Elasticsearch Service

Practical Tutorial

Product Documentation

Service Statement

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

Contents

Practical Tutorial
Data Migration and Sync
Migrate Data

Last updated：2024-12-02 09:49:06

If you want to migrate your Elasticsearch cluster built in Tencent Cloud or purchased from another cloud vendor to Tencent Cloud ES, you can choose a suitable migration solution based on your business needs. If your business can be paused for write operations, you can migrate your data using the following tools:

COS snapshot

Logstash

elasticsearch-dump

The migration methods are compared as follows:

Migration Method	Use Case
COS snapshot	Scenarios with high volumes of data (gigabyte, terabyte, or petabyte scale) Scenarios with higher requirement for migration speed
Logstash	Scenarios where full or incremental data is migrated with a lower requirement for real-timeliness Scenarios where data to be migrated needs to be simply filtered with query operations of Elasticsearch Scenarios where data to be migrated requires complicated filtering and processing Scenarios where data is migrated across versions with a large gap, such as migration from 5.x to 6.x or 7.x
elasticsearch-dump	Scenarios with small volumes of data

COS Snapshot

COS snapshot-based migration is to use the snapshot API of ES for migration. Its basic process is to create an index snapshot from the source Elasticsearch cluster and then restore it to the target ES cluster. If you migrate data with snapshot, you should pay special attention to the Elasticsearch version:

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

Creating repository in source cluster

Before creating a snapshot, you must create a repository, which can contain multiple snapshot files. There are several types of repositories as detailed below:

fs: a shared file system where snapshot files are stored.

url: the URL path to a specified file system, which supports HTTP, HTTPS, FTP, file, and JAR protocols.

s3: the snapshot is stored in AWS S3, which is supported as a plugin. You can install the plugin as instructed in S3 Repository Plugin.

hdfs: the snapshot is stored in HDFS, which is supported as a plugin. You can install the plugin as instructed in Hadoop HDFS Repository Plugin.

cos: the snapshot is stored in COS, which is supported as a plugin. You can install the plugin as instructed in COS Repository for Elasticsearch.

If you want to migrate data from your Elasticsearch cluster to a Tencent Cloud ES cluster, you can directly use a COS repository, but you need to install the cos-repository plugin in your Elasticsearch cluster first (the plugin can be used only after the cluster is restarted). Then, back up the data of your Elasticsearch cluster to COS and restore the backup to the Tencent Cloud ES cluster to complete data migration.

If it is inconvenient to install the cos-repository plugin in your Elasticsearch cluster, but the repository-s3 or repository-hdfs plugin has already been installed, you can back up the data to S3 or HDFS first, upload the backup files from S3 or HDFS to COS, and then restore the data to the Tencent Cloud ES cluster.

To migrate data with COS snapshot, you need to create a COS repository first by running 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 TencentCloud API key.

access_key_secret: `SecretKey` of your TencentCloud API key.

bucket: COS bucket name, which should not contain the `-{appId}` suffix.

region: COS bucket region, which must be same as the region of your ES cluster.

base_path: backup directory.

Creating snapshot in source 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 indices. For more information on specific API parameters, please see Snapshot and Restore.

Backing up all indices

Back up all indices in the source cluster to the `my_cos_backup` repository and name the snapshot `snapshot_1` :

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

This command will be returned immediately and executed asynchronously in the background until the end. If you want to block the execution of the snapshot creating command, you can add the `wait_for_completion` parameter:

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

Note:

The time it takes to execute the command depends on 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 value of the parameter `indices` contains multiple indices, they should be separated by `,` with no spaces.

Viewing snapshot status

You can run the following command to check whether the snapshot backup has been completed. If the `state` field is `SUCCESS` in the returned result, the backup has been completed:

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

Creating repository in target cluster

Creating a repository in the target cluster is exactly the same as that in the source cluster.

Restoring data from snapshot

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

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

If snapshot_1 contains 5 indices, then all of them will be restored to the ES cluster. You can also rename the indices through an additional option which allows you to match the index name by pattern and provide a new name through the restoration process. Use this option if you want to restore old data to verify the content or perform other operations without replacing existing data. Restore a single index from a snapshot and provide an alternate name:

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

indices: only restores `index_1` and ignores other indices in the snapshot.
rename_pattern: finds the index being restored that can be matched by the specified pattern.
rename_replacement: renames the matching index to the name specified in this parameter.

Viewing index restoration status

You can call the `_recovery` API to view the restoration progress of the specified index:

```
GET index_1/_recovery
```

In addition, you can call the following API to view the status of the specified index. If `status` is `green` in the returned result, the index has been completely restored:

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

Logstash

Logstash allows you to read data from an Elasticsearch cluster and then write it to another cluster; therefore, you can use Logstash for data migration. Before using Logstash to migrate data, you need to pay attention to the following:
You need to create a CVM instance for Logstash deployment in the same VPC as the Tencent Cloud ES cluster and make sure that the CVM instance can access the source Elasticsearch cluster.

You'd better choose a high-specced CVM instance to deploy Logstash; for example, you can use a CVM instance with 16 CPU cores and 32 GB memory.

The major version number of Logstash should be the same as that of the target ES cluster; for example, if the target ES cluster version is 6.8.2, the Logstash version should also be 6.8.

You need to pay special attention to the index type. As Elasticsearch's constraints on index type vary by version, during migration across major versions, problems such as failure in writing data to the target cluster may occur due to index type issues. For more information, please see the description of the `document_type` parameter in the `logstash-output-elasticsearch` plugin.

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

```
input {
    elasticsearch {
        hosts => "1.1.1.1:9200"
        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 syncs all the indices in the source cluster to the target cluster, and it can also be set to only sync specified indices. For more information on migration with Logstash, please see Elasticsearch input plugin and Elasticsearch output plugin.

elasticsearch-dump

elasticsearch-dump is an open-source Elasticsearch data migration tool available on GitHub.

1. Install elasticsearch-dump

elasticsearch-dump is developed using Node.js and can be installed directly using the npm package manager:

```
npm install elasticdump -g
```

2. Main parameters

```
--input: the source address in the format of {protocol}://{host}:
{port}/{index}, which can be an Elasticsearch cluster URL, file, or stdin and
allows you to specify an index
--input-index: the index in the source cluster
--output: the target address in the format of {protocol}://{host}:
{port}/{index}, which can be an Elasticsearch cluster URL, file, or stdout and
allows you to specify an index
--output-index: the index in the target cluster
--type: the migration type, which is data by default, indicating that only data
will be migrated. Value range: settings, analyzer, data, mapping, alias
```

3. If the cluster requires security authentication, you can use reindex to authenticate it as instructed below:
Add `user:password@` after the corresponding `http`. Please refer to the sample `elasticsearch-dump --input=http://192.168.1.2:9200/my_index --output=http://user:password@192.168.1.2:9200/my_index --type=data`.

4. Migrate a single index
The following operation migrates the `companydatabase` index in the `172.16.0.39` cluster to the `172.16.0.20` cluster by running the `elasticdump` command.

Note:

The first command migrates the settings of the index. If you directly migrate the mapping or data, the configuration information of the index in the source cluster will be lost, such as the number of shards and number of replicas. Of course, you can also directly create an index in the target cluster first before syncing the mapping and data.

```
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
```

5. Migrate all indices
The following operation migrates all indices in the `172.16.0.39` cluster to the `172.16.0.20` cluster by running the `elasticdump` command.

Note:

This operation cannot migrate the configurations of indices, such as the number of shards and number of replicas. You must migrate the configuration of each index separately, or directly create an index in the target cluster first before migrating the data.

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

Summary

1. When elasticsearch-dump or Logstash is used to migrate data from one cluster to another, the machine used to perform the migration task is required to have access to both clusters at the same time, because migration cannot be performed if there is no network connection. This limitation does not apply if snapshot is used, as it is completely offline. Therefore, elasticsearch-dump and Logstash are more suitable for migrating data between clusters on the same network. If you want to migrate your data between cloud vendors, such as from an Alibaba Cloud Elasticsearch cluster to a Tencent Cloud ES cluster, you can use snapshot or establish a connection between the vendors to achieve cluster connectivity, which, however, is costly.
2. elasticsearch-dump is similar to MySQL's mysqldump which is used for data backup, and both of them perform logical backup that involves exporting data entries one by one and then importing them. Therefore, this tool is suitable for migrating a small amount of data.
3. Snapshot is suitable for migrating a large amount of data.

Data Ingestion into ES

Last updated：2024-12-02 10:02:26

ES allows access to your cluster through private VIP within your VPC. You can write code to access your cluster through the Elasticsearch REST client and import your data into the cluster. You can also ingest your data through Elasticsearch's official components such as Logstash and Beats.

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:

Please note that the Logstash version should be the same as the Elasticsearch version.

2. Customize the `\\*.conf` configuration file based on the data source type. For more information, please see Data Source Configuration File Description.

3. Run Logstash.

