on demand.
2. **On-demand usage**: Pay-as-you-go billing, charging based on actual compute and storage resources used.
3. Beyond indexes, it is also a**Scenario-based One-stop Solution**: Offering out-of-the-box capabilities for each scenario to enhance overall usage efficiency. Currently, it supports the**Log Analysis** scenario.

It is worth mentioning that at the Enterprise Cloud and Computing Cloud Integration Industry Conference held on March 29, 2023, Tencent Cloud ES Serverless Service was awarded the "China Academy of Information and Communications Technology (CAICT) '2022 Trusted Computing Power Service · Pilot Program' Outstanding Case Award".

Search for the required CAM policy as needed, and click to complete policy association.



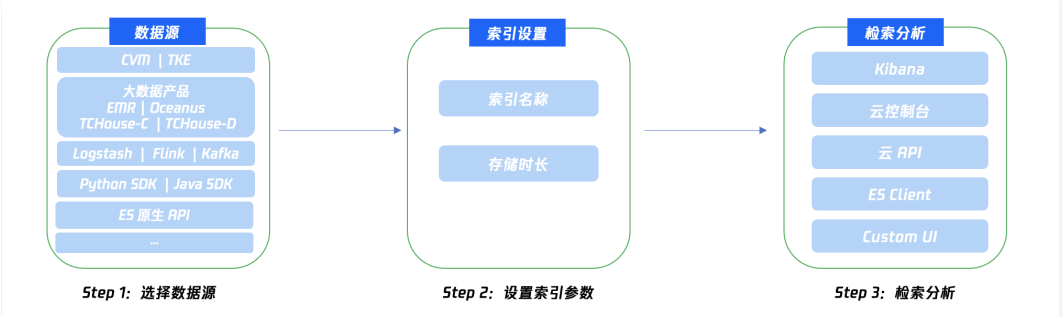
Design Concept

From a design perspective, ES Serverless Service adheres to the following points:

1. Improved usability: Besides offering an out-of-the-box, one-stop solution, it provides maintenance-free clusters, indexes, and data links.
2. Cost reduction: Utilizes proprietary compute-storage separation technology to achieve auto-scaling and eliminate redundant costs, with billing based solely on usage.
3. Enhanced stability: Clusters and indexes are maintained and optimized by the platform, preventing failures due to improper usage.
4. Seamless migration with full compatibility with native ES ecosystem and API.

Product Capabilities

From a product architecture perspective, geared towards log scenarios, ES Serverless service offers a **fully managed, automatically scalable** one-stop log analysis solution. When using it, users only need to focus on the data source and business logic such as retrieval and analysis. Traffic scheduling, link scheduling, and resource scheduling in between are all handled by the ES Serverless service, eliminating the need to worry about the underlying data link, message queue, cluster maintenance, and index configuration, etc. The platform provides end-to-end SLA guarantees.
In addition to supporting native ES API for writing, the console now supports one-stop data collection and analysis for cloud products like **CVM, TKE, EMR, Tencent Cloud Data Warehouse TCHouse** , and also supports delivering data to ES Serverless service indexes through Logstash, Flink, and Kafka. In short, with just 3 steps, we can implement a one-stop log analysis scenario.
In terms of index management, it provides configuration management, metrics monitoring, user management, and alert management capabilities, enabling us to quickly complete data applications. For search and analysis capabilities, besides supporting native Kibana capabilities, the ES Serverless service also embeds core Kibana functionalities within the console, allowing for swift search and analysis without needing external links.
Search for the required CAM policy as needed, and click to complete policy association.



Advantages

Automatic Elasticity: Index-level automatic AS to seamlessly handle sudden traffic surges, ensuring business continuity while reducing operational and management costs caused by traffic peaks and troughs in log analysis and observability scenarios.

Completely Ops Free: Built-in automatic shard optimization, intelligent lifecycle management, and self-healing capabilities allow users to create and use indexes as needed without worrying about underlying resource configuration, cluster scaling, and index settings, making the entire process completely maintenance-free.
High Flexibility and Ease of Use: Provides end-to-end capabilities from data ingestion to data management and data analysis, significantly lowering the barrier to cloud adoption and enabling quick business implementation within minutes.
Low Cost: Self-developed low-cost, high-performance, high-availability storage-compute separation architecture, with billing based on actual usage and storage, achieving dynamic matching of business load and resources on a pay-as-you-go basis, reducing redundancy costs from idle resources and significantly lowering overall costs.
Open Integration: Fully compatible with the Elastic Stack ecosystem, retaining users' original usage habits and enabling seamless migration to support rapid cloud adoption. Also integrates cloud data sources (such as CVM, TKE), lowering data access barriers and allowing quick business implementation within minutes.
Stable and Reliable: Cluster configuration and read/write performance are optimized uniformly by the backend, reducing issues caused by misuse, enhancing stability, and safeguarding business operations.

Log Analysis Practice

Log Collection

The ES Serverless service now provides one-stop TKE log collection in the console. Simply select the container cluster to collect, set the relevant collection parameters, and set the data retention duration to quickly complete TKE log access. Refer to the following:
1. In the ES Serverless console, select **TKE TKE:**

Search for the required CAM policy as needed, and click to complete policy association.

2. Select the TKE cluster. Currently, log filtering based on namespace, Pod tag, container name, or host path is supported:

Search for the required CAM policy as needed, and click to complete policy association.

3. To parse the reported logs, you can set collection and parsing. Currently, full text logs, JSON formats, and delimiter parsing are supported. If you have more complex parsing requirements, you can write your own processors for custom parsing. Here, we select

full text logs :

Parsing Mode	Description
Full Text Log	No key-value extraction is performed on the log data; the log content will be stored in a field named "message", and you can perform search and analysis using automatic word segmentation capabilities.
JSON	For logs in standard JSON format, extract the corresponding fields based on the log's key:value pairs.
Separator	For logs with fixed delimiters, extract key-value pairs based on the specified delimiter. The delimiter can be a single character or a string and can be selected or input in the console.

Search for the required CAM policy as needed, and click to complete policy association.

4. Set the index name and data retention duration, and also set the time field. This field refers to a date type field in the actual data. Setting this field helps improve query efficiency.

Search for the required CAM policy as needed, and click to complete policy association.

With these simple steps, we can quickly collect container logs into the ES Serverless index. Click **Confirm creation** , and wait for about 5–10 seconds.

Report Construction

ES Serverless has now integrated log retrieval, development tools, and monitoring and alert capabilities in the console, which can help us quickly implement retrieval and analysis. As a second-tier "senior" ES user, I am accustomed to using Kibana for analysis, which is convenient for sharing and exchanging through external links.

Monitoring and Alarms

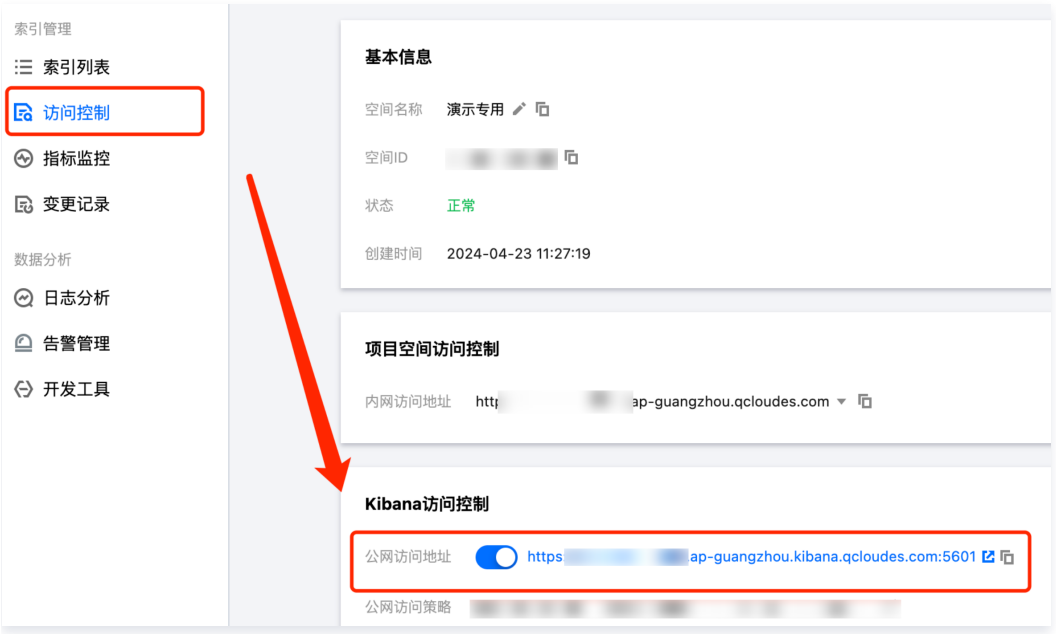
To ensure timely discovery and processing of error logs, we can first configure monitoring and alerts in the console. For example, we can quickly identify anomalies by counting the number of logs from the /var/log/error.log path within five minutes. When the threshold is exceeded, alerts can be sent to our email and WeCom.

Search for the required CAM policy as needed, and click to complete policy association.