```
 nohup ./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
```

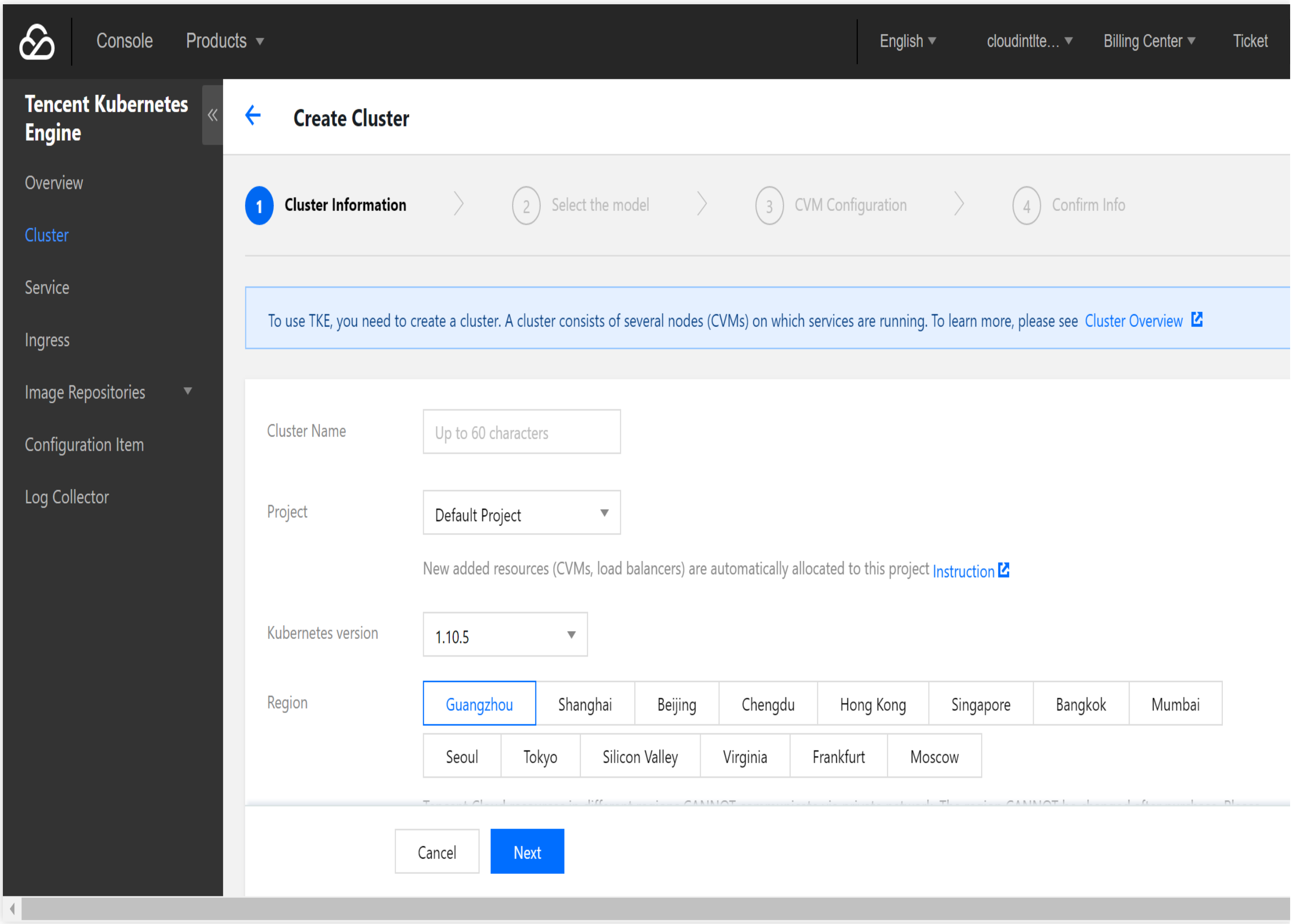
2. Customize the `\\*.conf` configuration file based on the data source type and place it in the `/usr/share/logstash/pipeline/` directory which can be customized.
3. Run Logstash.

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

Using 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 the TKE Console and select **Cluster** > **Create** on the left sidebar to create a cluster.



To use TKE, you need to create a cluster. A cluster consists of several nodes (CVMs) on which services are running. To learn more, please see Cluster Overview

Cluster Name: Up to 60 characters

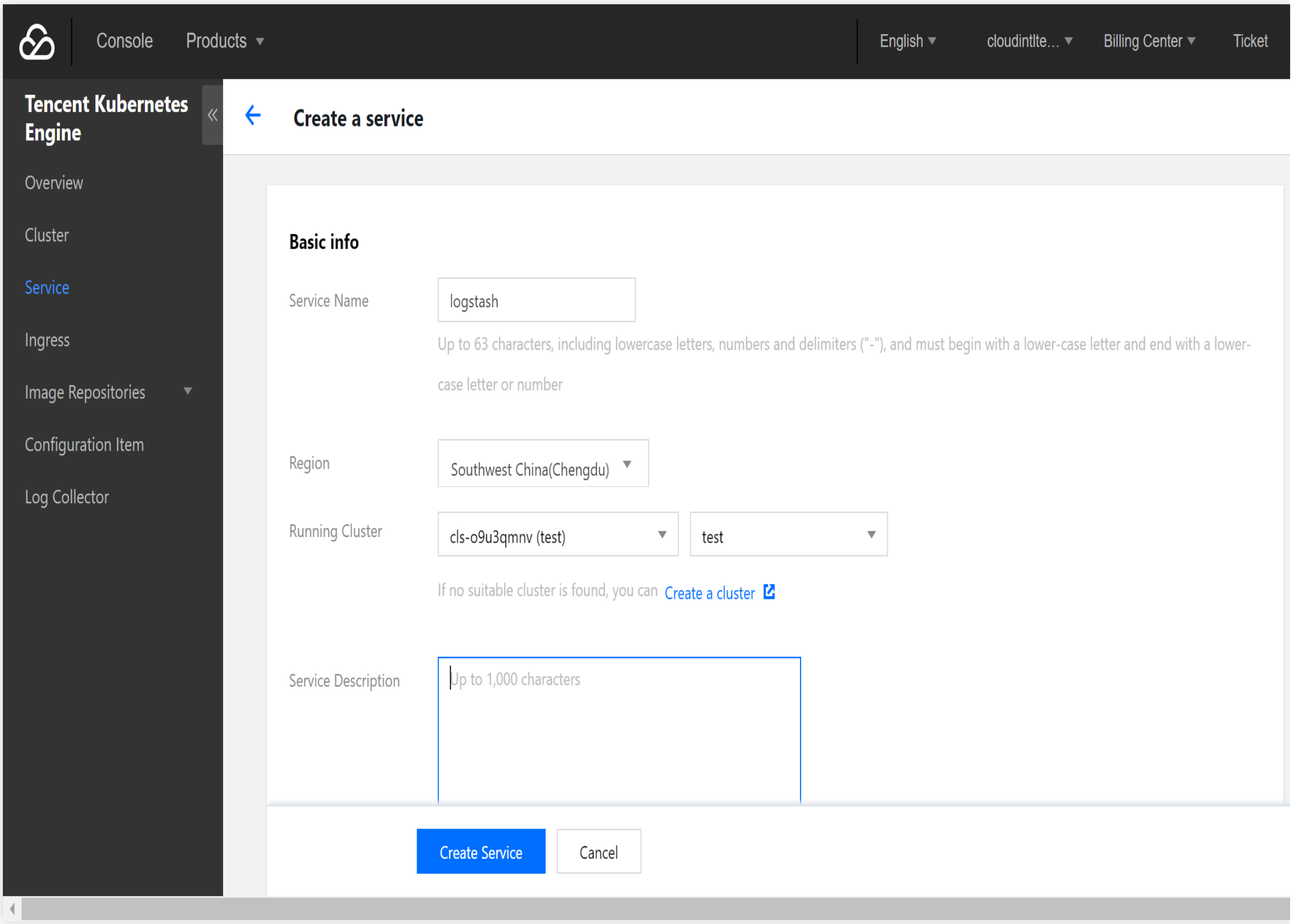
Project: Default Project

New added resources (CVMs, load balancers) are automatically allocated to this project Instruction

Kubernetes version: 1.10.5

Region: Guangzhou | Shanghai | Beijing | Chengdu | Hong Kong | Singapore | Bangkok | Mumbai | Seoul | Tokyo | Silicon Valley | Virginia | Frankfurt | Moscow

2. Select **Service** on the left sidebar and click **Create** to create a service.



Service Name: logstash

Up to 63 characters, including lowercase letters, numbers and delimiters ("-"), and must begin with a lower-case letter and end with a lower-case letter or number

Region: Southwest China(Chengdu)

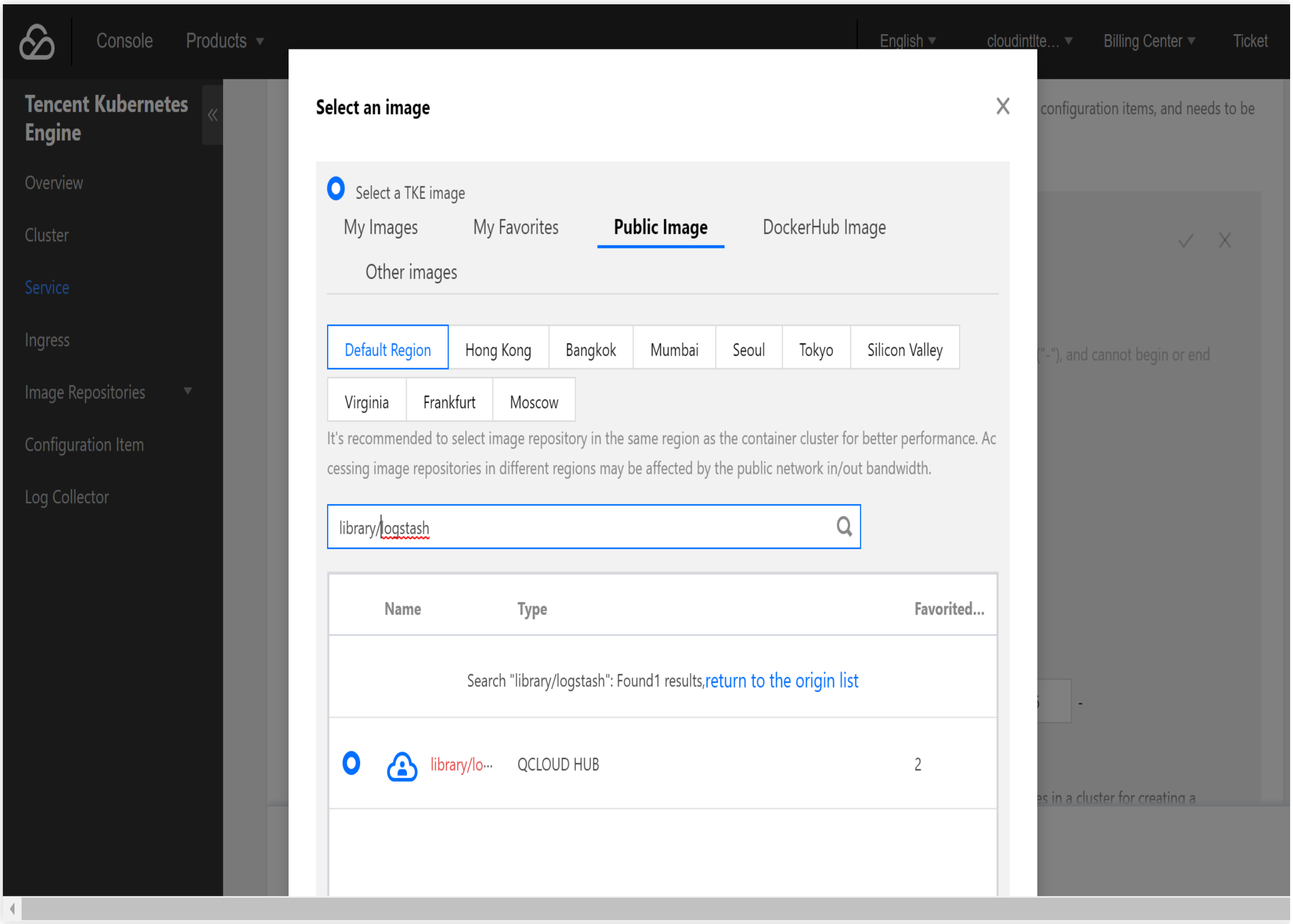
Running Cluster: cls-o9u3qmnv (test) | test

If no suitable cluster is found, you can Create a cluster

Service Description: Up to 1,000 characters

3. Select the official image of Logstash.

In this example, the Logstash image provided by TencentHub image registry is used. You can also create a Logstash image by yourself.



It's recommended to select image repository in the same region as the container cluster for better performance. Accessing image repositories in different regions may be affected by the public network in/out bandwidth.

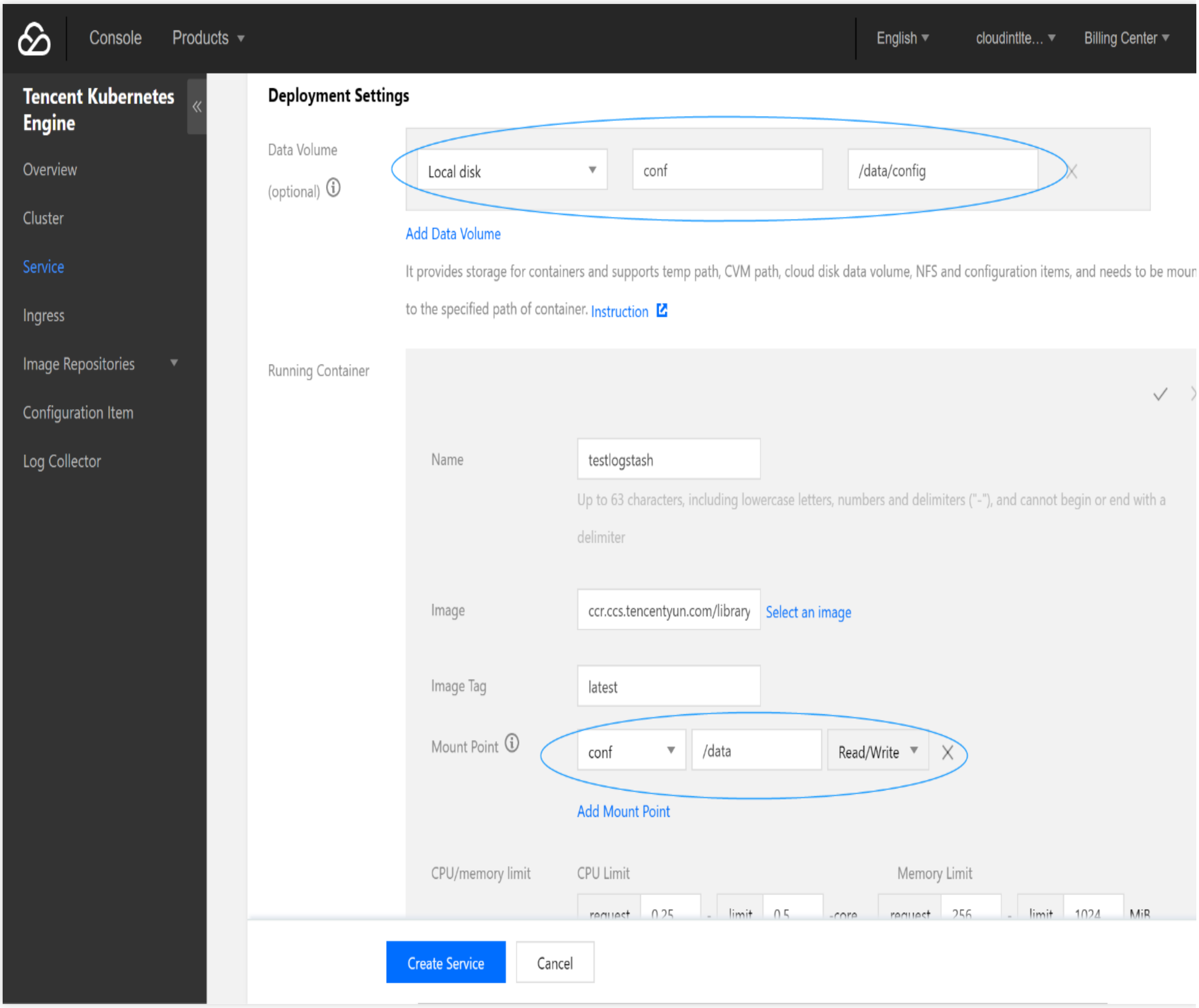
library/logstash

Name	Type	Favorited...
library/lo...	QCLOUD HUB	2

Search "library/logstash": Found1 results,return to the origin list

4. Create a data volume.

Create a data volume to store the Logstash configuration file. In this example, a configuration file named `logstash.conf` is added to the `/data/config` directory on the CVM instance and mounted to the `/data` directory of Docker, so that the `logstash.conf` file can be read when the container starts.



Deployment Settings

Data Volume (optional): Local disk | conf | /data/config

It provides storage for containers and supports temp path, CVM path, cloud disk data volume, NFS and configuration items, and needs to be moun to the specified path of container.

Running Container

Name: testlogstash

Up to 63 characters, including lowercase letters, numbers and delimiters ("-"), and cannot begin or end with a delimiter

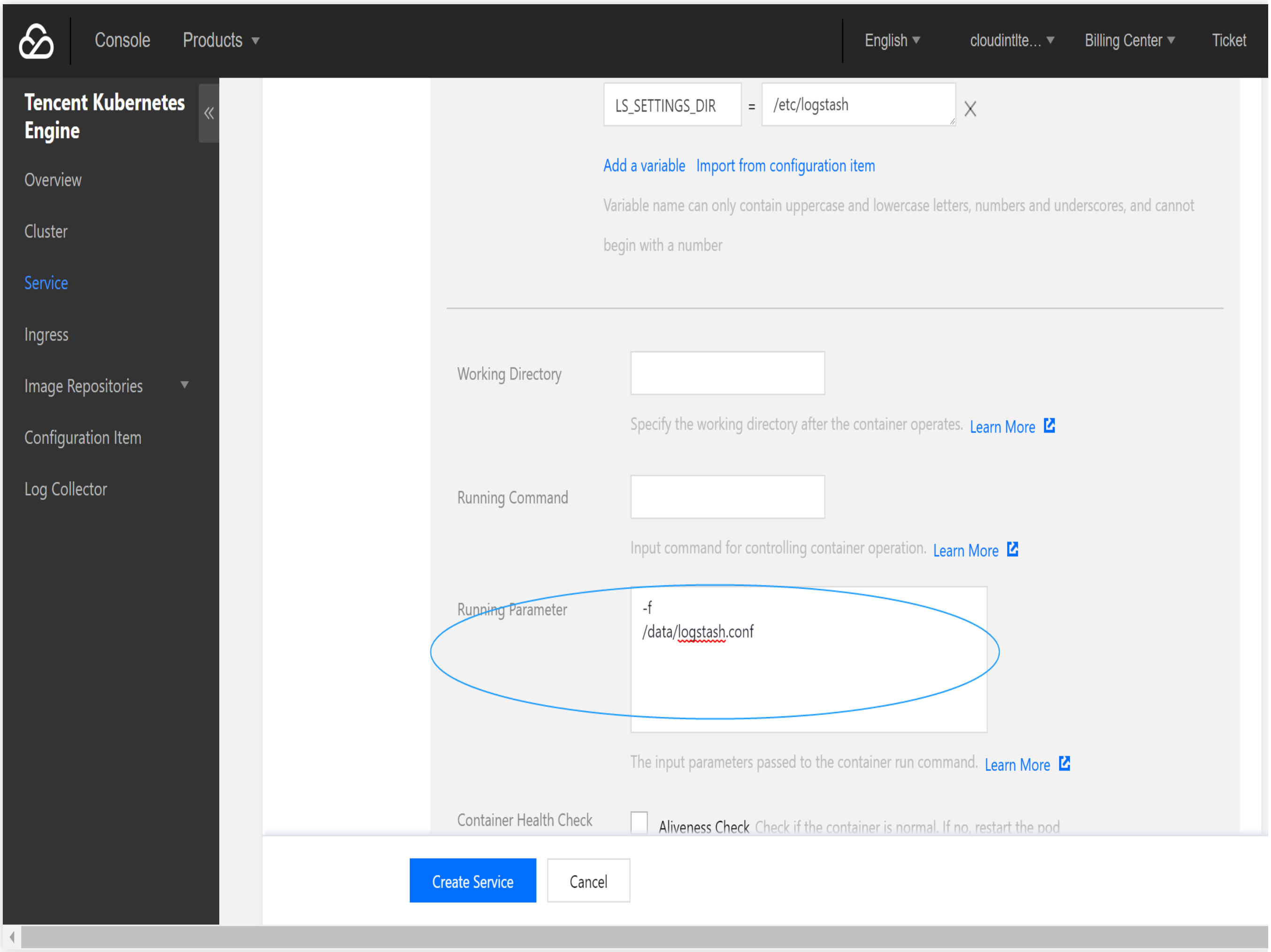
Image: ccr.ccs.tencentyun.com/library

Image Tag: latest

Mount Point: conf | /data | Read/Write

CPU/memory limit: CPU Limit request 0.25 - limit 0.5 -core; Memory Limit request 256 - limit 1024 MiB

5. Configure the execution parameters.



LS_SETTINGS_DIR = /etc/logstash

Add a variable Import from configuration item

Variable name can only contain uppercase and lowercase letters, numbers and underscores, and cannot begin with a number

Working Directory

Specify the working directory after the container operates. Learn More

Running Command

Input command for controlling container operation. Learn More

Running Parameter

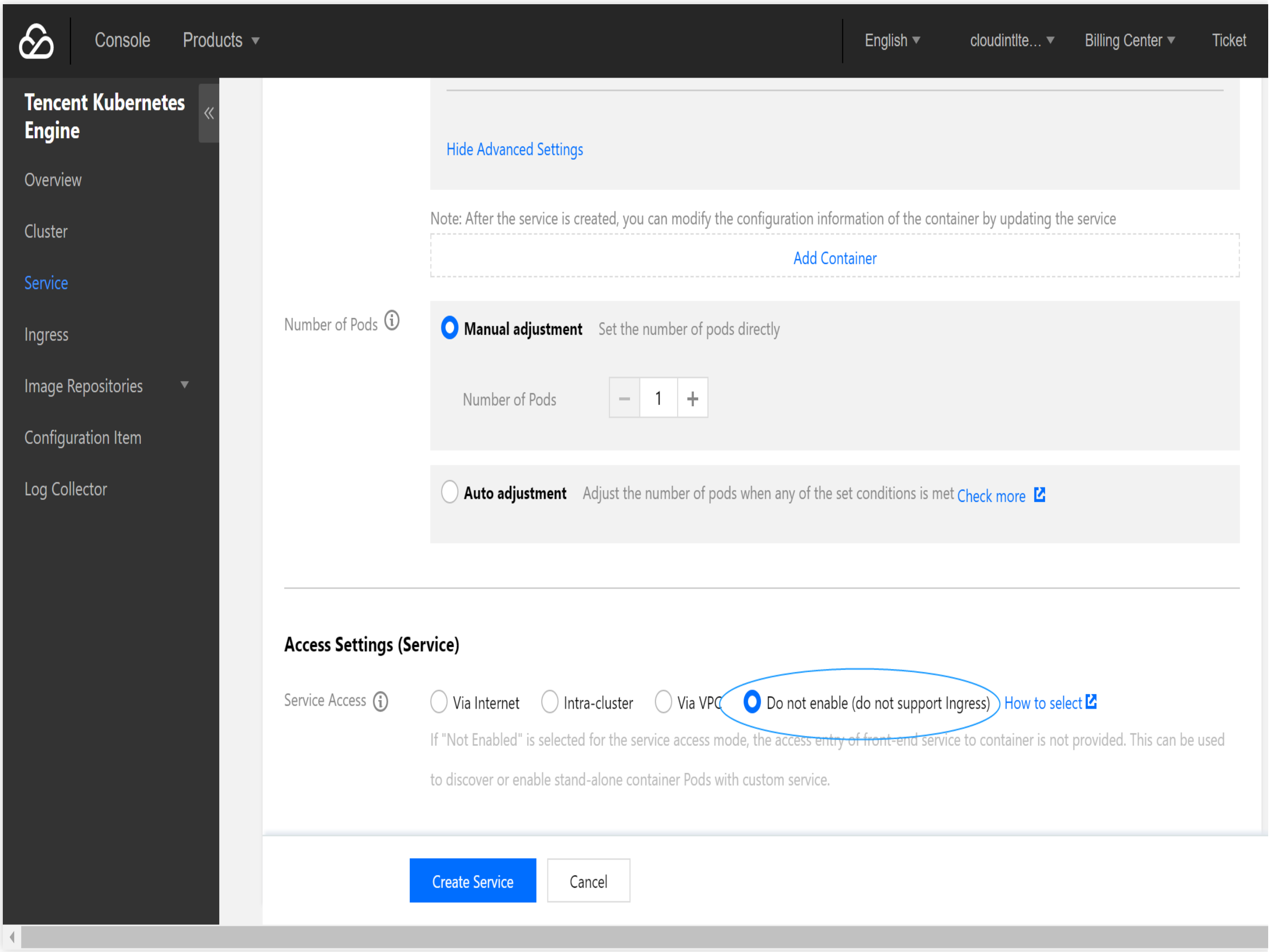
```
-f
/data/logstash.conf
```

The input parameters passed to the container run command. Learn More

Container Health Check □ Aliveness Check Check if the container is normal. If no, restart the pod

Create Service Cancel

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



Note: After the service is created, you can modify the configuration information of the container by updating the service

Number of Pods
● Manual adjustment Set the number of pods directly
Number of Pods 1
○ Auto adjustment Adjust the number of pods when any of the set conditions is met Check more

Access Settings (Service)

Service Access ○ Via Internet ○ Intra-cluster ○ Via VPC ● Do not enable (do not support Ingress) How to select

If "Not Enabled" is selected for the service access mode, the access entry of front-end service to container is not provided. This can be used to discover or enable stand-alone container Pods with custom service.

Configuration file description

File data sources