Observable Analysis

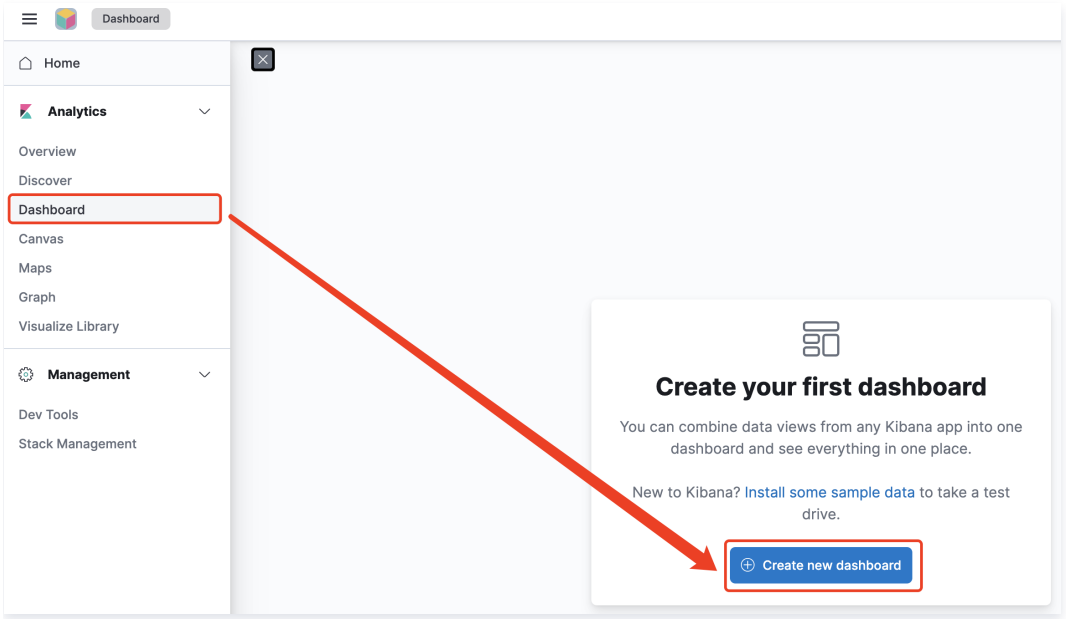
1. First, log in to Kibana. We find Kibana's public network access address. To ensure the security of public network access, we need to set our IP addresses in the public network access policy before accessing.

Search for the required CAM policy as needed, and click to complete policy association.



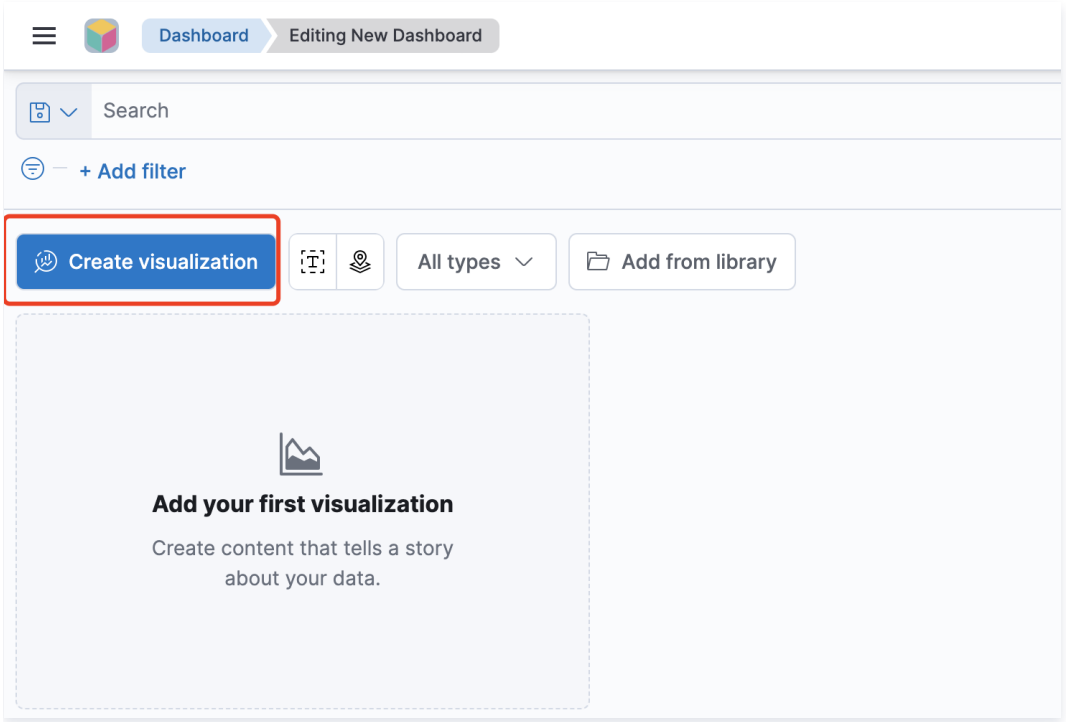
2. Click Dashboard. We can quickly create charts:

Search for the required CAM policy as needed, and click to complete policy association.



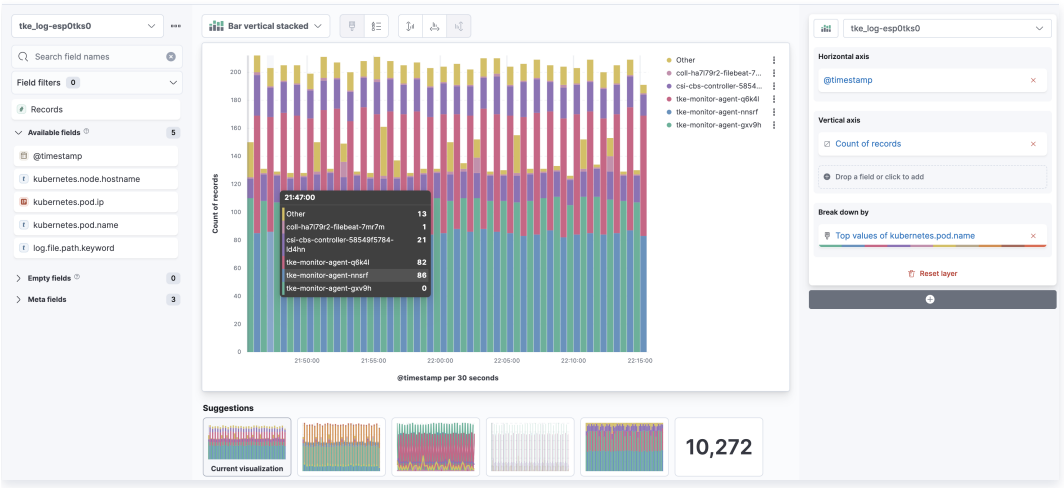
3. click **Create visualization:**

Search for the required CAM policy as needed, and click to complete policy association.



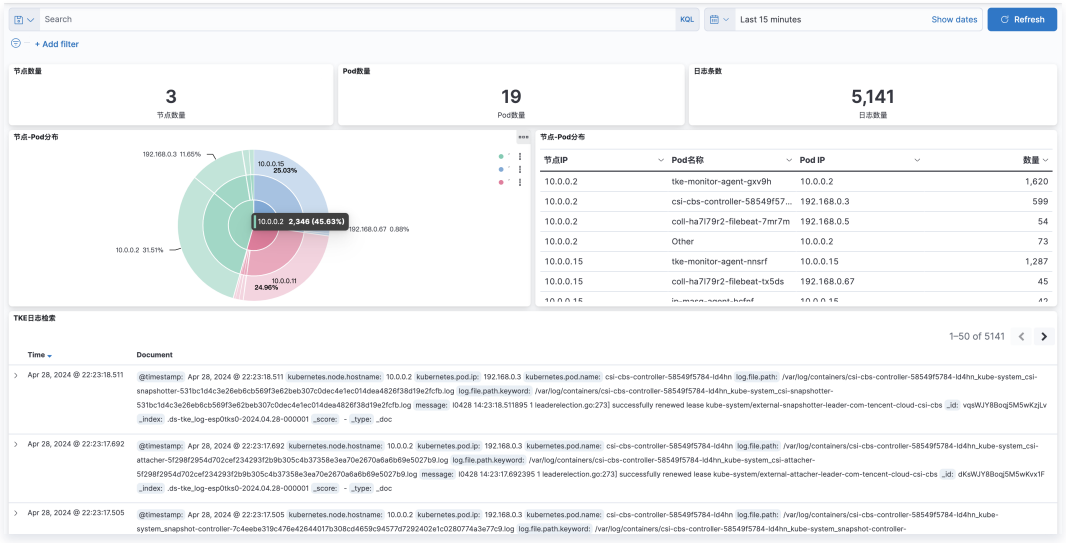
4. The Lens feature of Kibana is very easy to use and powerful. Simply move the cursor to the field on the left and drag it to the middle to generate a chart. At the same time, other chart suggestions are provided below, making it convenient for us to quickly analyze. As shown in the figure below, we dragged two fields, **@timestamp** and **pod.name** to the middle, and Lens quickly generated a bar chart. Through this chart, we can quickly understand the number of logs reported by each Pod in each time period.

Search for the required CAM policy as needed, and click to complete policy association.



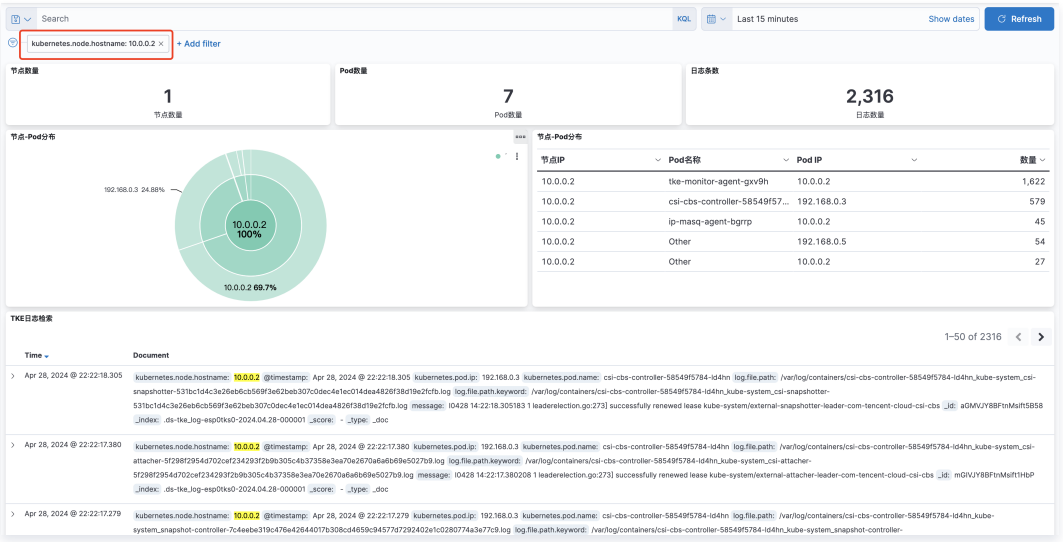
4.1 Firstly, the first line presents various overview data, such as the number of nodes, the number of Pods, and the number of log entries; the second line displays the node–Pod distribution; the third line shows specific log data. The overall structure follows **Overview** > **Multidimensional Distribution** > **Log**, drilling down layer by layer.

Search for the required CAM policy as needed, and click to complete policy association.



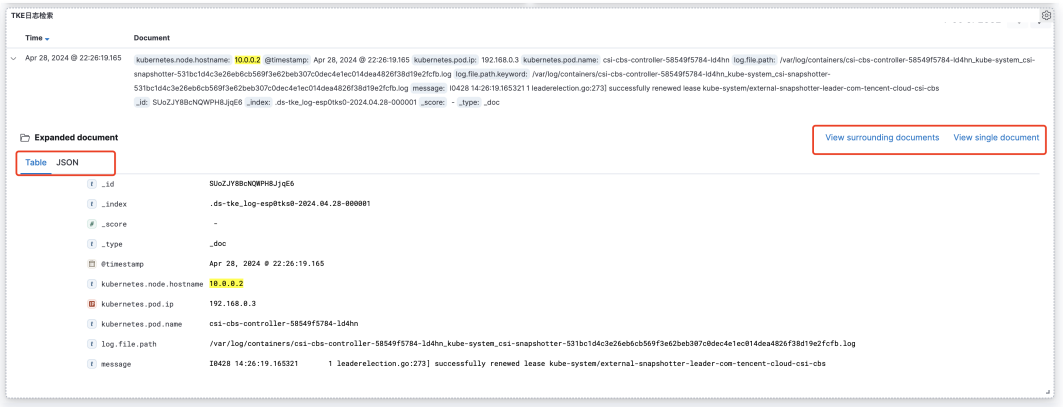
4.2 From the above figure, we can see that the log entries from the node 10.0.0.2 account for nearly half. We can focus on analyzing this node. Simply click on the area of the node in the **Node–Pod** Multidimensional Analysis Chart, and the filters will be automatically generated. All relevant data for this node will be filtered out and highlighted in the logs.

Search for the required CAM policy as needed, and click to complete policy association.



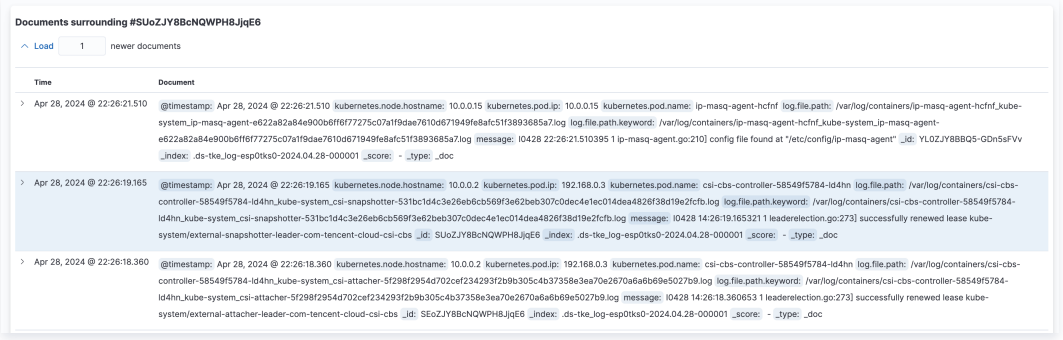
4.3 When you want to change the log display style, you can choose between Form Mode and JSON Schema below. If you notice a current log issue that might require context for further analysis, click **view surrounding documents**.

Search for the required CAM policy as needed, and click to complete policy association.



4.4 For example, view the log before and after the current one:

Search for the required CAM policy as needed, and click to complete policy association.



Permission Control

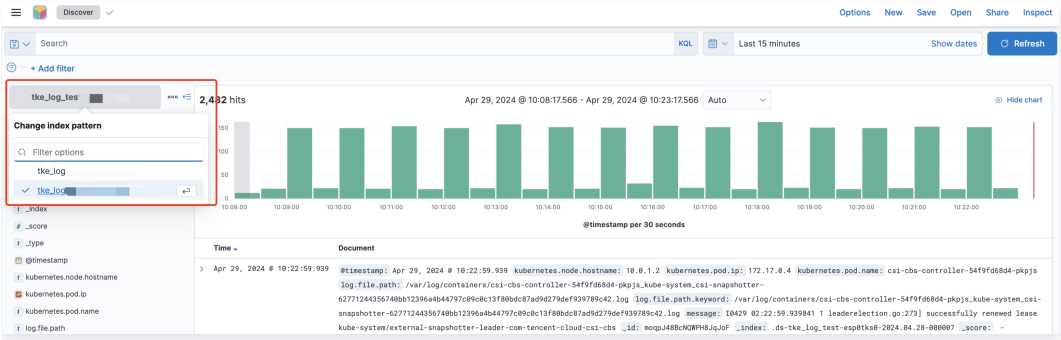
Of course, in actual business scenarios, we often need to share permissions with other team members. At this point, permission control becomes crucial. For instance, you might assign read permissions for certain indices to Colleague A, while giving read-write permissions for all indices to Colleague B.

Regarding this scenario, ES Serverless takes full consideration for users. Enter the ES console, go to the User Management page, and click **Create User** to start creating a user.

Search for the required CAM policy as needed, and click to complete policy association.

Currently, the supported Permission Types are **Read-only** and **read-write**. The Permission Scope includes all indices or specified indices. After filling out the details, click **Confirm**.
Search for the required CAM policy as needed, and click to complete policy association.

For example, if I set the **hd_man** user to have only read permissions for TKE-related indices, after completing the setup, if I log in to Kibana with this user, I can only view TKE log-related indices
Search for the required CAM policy as needed, and click to complete policy association.



Summary

This article briefly introduces implementing one-stop TKE Observable Analysis based on Tencent Cloud ES Serverless service. In practice, I completed log integration and observable analysis report construction within minutes. From this perspective, the ES Serverless service is truly lightweight and barrier-free. Of course, after seeing all of this, it is time to show the benefits I managed to get from the editor. The link is as follows:

- 1 Yuan Purchase for Free Trial Coupons and Resource Packs:

Tencent Cloud Big Data Elasticsearch Serverless Service Officially Launched

- External Developer Course:

ES Serverless One-Stop Log Analysis from Beginner to Advanced Learning Course_ES Serverless One-Stop Log Analysis from Beginner to Advanced Video Tutorials - Tencent Cloud Developer Community

If you are interested in this product, you can scan the code to join the communication group. The ES Serverless service product team is very nice

Search for the required CAM policy as needed, and click to complete policy association.