```
input {
    file {
        path => "/var/log/nginx/access.log" # File path
        }
}
filter {
}
output {
  elasticsearch {
    hosts => ["http://172.16.0.89:9200"] # Private VIP address and port of the
ES cluster
    index => "nginx_access-%{+YYYY.MM.dd}" # Custom index name suffixed with
date. One index is generated per 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" # Start consumption from the latest
offset
        consumer_threads => 5
        decorate_events => true # This attribute will bring the current topic,
offset, group, partition, and other information into the message
        topics => ["test1","test2"] # Array type. Multiple topics can be
configured
        type => "test" # Data source identification field
      }
}

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

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

Database data sources connected with JDBC

```
input {
    jdbc {
      # MySQL database address
      jdbc_connection_string => "jdbc:mysql://172.16.32.14:3306/test"
      # Username and password
      jdbc_user => "root"
      jdbc_password => "Elastic123"
      # Driver jar package. You need to download the jar when installing and
deploying Logstash on your own as it is not provided by Logstash by default
      jdbc_driver_library => "/usr/local/services/logstash-5.6.4/lib/mysql-
connector-java-5.1.40.jar"
      # Driver class name
      jdbc_driver_class => "com.mysql.jdbc.Driver"
      jdbc_paging_enabled => "true"
      jdbc_page_size => "50000"
      # Path and name of the SQL file to be executed
      #statement_filepath => "test.sql"
      # SQL statement to be executed
      statement => "select * from test_es"
      # Set the monitoring interval. The meanings of each field (from left to right) are minutes, hours, days, months, and years. If all of them are `*`, it indicates to update once 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 JDBC input plugin.

Using Beats to Access ES Cluster

Beats contains a variety of single-purpose collectors. These collectors are relatively lightweight and can be deployed and run on servers to collect data such as logs and monitoring information. Beats occupies less system resources than Logstash does.

Beats includes FileBeat for collecting file-type data, MetricBeat for collecting monitoring metric data, PacketBeat for collecting network packet data, etc. You can also develop your own Beats components based on the official `libbeat` library as needed.

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.tar.gz
```

2. Configure `filebeat.yml` .

3. Run Filebeat.

```
 nohup ./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 `\\*.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 Filebeat.

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

Using TKE

The deployment method of Filebeat through TKE is similar to that of Logstash, and you can use the Filebeat image provided by Tencent Cloud.

Configuration 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"]
```

Syncing MySQL Data to ES in Real Time

Last updated：2019-11-08 17:54:15

This document describes how to sync data in MySQL to ES in real time by syncing the MySQL binlog, which can implement low-latency data search and analysis. This method has been proven feasible and is illustrated for your reference.

MySQL binlog

MySQL's binlog is mainly used for master-slave replication and data recovery in databases. The binlog records the CRUD operations performed on the data. During the master-slave replication process, the master database syncs its binlog to the slave database which replays the events stored in the binlog to achieve master-slave sync.

MySQL binlog has three logging modes, namely:

ROW: It records the changes to each data row, which, however, generates a high number of logs.

STATEMENT: It records each executed SQL statement that modifies the data, which reduces the number of logs; however, master-slave inconsistency often occurs when SQL statements use functions or triggers.

MIXED: It combines the advantages of ROW and STATEMENT by choosing either ROW or STATEMENT to generate logs according to the specific SQL statement that performs data operations.

To sync data to an ES cluster with MySQL binlog, you can only use ROW mode, because only this mode knows what is modified in the MySQL data. The following shows the binlog contents in ROW and STATEMENT modes with an UPDATE operation as an example.

The binlog contents in ROW mode are as follows:

```
SET TIMESTAMP=1527917394/*!*/;
BEGIN
/*!*/;
# at 3751
#180602 13:29:54 server id 1  end_log_pos 3819 CRC32 0x8dabdf01
Table_map: `webservice`.`building` mapped to number 74
# at 3819
#180602 13:29:54 server id 1  end_log_pos 3949 CRC32 0x59a8ed85
Update_rows: table id 74 flags: STMT_END_F
BINLOG '
UisSWxMBAAAARAAAAOsOAAAAAEoAAAAAAAEACndlYnNlcnZpY2UACGJ1aWxkaW5nAAYIDwEPEREG
wACAAQAAAAHfq40=
UisSWx8BAAAAggAAAG0PAAAAAEoAAAAAAAEAAgAG///A1gcAAAAAAAALYnVpbGRpbmctMTAADwB3
UkRNbjNLYlV5d1k3ajVbD64WWw+uFsDWBwAAAAAAAAtidWlsZGluZy0xMAEPAHdSRE1uM0tiVXl3
WTdqNVsPrhZbD64Whe2oWQ==
'/*!*/;
### UPDATE `webservice`.`building`
### WHERE
###   @1=2006 /* LONGINT meta=0 nullable=0 is_null=0 */
###   @2='building-10' /* VARSTRING(192) meta=192 nullable=0 is_null=0 */
###   @3=0 /* TINYINT meta=0 nullable=0 is_null=0 */
###   @4='wRDMn3KbUywY7j5' /* VARSTRING(384) meta=384 nullable=0 is_null=0 */
###   @5=1527754262 /* TIMESTAMP(0) meta=0 nullable=0 is_null=0 */
###   @6=1527754262 /* TIMESTAMP(0) meta=0 nullable=0 is_null=0 */
### SET
###   @1=2006 /* LONGINT meta=0 nullable=0 is_null=0 */
###   @2='building-10' /* VARSTRING(192) meta=192 nullable=0 is_null=0 */
###   @3=1 /* TINYINT meta=0 nullable=0 is_null=0 */
###   @4='wRDMn3KbUywY7j5' /* VARSTRING(384) meta=384 nullable=0 is_null=0 */
###   @5=1527754262 /* TIMESTAMP(0) meta=0 nullable=0 is_null=0 */
###   @6=1527754262 /* TIMESTAMP(0) meta=0 nullable=0 is_null=0 */
# at 3949
#180602 13:29:54 server id 1  end_log_pos 3980 CRC32 0x58226b8f         Xid =
182
COMMIT/*!*/;
```

The binlog contents in STATEMENT mode are as follows:

```
SET TIMESTAMP=1527919329/*!*/;
update building set Status=1 where Id=2000
/*!*/;
# at 688
#180602 14:02:09 server id 1  end_log_pos 719 CRC32 0x4c550a7d  Xid = 200
COMMIT/*!*/;
```

As can be seen from the log contents of the UPDATE operation in ROW and STATEMENT modes, the ROW mode completely records the values of all fields in the row to be modified before and after the update, while the STATEMENT mode only records the SQL statement of the UPDATE operation. Therefore, if you need to sync MySQL data to ES in real time, you can only choose the binlog in ROW mode, obtain and parse the data contents in the binlog, and execute the ES document API to sync the data to the ES cluster.

mysqldump tool

`mysqldump` is a tool for exporting full data from a MySQL database. It can be used in the following way:

```
mysqldump -uelastic -p'Elastic_123' --host=172.16.32.5 -F webservice > dump.sql
```