RAG Practice: Build Your Exclusive AI Assistant in Ten Minutes with Tencent Cloud ES and HunYuan

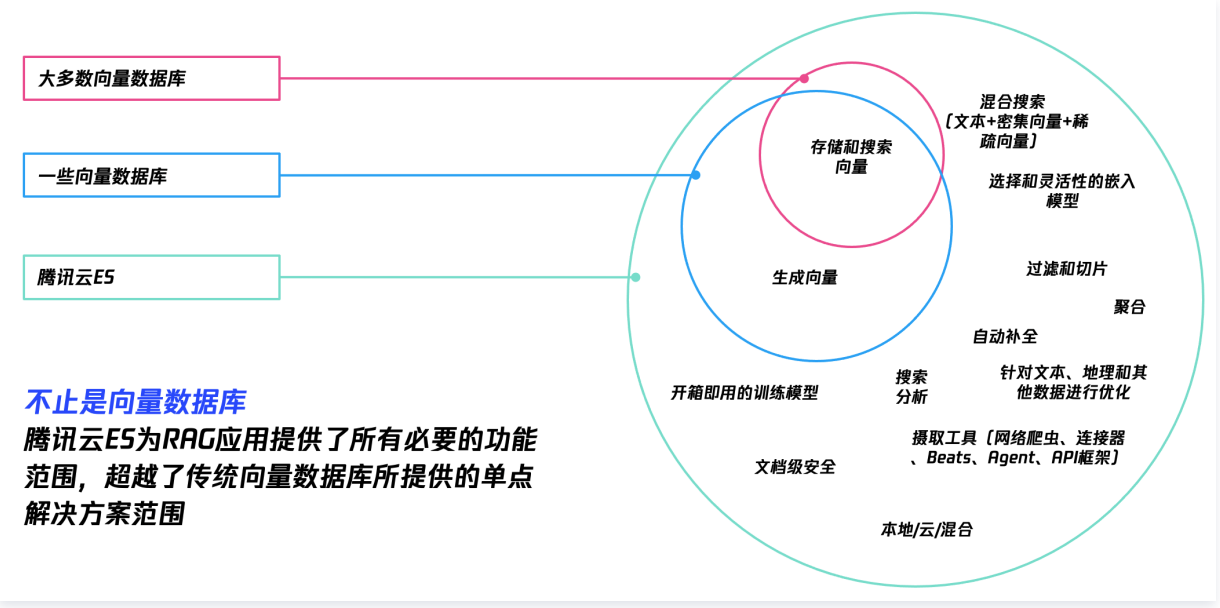
Last updated：2024-10-25 09:37:20

Background Overview

With the continuous development of data intelligence technology, content generation technology represented by AIGC driven by large language models (LLM) has become an indispensable part of enterprise data intelligence capabilities. However, traditional content generation technologies face issues such as untimely information updates, lack of vertical domain knowledge, and model hallucinations. Promoting the application of large models across various industries and business scenarios is a common concern, and Retrieval-Augmented Generation (RAG) technology offers an effective solution, becoming a major trend in the data intelligence era. RAG is a technical solution that combines retrieval and large language model content generation. By referencing external knowledge bases, it retrieves knowledge upon user input queries, then the model generates user responses based on credible knowledge. RAG has high interpretability and customization capabilities, significantly reducing model hallucinations, and is applicable in various natural language processing tasks such as question-answering systems, document generation, and intelligent assistants. This article introduces the Tencent Cloud ES one-stop RAG solution and demonstrates how to quickly build RAG applications by combining Tencent Cloud ES with HunYuan large models.

Tencent Cloud ES One-Stop RAG Solution

Tencent Cloud ES is a fully managed cloud service for massive data search and analysis, featuring a high-performance self-developed kernel, integrated with X-Pack. It supports easy cluster management through features like autonomous indexing, storage-compute separation, and cluster inspection. It also supports a Serverless mode that is maintenance-free, auto-scalable, and pay-as-you-go. Tencent Cloud ES leverages the operational experience from Tencent’s massive internal and external services and has optimized the ES kernel for cost, performance, stability, and scalability. It is the top open-source contributor team in the Asia-Pacific region. With Tencent Cloud ES, you can efficiently build online search, vector search, log analysis, OPS monitoring, intelligent Q&A, and other services.
In terms of RAG, Tencent Cloud ES supports one-stop vector retrieval, text + vector hybrid search, rerank fusion, integration with large models, high-performance GPU inference, and field-level permission control. A significant number of optimizations have been made to query performance, effectively improving data retrieval efficiency.
Search for the required CAM policy as needed, and click to complete policy association.



Additionally, as the first public cloud platform in China to provide an end-to-end one-stop solution from natural language processing to vector generation/storage/retrieval and integration with large models, Tencent Cloud ES has participated as a core editor in the RAG standard formulation organized by the CAICT and has become the first company to pass the authoritative RAG certification.

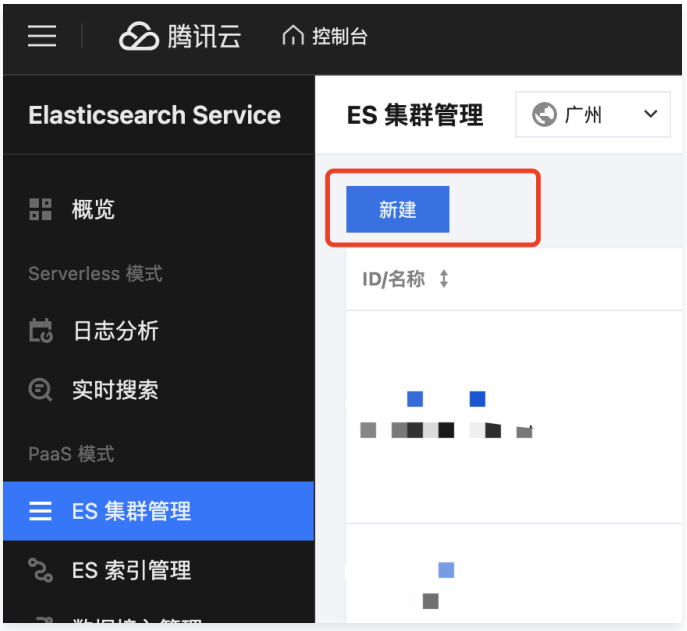
Search for the required CAM policy as needed, and click to complete policy association.

知识库构建能力	知识检索能力	内容生成能力	质量评估能力	平台能力
数据读取	查询优化	Prompt优化	评估指标	用户管理
数据预处理	检索能力	大模型生成	评估方法	数据权限管理
内容增强	检索结果优化	生成内容优化	评估数据集	模型权限管理
索引构建				资源及任务管理
知识库管理				日志管理

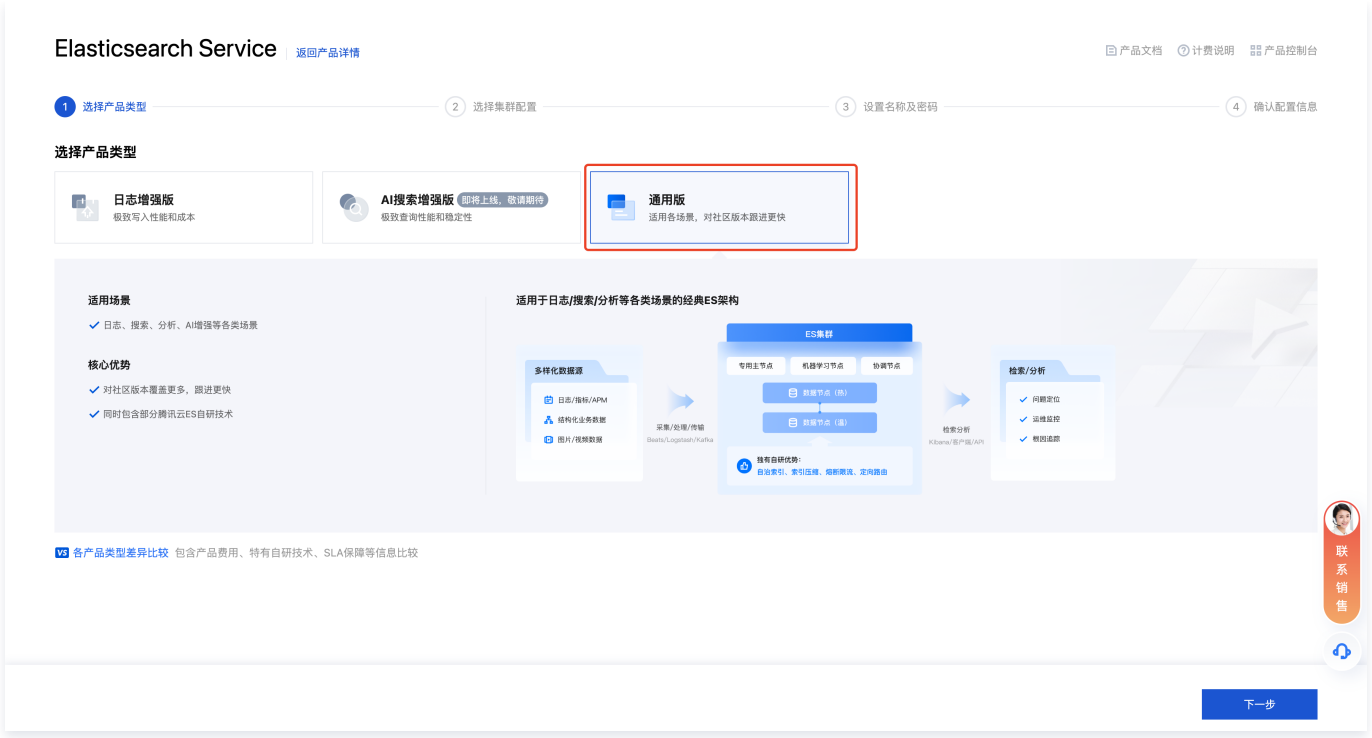
AI Assistant construction

Purchase ES Cluster

1. log in to Tencent Cloud ES Console, go to ES Cluster Management page, click **New**.



2. Enter the New page and select the product type: **General Edition**.



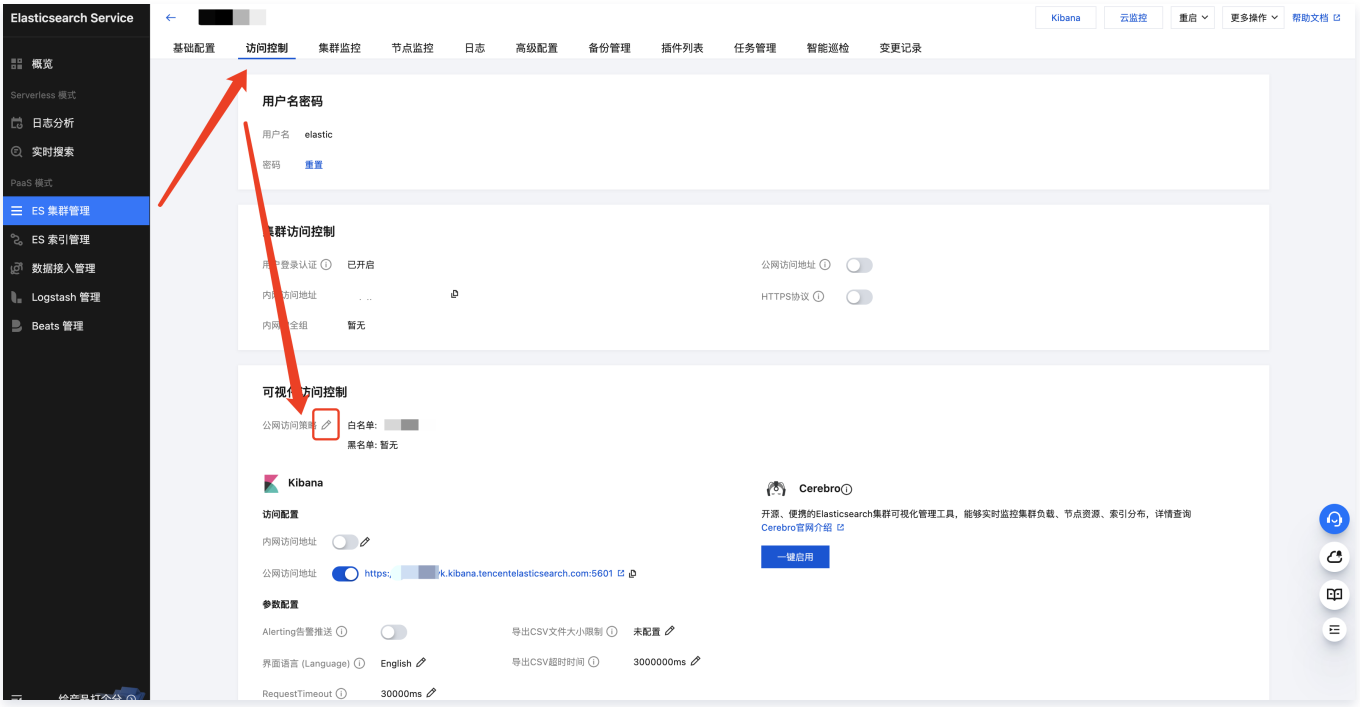
3. Select version: Billing mode is **pay-as-you-go**, ES Version is **8.13.3**, Business Feature is **Platinum Edition**.

4. ES Node Configuration, for test environment, you can choose **ES.S1(2-core 4GB)**, Number of nodes is **2**, Disk is **Balanced SSD**, Disk capacity is **20GB**.

5. Other configurations can be set to default.

log in to Kibana

1. Access Kibana, set Public Network Access Policy.



2. Click Kibana Public Network Access Address to visit Kibana.

Deploy Embedding Model

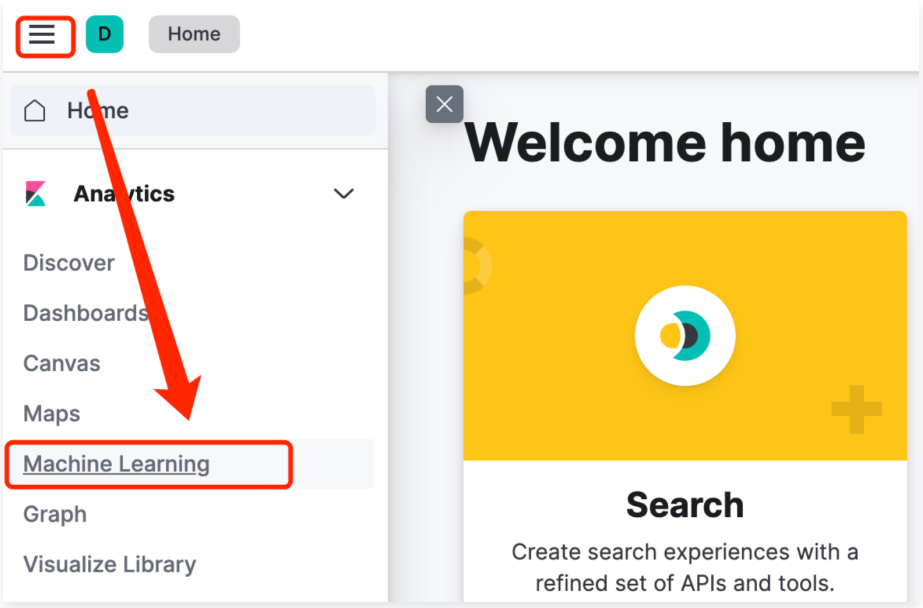
After the cluster purchase is complete, go to Kibana to deploy the Embedding Model, create Knowledge Base Indexing and Vectorization Pipeline.

1. Enable **Outbound Access for Nodes**. Only the data node needs to be enabled. If there is a dedicated primary node, only the dedicated primary node needs to be enabled. (This feature is allowlist-only. Please contact Work Order for handling.)

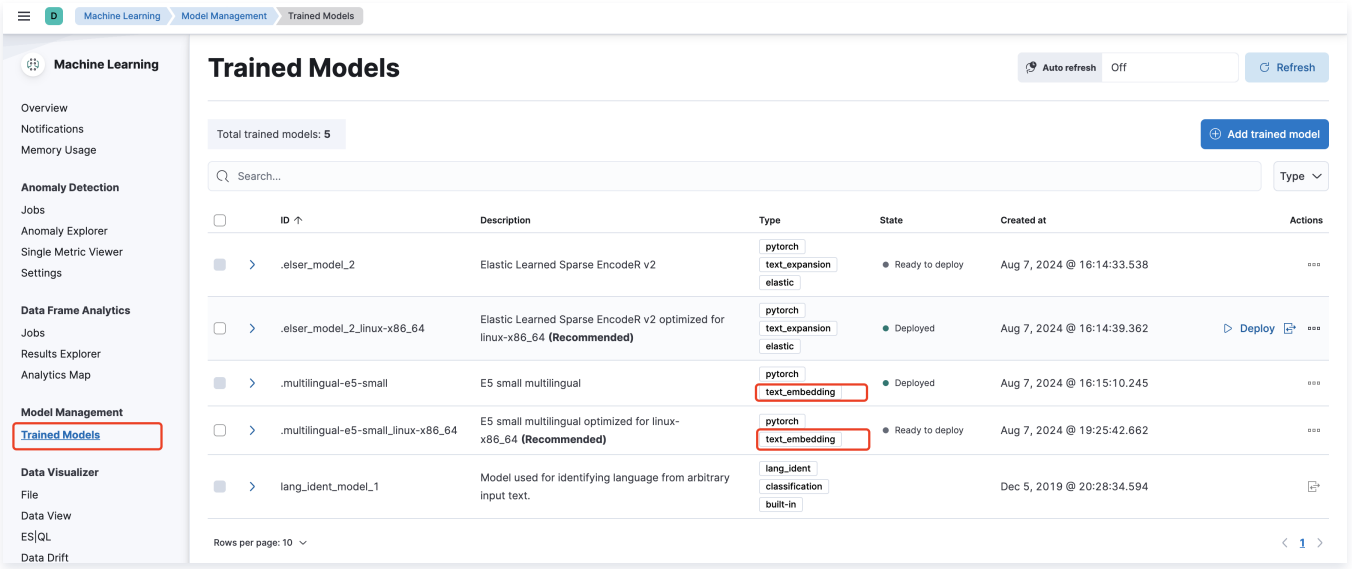
> **Note:**
> For uploading custom models or models from third-party platforms (e.g., Huggingface), refer to the documentation.

Search for the required CAM policy as needed, and click to complete policy association.