The command above indicates to export all data of `database:webservice` from the remote database `172.16.32.5:3306` and write to the `dump.sql` file. The `-F` parameter indicates to generate a new binlog file after exporting the data to log all subsequent data operations. The contents of the `dump.sql` file are as follows:

```
-- MySQL dump 10.13  Distrib 5.6.40, for Linux (x86_64)
--
-- Host: 172.16.32.5    Database: webservice
-- ------------------------------------------------------
-- Server version       5.5.5-10.1.9-MariaDBV1.0R012D002-20171127-1822

/*!40101 SET @OLD_CHARACTER_SET_CLIENT=@@CHARACTER_SET_CLIENT */;
/*!40101 SET @OLD_CHARACTER_SET_RESULTS=@@CHARACTER_SET_RESULTS */;
/*!40101 SET @OLD_COLLATION_CONNECTION=@@COLLATION_CONNECTION */;
/*!40101 SET NAMES utf8 */;
/*!40103 SET @OLD_TIME_ZONE=@@TIME_ZONE */;
/*!40103 SET TIME_ZONE='+00:00' */;
/*!40014 SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0 */;
/*!40014 SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS, FOREIGN_KEY_CHECKS=0
*/;
/*!40101 SET @OLD_SQL_MODE=@@SQL_MODE, SQL_MODE='NO_AUTO_VALUE_ON_ZERO' */;
/*!40111 SET @OLD_SQL_NOTES=@@SQL_NOTES, SQL_NOTES=0 */;

--
-- Table structure for table `building`
--

DROP TABLE IF EXISTS `building`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `building` (
  `Id` bigint(20) unsigned NOT NULL AUTO_INCREMENT COMMENT 'ID',
  `BuildingId` varchar(64) NOT NULL COMMENT 'Virtual building ID',
  `Status` tinyint(4) NOT NULL DEFAULT '0' COMMENT 'Virtual building status. 0:
processing; 1: normal; -1: stopped; -2: terminating; -3: terminated',
  `BuildingName` varchar(128) NOT NULL DEFAULT '' COMMENT 'Virtual building
name',
  `CreateTime` timestamp NOT NULL DEFAULT '2017-12-03 16:00:00' COMMENT
'Creation time',
  `UpdateTime` timestamp NOT NULL DEFAULT '2017-12-03 16:00:00' COMMENT 'Update
time',
  PRIMARY KEY (`Id`),
  UNIQUE KEY `BuildingId` (`BuildingId`)
) ENGINE=InnoDB AUTO_INCREMENT=2010 DEFAULT CHARSET=utf8 COMMENT='Virtual
building table';
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Dumping data for table `building`
--
```

```
LOCK TABLES `building` WRITE;
/*!40000 ALTER TABLE `building` DISABLE KEYS */;
INSERT INTO `building` VALUES (2000,'building-2',0,'6YFcmntKrNBIeTA','2018-05-30 13:28:31','2018-05-30 13:28:31'),(2001,'building-4',0,'4rY8PcVUZB1vtrL','2018-05-30 13:28:34','2018-05-30 13:28:34'),(2002,'building-5',0,'uyjHVUYrg9KeGqi','2018-05-30 13:28:37','2018-05-30 13:28:37'),(2003,'building-7',0,'DNhyEBO4XEkXpgW','2018-05-30 13:28:40','2018-05-30 13:28:40'),(2004,'building-1',0,'TmtYX6ZC0RNB4Re','2018-05-30 13:28:43','2018-05-30 13:28:43'),(2005,'building-6',0,'t8YQcjeXefWpcyU','2018-05-30 13:28:49','2018-05-30 13:28:49'),(2006,'building-10',0,'WozgBc2IchNyKyE','2018-05-30 13:28:55','2018-05-30 13:28:55'),(2007,'building-3',0,'yJk27cmLOVQLHf1','2018-05-30 13:28:58','2018-05-30 13:28:58'),(2008,'building-9',0,'RSbjotAh8tymfxs','2018-05-30 13:29:04','2018-05-30 13:29:04'),(2009,'building-8',0,'IBOMlhaXV6k226m','2018-05-30 13:29:31','2018-05-30 13:29:31');
/*!40000 ALTER TABLE `building` ENABLE KEYS */;
UNLOCK TABLES;

/*!40103 SET TIME_ZONE=@OLD_TIME_ZONE */;

/*!40101 SET SQL_MODE=@OLD_SQL_MODE */;
/*!40014 SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS */;
/*!40014 SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS */;
/*!40101 SET CHARACTER_SET_CLIENT=@OLD_CHARACTER_SET_CLIENT */;
/*!40101 SET CHARACTER_SET_RESULTS=@OLD_CHARACTER_SET_RESULTS */;
/*!40101 SET COLLATION_CONNECTION=@OLD_COLLATION_CONNECTION */;
/*!40111 SET SQL_NOTES=@OLD_SQL_NOTES */;

-- Dump completed on 2018-06-02 14:23:51
```

As can be seen from the above, the SQL file exported by mysqldump contains SQL statements for creating table, dropping table, and inserting data, but does not contain the statement for creating database.

Using the open-source go-mysql-elasticsearch tool to sync data to ES

`go-mysql-elasticsearch` is an open-source tool for syncing MySQL data to an ES cluster. For more information, see its GitHub page.

The following describes how `go-mysql-elasticsearch` works. When you launch it for the first time, use the `mysqldump` tool to perform a full sync of the source MySQL database first and write the data to ES through the Elasticsearch client; then, implement a MySQL client as the slave, connect it to the source MySQL database which, as the master, will sync all data updates to the slave through binlog events. By parsing such events, the updated contents of the data can be obtained and written to ES.

In addition, this tool also provides the feature to count operations. When a data addition, deletion, or update is performed, the count of the corresponding operation will be increased by 1. When the program starts, an HTTP service will be launched, and the number of additions, deletions, and updates can be viewed by calling the HTTP API.

Use limits

1. MySQL binlog must use ROW mode (which is enabled in TencentDB for MySQL by default).
2. The MySQL data table to be synced must contain a primary key; otherwise, it will be ignored directly. If the data table has no primary key, the UPDATE and DELETE operations cannot find the corresponding documents in ES, causing a sync failure.
3. Table structure cannot be modified when the program is running.
4. You need to grant the account used to connect to MySQL the RELOAD and REPLICATION permission.

```
GRANT REPLICATION SLAVE ON *.* TO 'elastic'@'172.16.32.44';
GRANT RELOAD ON *.* TO 'elastic'@'172.16.32.44';
```

How to use

1. Install Go 1.10+. You can simply install the latest version of Go and then set the `GOPATH` environment variable.
2. Run the `go get github.com/siddontang/go-mysql-elasticsearch` command.
3. Run the `cd $GOPATH/src/github.com/siddontang/go-mysql-elasticsearch` command.
4. Run the `make` command to start compiling. After the compilation is successful, an executable file named `go-mysql-elasticsearch` will be generated in the `go-mysql-elasticsearch/bin` directory.
5. Run the `vi etc/river.toml` command to modify the configuration file so as to sync the `webservice.building` table in the `172.16.0.101:3306` database to the building index of the `172.16.32.64:9200` ES cluster. (For detailed description of the configuration file, see the project documentation.)

```
 # MySQL address, user and password
 # user must have replication privilege in MySQL.
 my_addr = "172.16.0.101:3306"
 my_user = "bellen"
 my_pass = "Elastic_123"
 my_charset = "utf8"

 # Set true when elasticsearch use https
 #es_https = false
 # Elasticsearch address
 es_addr = "172.16.32.64:9200"
 # Elasticsearch user and password, maybe set by shield, nginx, or x-pack
```

```
es_user = ""
es_pass = ""

# Path to store data, like master.info, if not set or empty,
# we must use this to support breakpoint resume syncing.
# TODO: support other storage, like etcd.
data_dir = "./var"

# Inner Http status address
stat_addr = "127.0.0.1:12800"

# pseudo server id like a slave
server_id = 1001

# mysql or mariadb
flavor = "mariadb"

# mysqldump execution path
# if not set or empty, ignore mysqldump.
mysqldump = "mysqldump"

# if we have no privilege to use mysqldump with --master-data,
# we must skip it.
#skip_master_data = false

# minimal items to be inserted in one bulk
bulk_size = 128

# force flush the pending requests if we don't have enough items >= bulk_size
flush_bulk_time = "200ms"

# Ignore table without primary key
skip_no_pk_table = false

# MySQL data source
[[source]]
schema = "webservice"
tables = ["building"]
[[rule]]
schema = "webservice"
table = "building"
index = "building"
type = "buildingtype"
```

6. Create a building index in the ES cluster, because the tool does not use the "auto create index" feature of ES, and an error will be displayed if the index does not exist.

7. Run the `./bin/go-mysql-elasticsearch -config=./etc/river.toml` command.
8. Output the result in the console.

```
 2018/06/02 16:13:21 INFO  create BinlogSyncer with config {1001 mariadb
172.16.0.101 3306 bellen   utf8 false false <nil> false false 0 0s 0s 0}
 2018/06/02 16:13:21 INFO  run status http server 127.0.0.1:12800
 2018/06/02 16:13:21 INFO  skip dump, use last binlog replication pos (mysql-
bin.000001, 120) or GTID %!s(<nil>)
 2018/06/02 16:13:21 INFO  begin to sync binlog from position (mysql-
bin.000001, 120)
 2018/06/02 16:13:21 INFO  register slave for master server 172.16.0.101:3306
 2018/06/02 16:13:21 INFO  start sync binlog at binlog file (mysql-bin.000001,
120)
 2018/06/02 16:13:21 INFO  rotate to (mysql-bin.000001, 120)
 2018/06/02 16:13:21 INFO  rotate binlog to (mysql-bin.000001, 120)
 2018/06/02 16:13:21 INFO  save position (mysql-bin.000001, 120)
```

9. Test: Data insertion, update, and deletion performed in MySQL can be reflected in ES.

User experience

`go-mysql-elasticsearch` provides the most basic capability to sync data from MySQL to ES in real time. If your business requires more complex features such as modifying the MySQL table structure during operation, you can customize it for secondary development.

The tool is not good at handling exceptions. If parsing binlog events fails, it will throw an exception directly.

As described by the tool developer, the tool has not been applied to a production environment, so you are recommended to read the source code carefully to understand its pros and cons.

Syncing data to an ES cluster using mypipe

mypipe is a MySQL binlog sync tool initially designed to send binlog events to Kafka. It can also sync data to any storage media based on your actual business needs. For more information, see its GitHub page.

Use limits

1. MySQL binlog must use ROW mode.
2. You need to grant the account used to connect to MySQL the REPLICATION permission.

```
 GRANT REPLICATION SLAVE, REPLICATION CLIENT ON *.* TO 'elastic'@'%' IDENTIFIED
BY 'Elastic_123'
```

3. mypipe just parses the contents of the binlog, encodes them into Avro format, and pushes them to a Kafka broker instead of directly pushing data to Kafka. If you need to sync data to an ES cluster, you can consume the data from

Kafka first and then write it to ES.

4. To consume messages in Kafka (operations log of MySQL), you need to perform Avro parsing on the message contents and obtain the corresponding data operations for further processing. mypipe encapsulates a `KafkaGenericMutationAvroConsumer` class that can be directly inherited for use. You can also implement the parsing on your own.

5. mypipe supports syncing only binlog but not existing data, i.e., after the mypipe program is started, existing data in MySQL cannot be synced.