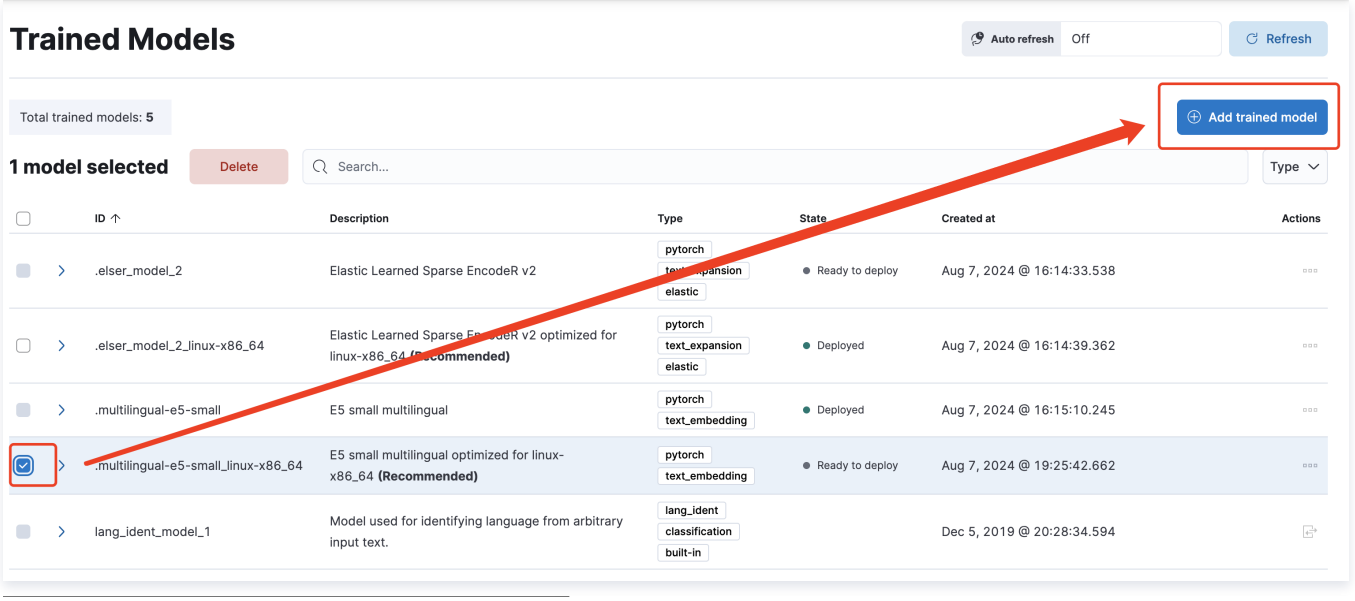
2. After you log in to Kibana, find the **Machine Learning** feature in the left navigation bar.

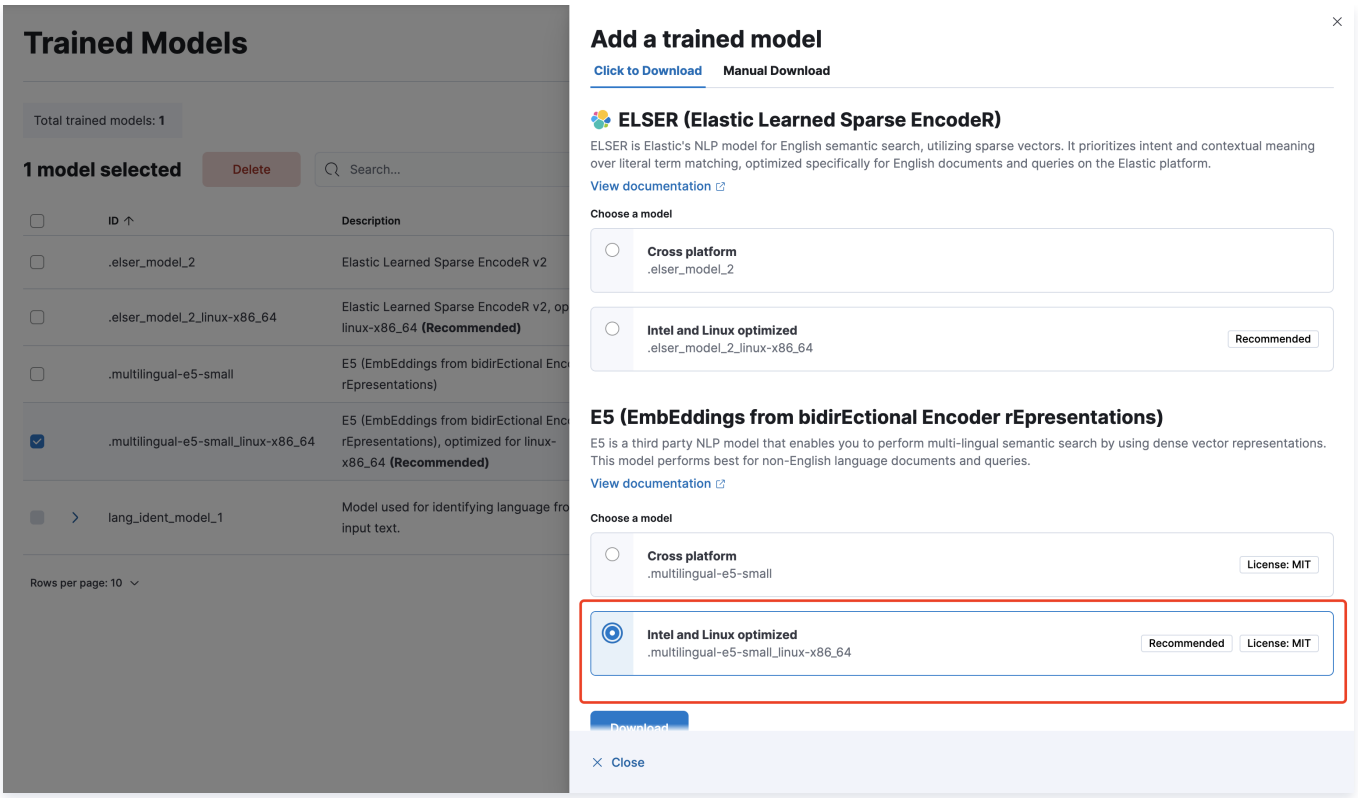


3. Go to the Model Management page and find the model of type **text_embedding**.

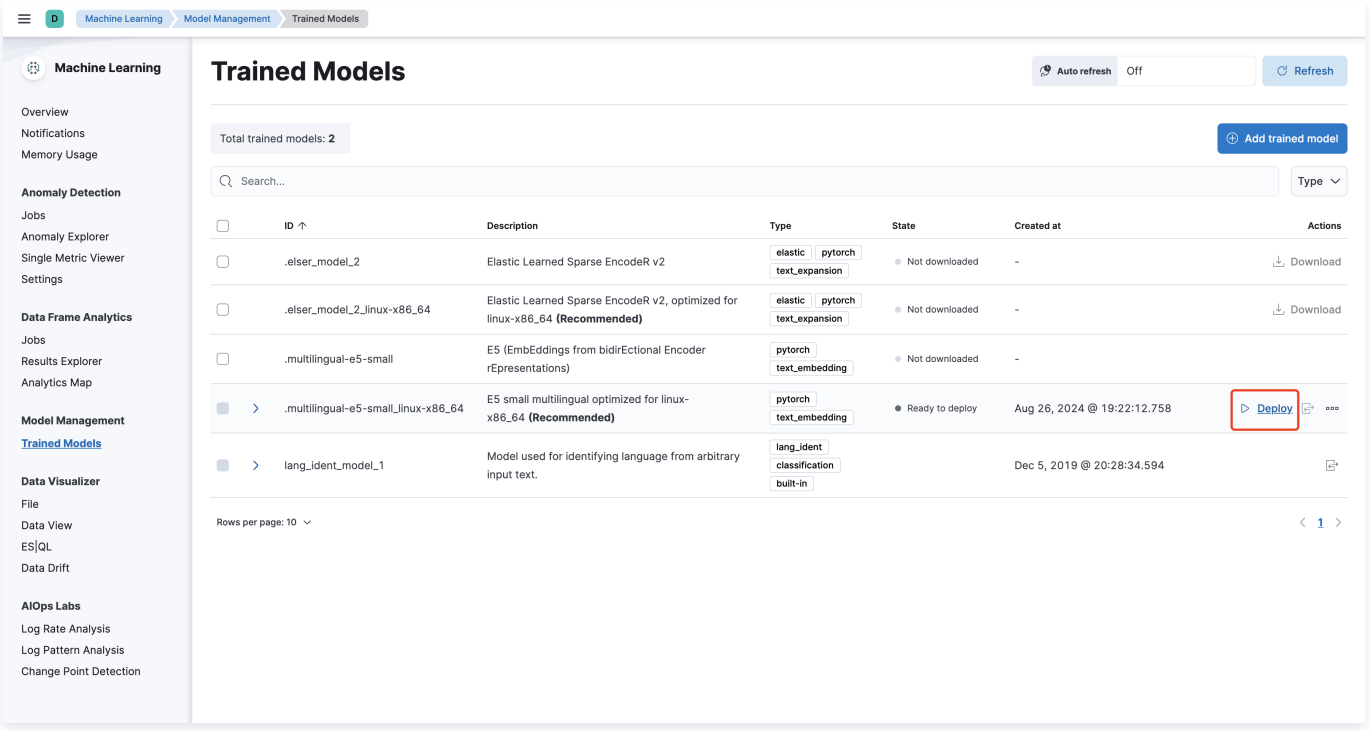


4. If it is in an undownloaded state, select the model and click **Add trained model**. For this demonstration, we are using the **.multilingual-e5-small_linux-x86_64** model.