How to use

1. Run the `git clone https://github.com/mardambey/mypipe.git` command.
2. Run the `./sbt package` command.
3. Configure `mypipe-runner/src/main/resources/application.conf` .

```
mypipe {

  # Avro schema repository client class name
  schema-repo-client = "mypipe.avro.schema.SchemaRepo"

  # consumers represent sources for mysql binary logs
  consumers {

    localhost {
      # database "host:port:user:pass" array
      source = "172.16.0.101:3306:elastic:Elastic_123"
    }
  }

  # data producers export data out (stdout, other stores, external services, etc.)
  producers {

    kafka-generic {
      class = "mypipe.kafka.producer.KafkaMutationGenericAvroProducer"
    }
  }

  # pipes join consumers and producers
  pipes {

    kafka-generic {
      enabled = true
      consumers = ["localhost"]
      producer {
        kafka-generic {
           metadata-brokers = "172.16.16.22:9092"
         }
       }
       binlog-position-repo {
         # saved to a file, this is the default if unspecified
         class =
"mypipe.api.repo.ConfigurableFileBasedBinaryLogPositionRepository"
         config {
           file-prefix = "stdout-00"     # required if binlog-position-repo is
specifiec
           data-dir = "/tmp/mypipe/data" # defaults to mypipe.data-dir if not
present
         }
       }
     }
   }
 }
```

4. Configure `mypipe-api/src/main/resources/reference.conf` by modifying the `include-event-condition` option to specify the database and table to be synced.

```
 include-event-condition = """ db == "webservice" && table =="building" """
```

5. Create `topic: webservice_building_generic` on the Kafka broker. By default, mypipe uses `${db}_${table}_generic` as the topic name to send data to the topic.

6. Run the `./sbt "project runner" "runMain mypipe.runner.PipeRunner"` command.

7. Test: Insert data into the MySQL building table and write a simple consumer to consume the messages pushed to Kafka by mypipe.

8. The consumed data that has not been parsed yet is as follows:

```
 ConsumerRecord(topic=u'webservice_building_generic', partition=0, offset=2,
timestamp=None, timestamp_type=None, key=None,
value='\\x00\\x01\\x00\\x00\\x14webservice\\x10building\\xcc\\x01\\x02\\x91,\\x
ae\\xa3fc\\x11\\xe8\\xa1\\xaaRT\\x00Z\\xf9\\xab\\x00\\x00\\x04\\x18BuildingName
\\x06xxx\\x14BuildingId\\nId-10\\x00\\x02\\x04Id\\xd4%\\x00',
checksum=128384379, serialized_key_size=-1, serialized_value_size=88)
```

User experience

mypipe is more sophisticated than go-mysql-elasticsearch. It supports performing ALTER TABLE operations during running and configuring different policies to handle exceptions of binlog parsing.

mypipe cannot sync existing data. If you need this feature, you can use other means to sync full data and then use mypipe for incremental sync.

mypipe only syncs binlog. If you need to sync data to ES, separate development will be required.

Use Case Construction

Building a Log Analysis System

Last updated：2019-11-12 16:08:31

An instance provided by ES consists of an ES cluster and a Kibana Console. The former can be accessed via the VIP address and port of your VPC, and the latter provides a public IP address for you to access from a browser. Currently, you can only ingest your data into the ES cluster on your own. The following describes how to import your logs into ES and access Kibana from a browser to perform query and analysis by taking 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

In this example, NGINX log is used as the input source, and the output is configured as the private VIP address and port of the ES cluster. If you use a Platinum Edition cluster, you need to add username and password to the output. Enter the `filebeat-6.4.3-linux-x86_64` directory and modify the `filebeat.yml` configuration file, as shown below:

```
filebeat.inputs:
- type: log
      enabled: true
      paths:
        - /var/log/nginx/access.log
output.elasticsearch:
      hosts: ["10.0.130.91:9200"]
      protocol: "http"
      username: "elastic"
      password: "test"
```

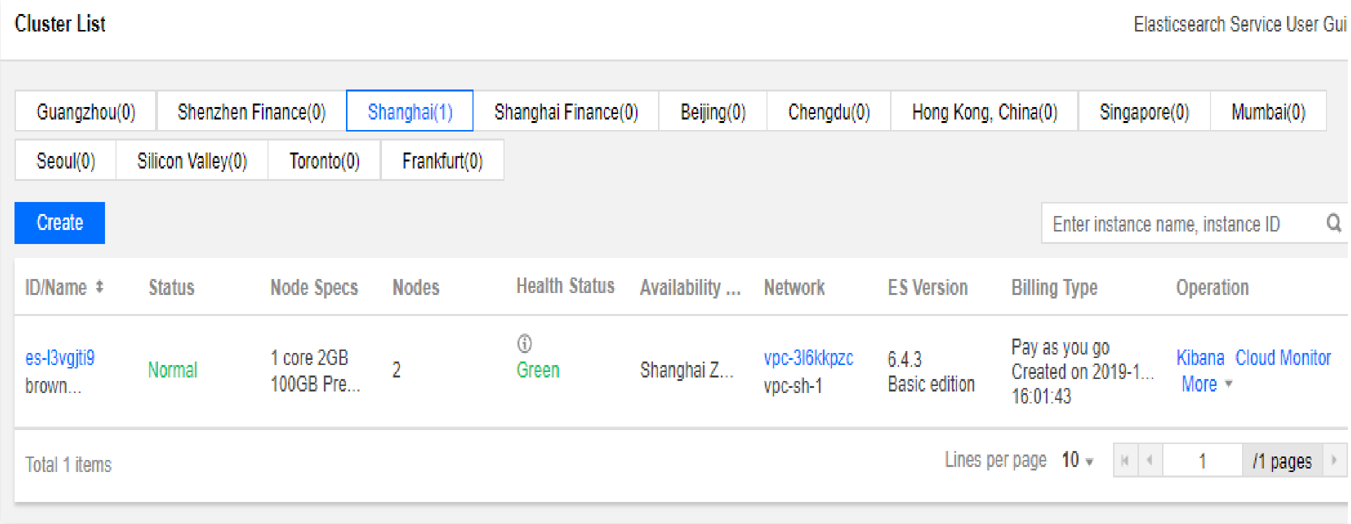
3. Run Filebeat

In the `filebeat-6.4.3-linux-x86_64` directory, run:

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

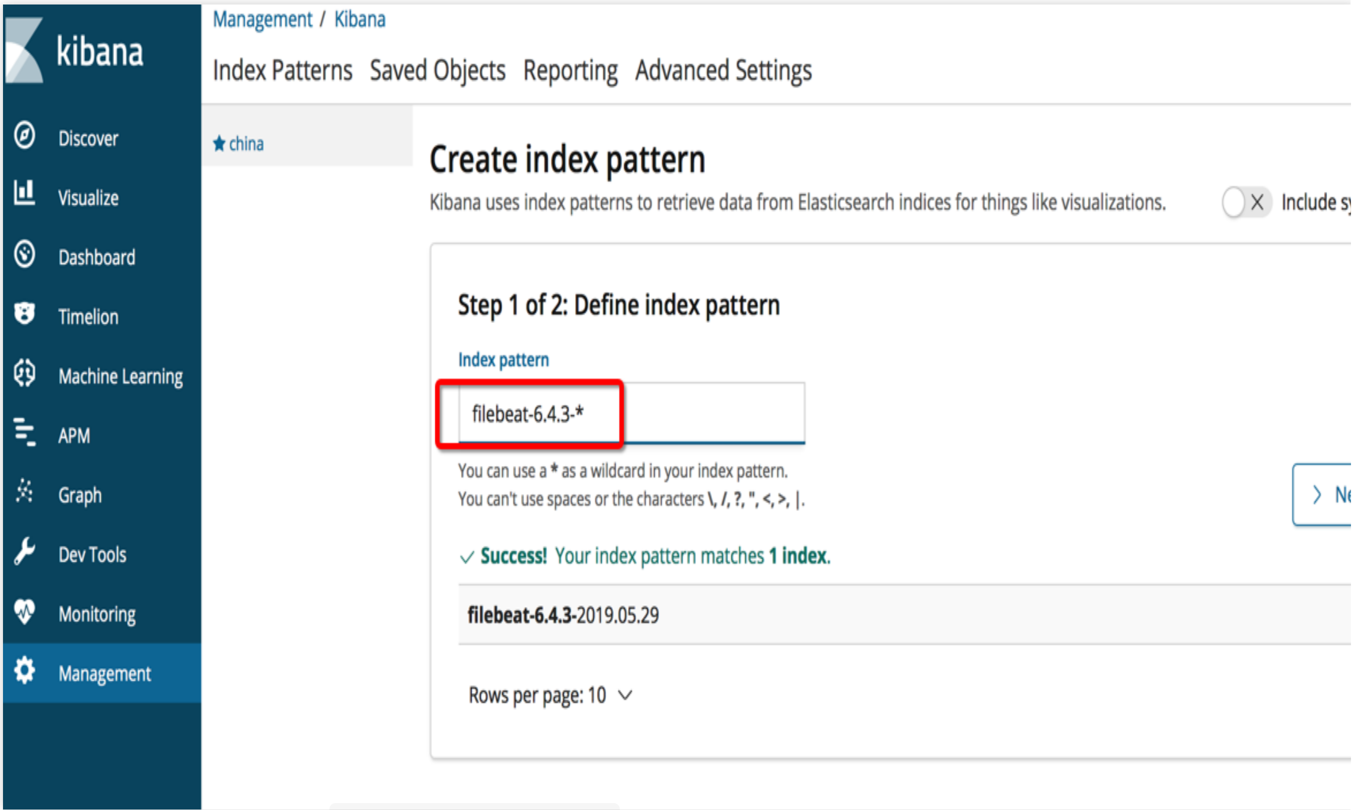
Querying a log

1. On the cluster list page in the ES Console, select **Operation** > **Kibana** to enter the Kibana Console.



ID/Name	Status	Node Specs	Nodes	Health Status	Availability ...	Network	ES Version	Billing Type	Operation
es-l3vgjti9 brown...	Normal	1 core 2GB 100GB Pre...	2	Green	Shanghai Z...	vpc-3l6kkpzc vpc-sh-1	6.4.3 Basic edition	Pay as you go Created on 2019-1... 16:01:43	Kibana Cloud Monitor More

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



Create index pattern

Kibana uses index patterns to retrieve data from Elasticsearch indices for things like visualizations.

Step 1 of 2: Define index pattern

Index pattern

filebeat-6.4.3-*

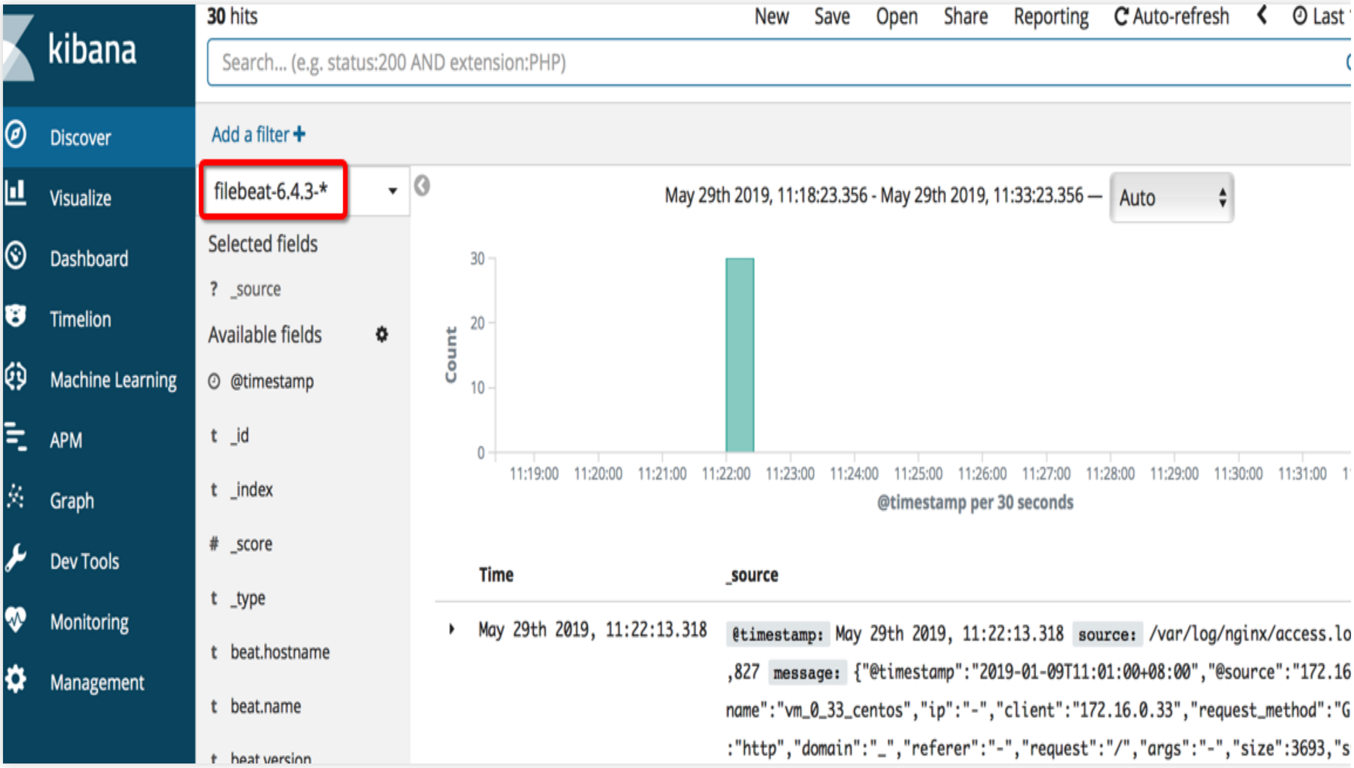
You can use a * as a wildcard in your index pattern.
You can't use spaces or the characters \, /, ?, ", <, >, |.

✓ Success! Your index pattern matches 1 index.

filebeat-6.4.3-2019.05.29

Rows per page: 10

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



30 hits

Search... (e.g. status:200 AND extension:PHP)

filebeat-6.4.3-*

May 29th 2019, 11:18:23.356 - May 29th 2019, 11:33:23.356 —

Time	_source
May 29th 2019, 11:22:13.318	@timestamp: May 29th 2019, 11:22:13.318 source: /var/log/nginx/access.lo ,827 message: {"@timestamp":"2019-01-09T11:01:00+08:00","@source":"172.16 name":"vm_0_33_centos","ip":"-","client":"172.16.0.33","request_method":"G :"http","domain":"_","referer":"-","request":"/","args":"-","size":3693,"s

Logstash + Elasticsearch + Kibana

Environment preparations

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.

A CVM instance should have at least 2 GB of memory.

You need to install Java 8 or above on the CVM instances.

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, NGINX log is used as the input source, and the output is configured as the private VIP address and port of the ES cluster. Create a `test.conf` configuration file, as shown below:

```
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 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 extracted `logstash-6.4.3` directory of the Logstash package and run the following command to start

Logstash in the background. Be sure to enter the path you create as the path to the configuration file.

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

View the `logs` directory in the `logstash-6.4.3` directory to confirm that Logstash has been started and can record the following logs:

```
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 a log

See Querying a log.

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

Index Configuration

Default Index Template Description and Adjustment

Last updated：2024-12-02 11:24:42

Default template description

An index template is a predefined template that is automatically applied when a new index is created. It typically contains configuration items for index settings, mapping, and template priority. ES offers a default index template for cluster creation, which can be viewed by running the `GET _template/default@template` command in **Dev Tools** on the Kibana page. The following describes the default template and its configuration items, which can be adjusted based on your actual needs.

```
{
  "default@template": {
    "order": 1, // Priority of the template. The greater the value, the higher the priority
    "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": {}
  }
}
```

Template adjustment

You can create your own custom index template by running the `PUT _template/my_template` command in **Dev Tools** on the Kibana page. Then, you can overwrite the configurations in the default index template by setting the `order` parameter of your own template to be greater than that of the default template.

Note:

The index template is only applied when an index is created; therefore, template adjustment will not affect existing indices.

Adjusting 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 value of the `order` field in the
template is greater than 1
    "settings" : {
        "index": {
            "number_of_shards" : 1
        }
    }
}
```

Adjusting field type

In the default template, fields of `string` type are dynamically mapped to `keyword` type in order to prevent full-text indexing of all the data of `text` type. You can modify the specified fields of `string` type to `text` based on your business needs to allow 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"
        }
    }
  }
}
```

Other business scenarios

For example, if you want that a document can be searched for 10 seconds after being indexed by any `search-*` index, you can create a template as shown below:

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

Managing Indices with Curator

Last updated：2024-12-02 11:25:33

Curator is Elasticsearch's official tool for managing indices. It can perform a wide variety of index lifecycle management tasks, such as clearing indices created 7 days ago, backing up specified indices regularly every day, and migrating indices from a hot node to a warm node regularly. For more information on operations in Curator, please see Actions.

Curator provides a CLI which allows you to configure the tasks to be executed using parameters. It also comes with a complete set of Python APIs that can be used together with SCF, such as automatically deleting expired data from Elasticsearch using Curator. SCF has a pre-defined template for Curator, which can be run after simple parameter configuration. For more information on how to use SCF, please see Serverless Cloud Function.

Curator Usage Example

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

Installation

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 filters out the indices named in the format of `logstash-20xx-xx-xx` and created 7 days ago and then deletes them.

Caution:

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 purpose 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, there are a lot of parameters, or you don't want to use command line parameters, you can place the parameter in configuration files.

In the specified `config` directory, you need to edit the config.yml and action.yml configuration files.

```
curator_cli --config PATH
```

Scheduled run

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

Using the API

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

Hot/Warm Architecture and Index Lifecycle Management

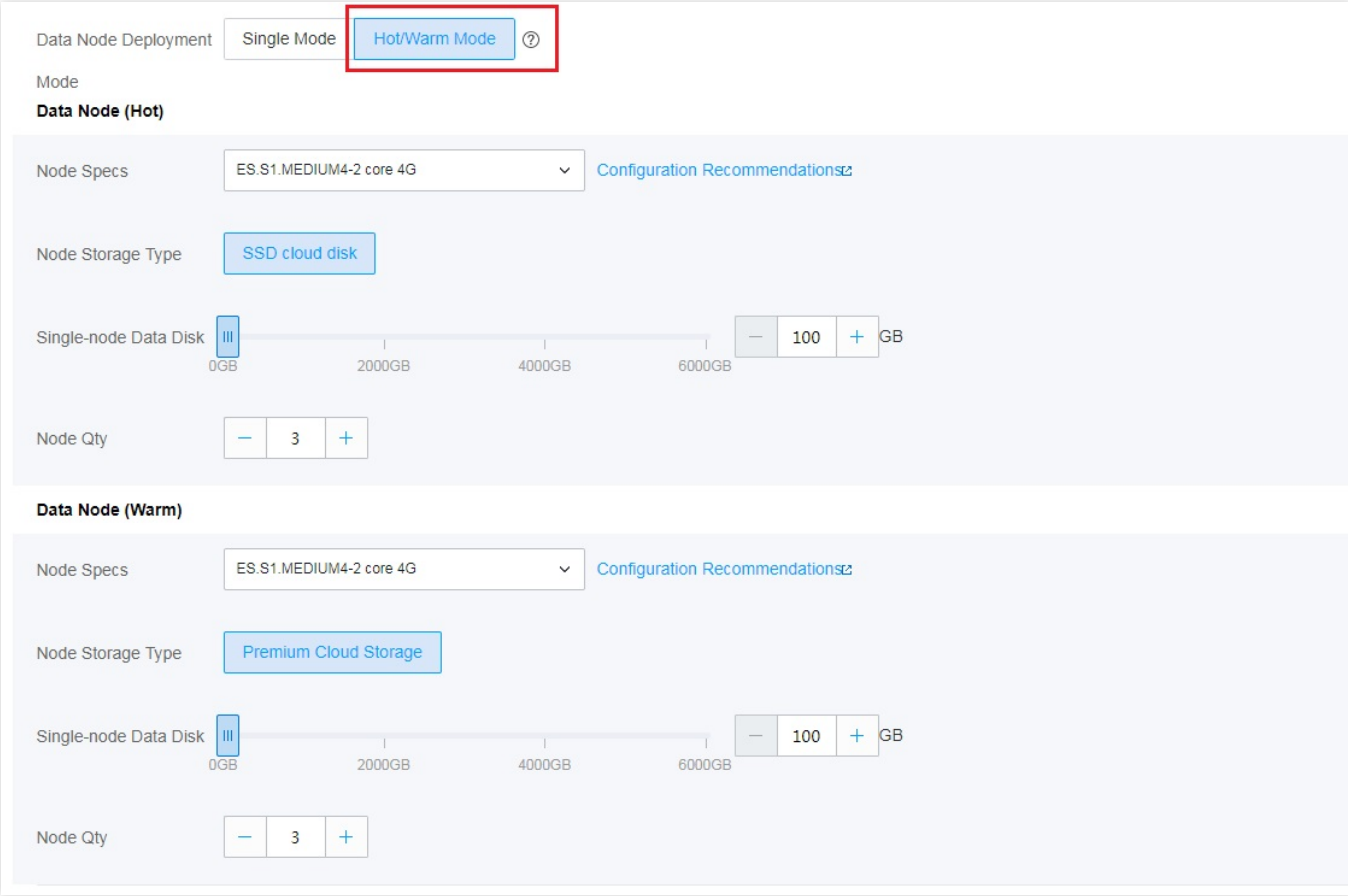
Last updated：2020-10-10 15:02:52

Elasticsearch is mainly used for store and retrieving massive amounts of data. In order to ensure the read/write performance, SSD disks are an ideal choice for storage; however, storing all data on SSD disks will undoubtedly be very costly. The classic hot-warm architecture of Elasticsearch can be used to solve this problem with ease. In a cluster under this architecture, different types of nodes can be configured to balance performance and capacity. Specifically, hot data that requires high read/write performance (such as logs generated within the past 7 days) can be stored on hot nodes with SSD disks. Indices that require large storage capacity but low read/write performance (such as logs generated more than one month ago) can be stored on warm nodes with SATA disks. This architecture not only guarantees the read/write performance of hot data, but also reduces the storage costs. Tencent Cloud ES provides the ability to quickly configure hot/warm clusters. You can specify the specifications of hot and warm nodes at Tencent Cloud's official website based on your business needs and build an ES cluster in the hot-warm architecture in minutes.