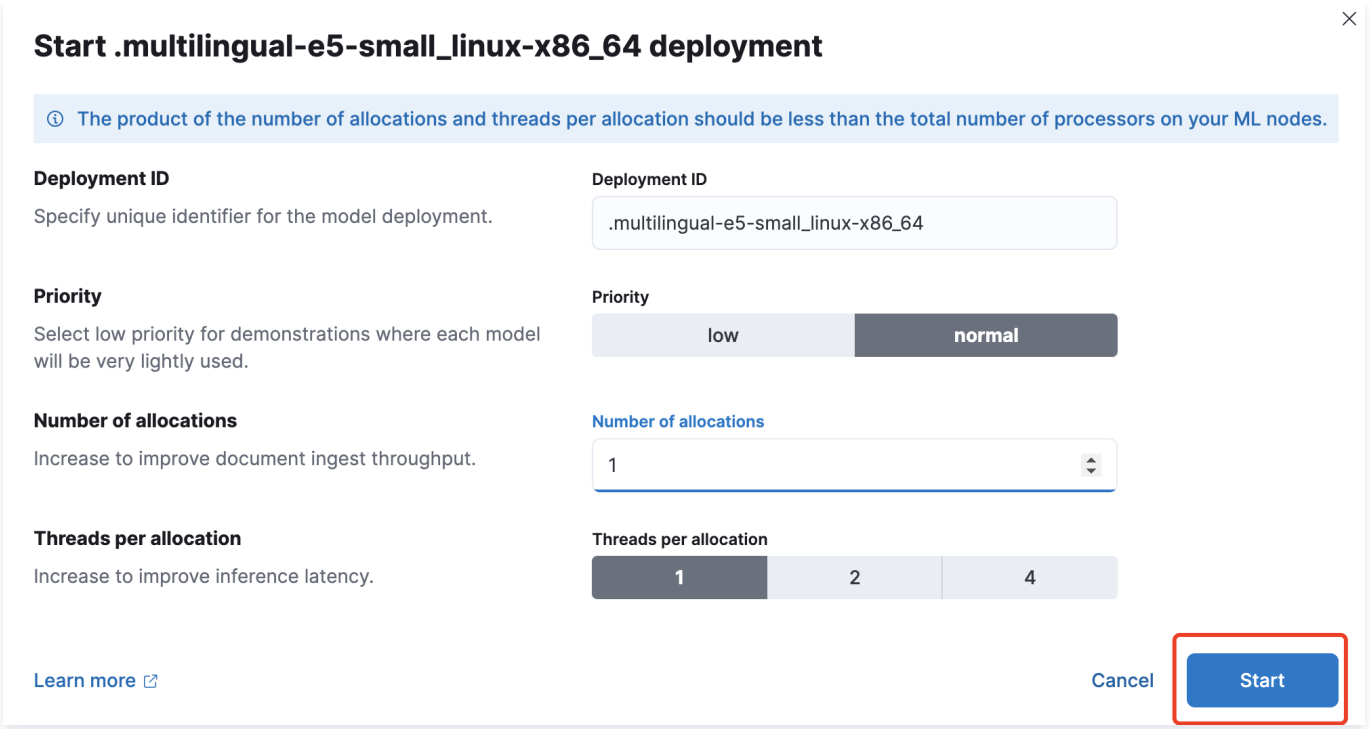




5. After the download is complete, click **Deployment**.

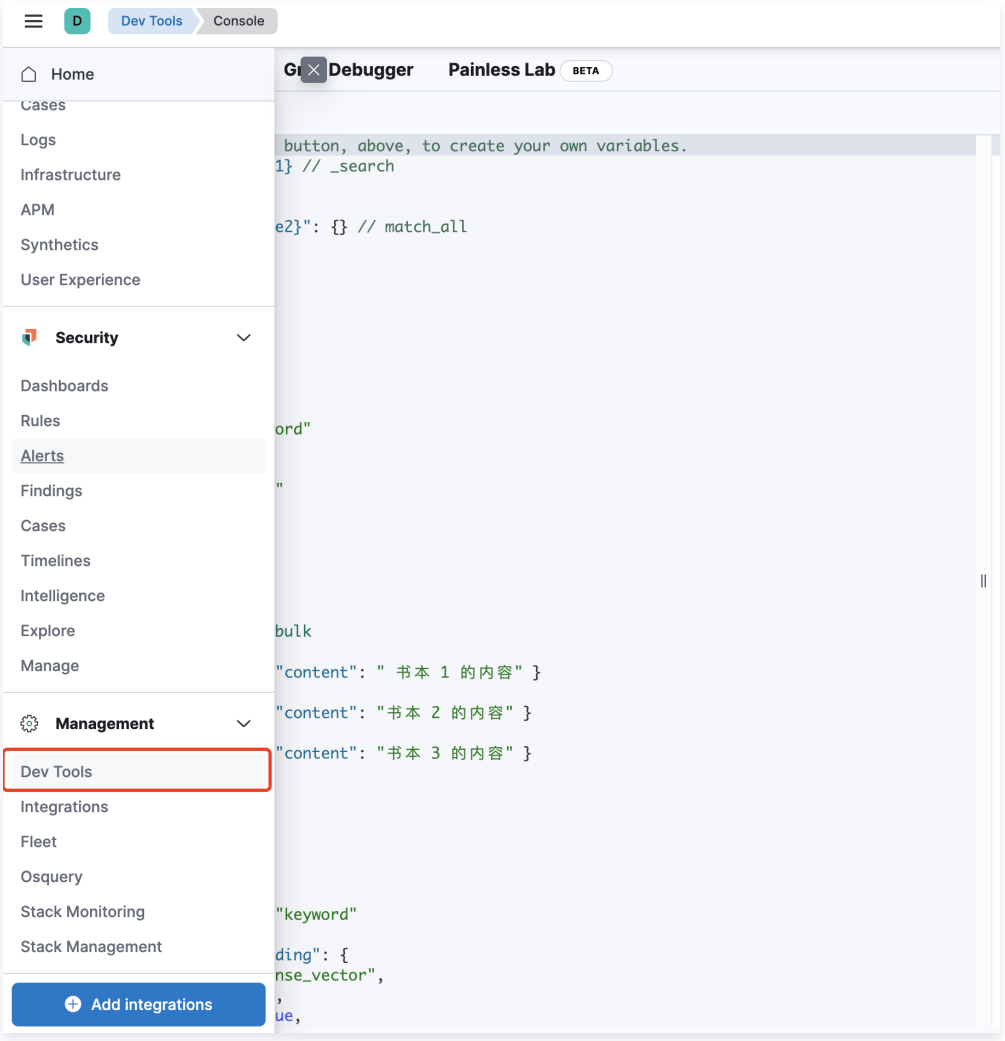


6. For a quick experience, use the default values for configuration.



Create index and vector pipeline

1. Click to enter the **Dev Tools** page.



2. Create Knowledge Base Index

   index-name is the index name; you can name it as needed.

```
PUT /index-name
{
  "mappings": {
    "properties": {
    "title": {
        "type": "keyword"
      },
      "content": {
        "type": "text"
      },
      "url": {
        "type": "keyword"
      }
    }
  }
}
```

3. Create an inference pipeline for data vectorization when writing data.

```
PUT /_ingest/pipeline/index-name-pipeline
{
  "description": "Text embedding pipeline",
  "processors": [
    {
      "inference": {
        "model_id": ".multilingual-e5-small_linux-x86_64",
        "input_output": [
          {
            "input_field": "content",
            "output_field": "content_embedding"
          },
          {
            "input_field": "title",
            "output_field": "title_embedding"

          }
        ]
      }
    }
  ]
}
```

The above pipeline will store the vectorized content of fields content and title in new fields using the .multilingual-e5-small_linux-x86_64 model.

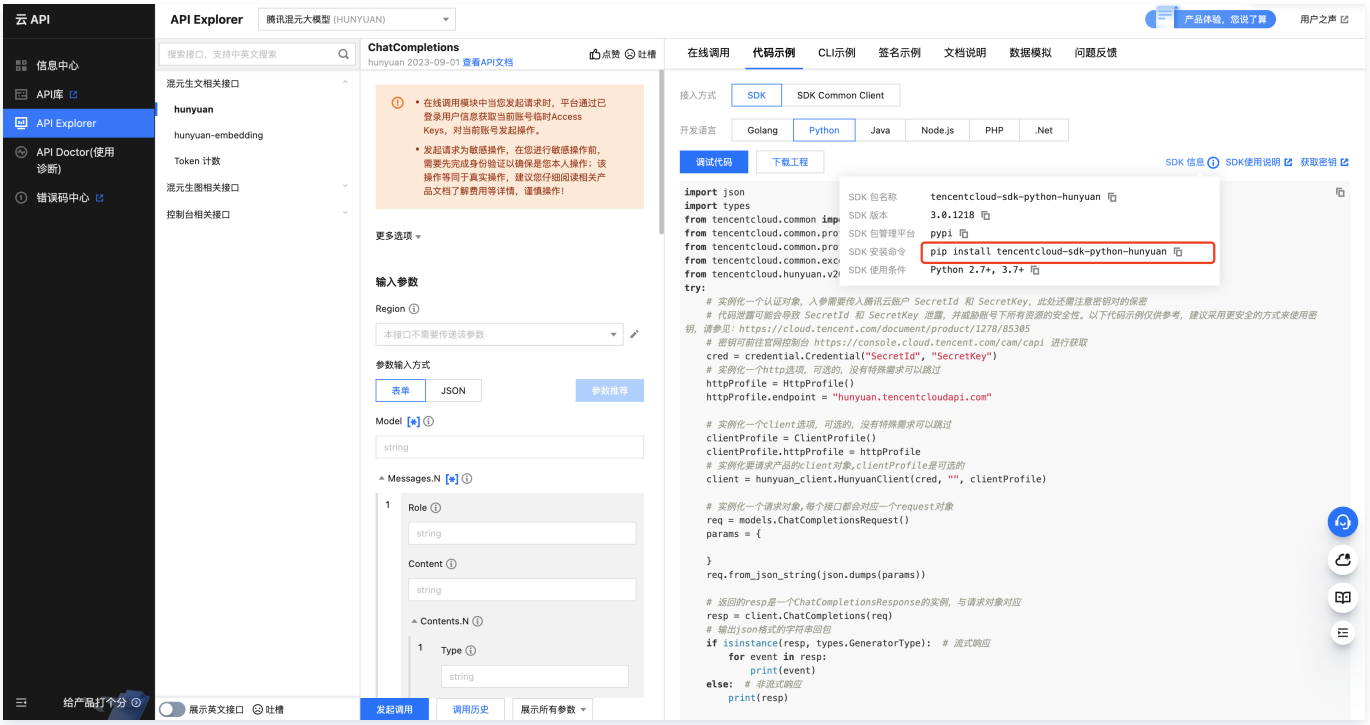
Write Knowledge Base Data

Use the Bulk API to batch write data. Replace the content of title, content, and url with your actual knowledge base data.

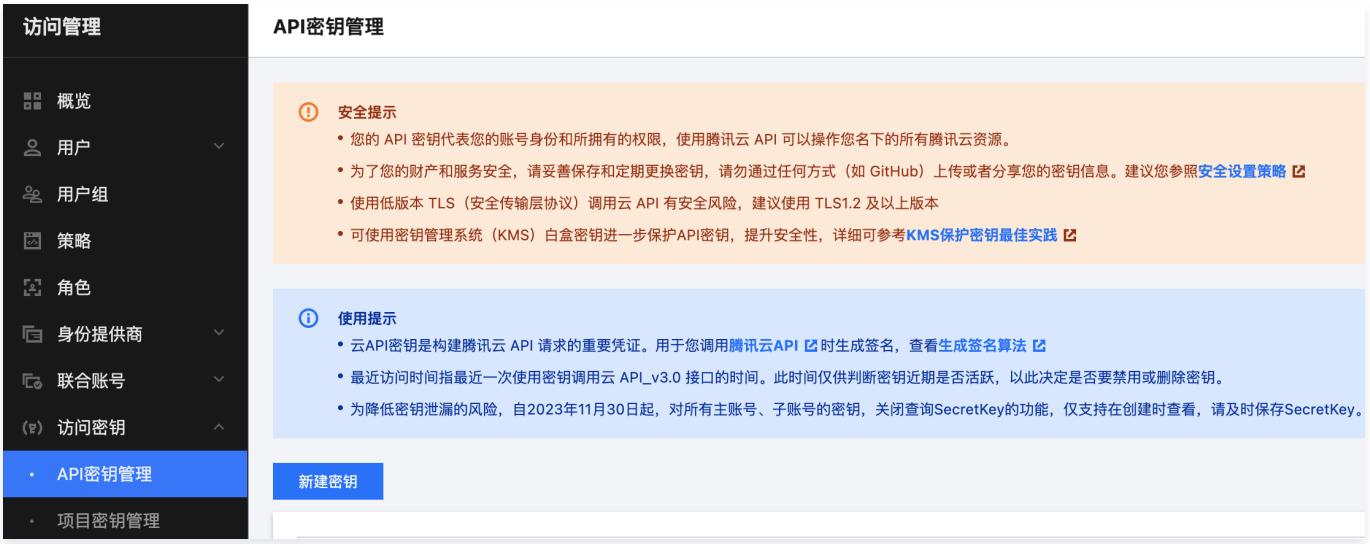
```
POST index-name/_bulk?pipeline=index-name-pipeline&refresh
{ "index" : {} }
{ "title" : "Title 1","content": "Content 1","url": "https:url1"  }
{ "index" : {} }
{ "title" : "Title 2","content": "Content 2","url": "https:url2"  }
```

Invoke Hunyuan Model

1. Name the Python file as **hunyuan.py**.
2. Go to the API Explorer page and install the relevant libraries, as follows:



3. Go to the API Key Management page to obtain SecretID and SecretKey.




```
import json
from tencentcloud.common import credential
from tencentcloud.common.profile.client_profile import ClientProfile
from tencentcloud.common.profile.http_profile import HttpProfile
from tencentcloud.common.exception.tencent_cloud_sdk_exception import TencentCloudSDKException
from tencentcloud.hunyuan.v20230901 import hunyuan_client, models

def hunyuan_gpt(system_prompt, content):
    try:
        # You can obtain the keys from the official console at https://console.cloud.tencent.com/cam/capi
        cred = credential.Credential("SecretId", "SecretKey")
        # Instantiate an HTTP option, optional, can be skipped if not needed
        httpProfile = HttpProfile()
        httpProfile.endpoint = "hunyuan.tencentcloudapi.com"
```

```
        # Instantiate a client option, optional, can be skipped if not needed
        clientProfile = ClientProfile()
        clientProfile.httpProfile = httpProfile

        # Instantiate the client object for the product you want to request, clientProfile is optional
        client = hunyuan_client.HunyuanClient(cred, "", clientProfile)

        # Instantiate a request object, each interface will correspond to a request object
        req = models.ChatCompletionsRequest()
        params = {
            "Model": "hunyuan-standard",
            "Messages": [
                {"Role": "system", "Content": system_prompt},
                {"Role": "user", "Content": content}
            ]
        }
        req.from_json_string(json.dumps(params))

        # The response returned is an instance of ChatCompletionsResponse, corresponding to the request
object
        resp = client.ChatCompletions(req)

        return resp

    except TencentCloudSDKException as err:
        return str(err)
```

Intelligent Q&A System Construction

1. Install streamlit.

```
pip install streamlit
```

2. Obtain the ES access address.

   The username is elastic, and the password is set when creating the cluster. When testing on a local Mac, you can enable public access. For actual production, it is recommended to use the intranet access address.

3. Run the following code, name the file **demo.py** (it should be in the same directory as **hunyuan.py**).

```
import streamlit as st
from elasticsearch import Elasticsearch
from hunyuan import hunyuan_gpt

es_client = Elasticsearch(
```

```
    "ES cluster access address",basic_auth=("username", "password"))

def get_elasticsearch_results(query):
    es_query = {
        "knn": {
            "field": "content_embedding",
            "num_candidates": 100,
            "query_vector_builder": {
                "text_embedding": {
                    "model_id": ".multilingual-e5-small_linux-x86_64",
                    "model_text": query
                }
            }
        },
        "query":{
            "match":{
                "content":query
            }
        },
        "rank":{
            "rrf":{
                "window_size":100,
                "rank_constant":20
            }
        }
    }
    result = es_client.search(index="index-name", body=es_query)
    return result["hits"]["hits"]

def create_hunyuan_prompt(results,question):
    context = ""
    for hit in results:
        source_field = hit["_source"]["content"]
        hit_context = source_field
        context += f"{hit_context}\n"
    prompt = f"""
Instructions:

Answer this question: {question}
In your answer, you can only refer to the document {context} to generate the answer. If an answer cannot be
produced, please reply: Sorry, I cannot answer this question.

"""
    return prompt

def generate_hunyuan_completion(user_prompt, question):
    response = hunyuan_gpt(user_prompt, question)
    return response.Choices[0].Message.Content

def format_result(hit):
    title = hit["_source"]["title"]
    return f'[{title}]'

def main():
    st.title("My Dedicated AI Assistant")

    # Create input box and query button
    question = st.text_input("Please enter your question:")
    if st.button("Search and Generate Answer"):
        # Retrieve Elasticsearch results and create HunYuan prompt
        elasticsearch_results = get_elasticsearch_results(question)
        user_prompt = create_hunyuan_prompt(elasticsearch_results,question)
```

```
        system_prompt = f"""
You are an assistant for a Q&A task. Use the presented context to answer the question truthfully and
factually.
"""

        # Generate the answer using the HunYuan model
        openai_completion = generate_hunyuan_completion(system_prompt, user_prompt)

        # Display results
        st.write(openai_completion)

        # Show Elasticsearch search results
        st.write("Documents referred by the large model:")
        for hit in elasticsearch_results:
            st.markdown(format_result(hit))

if __name__ == "__main__":
    main()
```

4. In the directory of the above Python file, run the system using the following command:

```
streamlit run demo.py
```

5. The generated interface is as follows:

Summary

This article introduces the Tencent Cloud ES one-stop RAG scheme and demonstrates how to quickly build RAG applications by combining Tencent Cloud ES with the HunYuan Large Model. Leveraging its extensive experience accumulated in traditional PB-level logging and massive search scenarios, Tencent Cloud ES successfully applies years of performance optimization, index building, and operation management experience to the RAG field through deep reconstruction of the underlying system. It actively explores the integration of vector recall and traditional search technologies to fully exploit the advantages of both, providing users with a more accurate and efficient search experience. Looking ahead, Tencent Cloud ES will continue to deepen its efforts in the intelligent retrieval field, continuously improving in cost, performance, stability, and other aspects to help customers reduce costs and increase efficiency while achieving sustainable business growth. Stay tuned!