Creating Hot/Warm Cluster

Configuring directly during cluster purchase

1. Go to the ES cluster creation page and enter the relevant information for creating the cluster.
2. Select hot/warm mode as the data node deployment mode and select the hot and warm node specifications as shown below.



Data Node Deployment Mode: Single Mode | Hot/Warm Mode (selected)

Data Node (Hot)

- Node Specs: ES.S1.MEDIUM4-2 core 4G
- Node Storage Type: SSD cloud disk
- Single-node Data Disk: 100 GB
- Node Qty: 3

Data Node (Warm)

- Node Specs: ES.S1.MEDIUM4-2 core 4G
- Node Storage Type: Premium Cloud Storage
- Single-node Data Disk: 100 GB
- Node Qty: 3

3. Set other parameters of the cluster, confirm, and pay.

Transforming existing cluster into hot/warm cluster

On the cluster management page, click **More Operations** in the top-right corner, select **Adjust Configuration** from the drop-down list, and select **Hot/Warm Mode**. Set the hot and warm node specifications and relevant configuration items as needed to convert the existing cluster into a hot/warm cluster.

Using Hot/Warm Cluster

Viewing node role

To check the hot/warm attribute of a node, run 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
...
```

Specifying hot/warm attribute for index

You can determine the hot/warm attribute of an index according to the actual business situation.

For hot data, set the index as follows:

```
PUT hot_data_index/_settings
{
"index.routing.allocation.require.temperature": "hot"
}
```

For warm data, set the index as follows:

```
PUT warm_data_index/_settings
{
"index.routing.allocation.require.temperature": "warm"
}
```

Verifying index settings

1. Create an index.

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

2. View shard allocation. As you can see, the shards are evenly allocated across the 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 as hot index.

```
PUT hot_warm_test_index/_settings
{
     "index.routing.allocation.require.temperature": "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 warm index.

```
PUT hot_warm_test_index/_settings
{
     "index.routing.allocation.require.temperature": "warm"
}
```

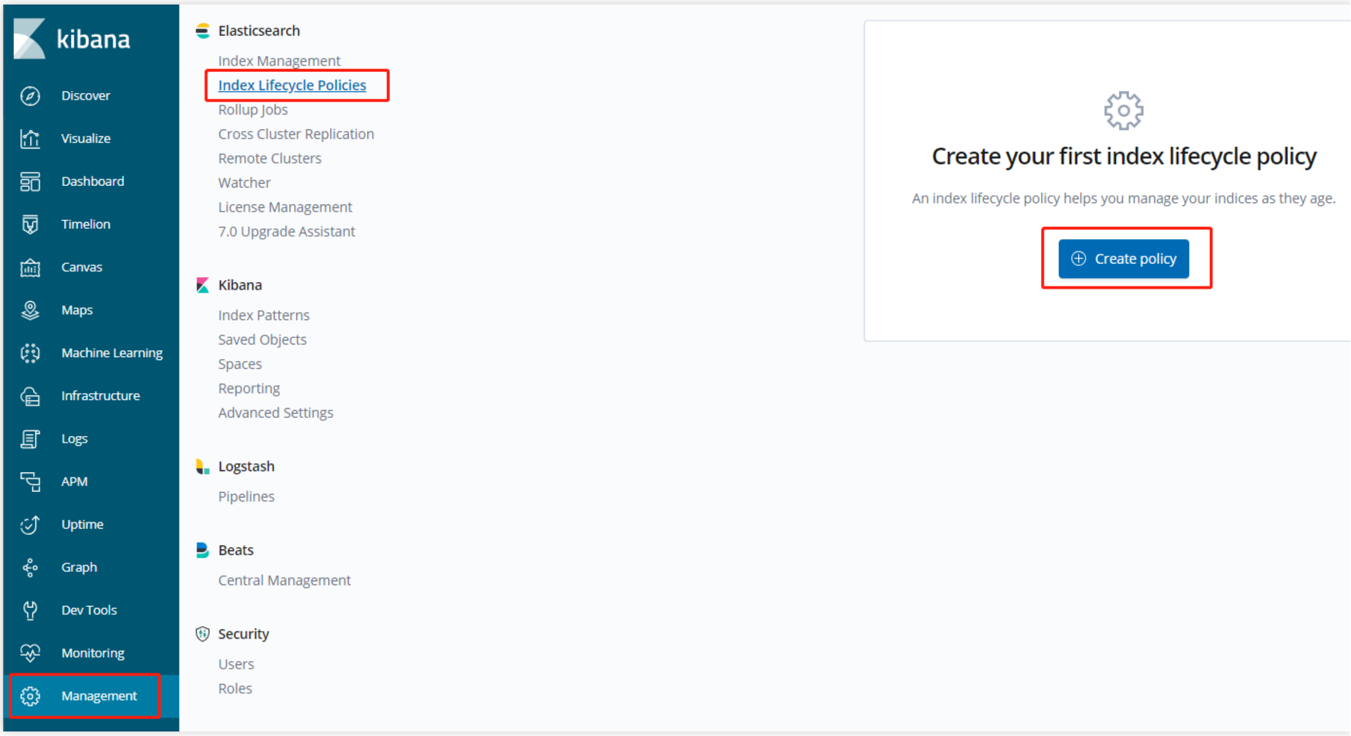
View the shard allocation. As you can see, all shards are allocated on warm nodes.

```
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 ES currently provides clusters on v6.8.2. This version of Elasticsearch (above v6.6) has an index lifecycle management feature that can be configured through API or Kibana. For more information, please see index-lifecycle-management. The following describes how to use index lifecycle management together with the hot-warm architecture to achieve dynamic management of indexed data in Kibana.

The index lifecycle management entry in Kibana is as shown below (v6.8.2):



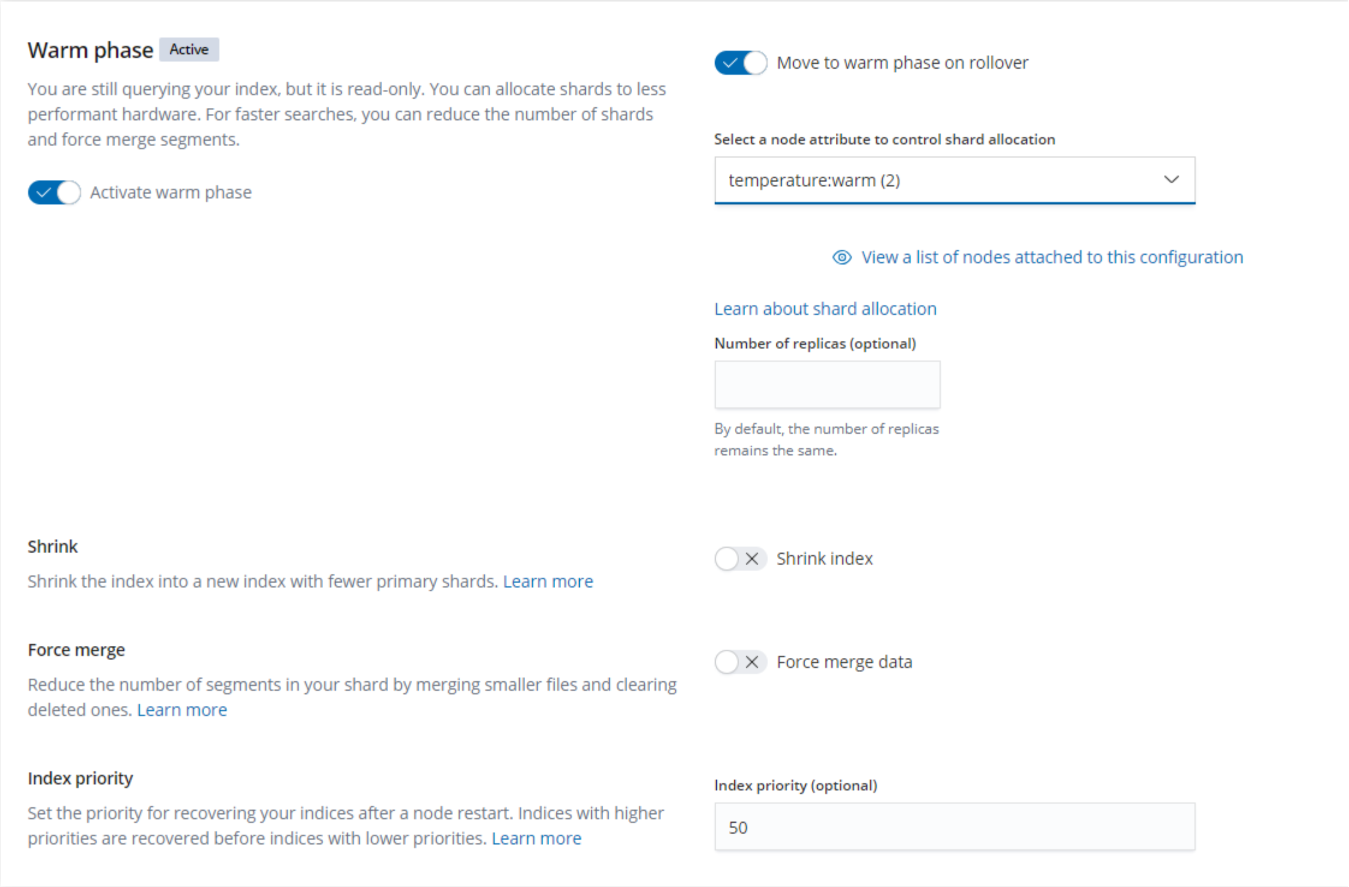
Create your first index lifecycle policy

An index lifecycle policy helps you manage your indices as they age.

Click **Create policy** to enter the configuration page. The lifecycle of an index can be divided into four phases: `Hot phase` , `Warm phase` , `Cold phase` , and `Delete phase` .

Hot phase: in this phase, you can decide whether to call the `rollover` API to roll over the index according to the number, size, and duration of the indexed documents. For more information, please see indices-rollover-index. As this phase has little to do with this document, it will not be detailed here.

Warm phase: when an index is rolled over in `Hot phase` , it will enter the `Warm phase` and will be set to read-only. You can set the attribute to be used for this index. For example, for the hot/warm separation policy, the `temperature: warm` attribute can be selected here. In addition, you can perform operations such as `forceMerge` and `shrink` on the index. For more information on these two operations, please see shrink API and force merge.



Warm phase Active

You are still querying your index, but it is read-only. You can allocate shards to less performant hardware. For faster searches, you can reduce the number of shards and force merge segments.

Activate warm phase

Move to warm phase on rollover

Select a node attribute to control shard allocation

temperature:warm (2)

View a list of nodes attached to this configuration

Learn about shard allocation

Number of replicas (optional)

By default, the number of replicas remains the same.

Shrink

Shrink the index into a new index with fewer primary shards. Learn more

Shrink index

Force merge

Reduce the number of segments in your shard by merging smaller files and clearing deleted ones. Learn more

Force merge data

Index priority

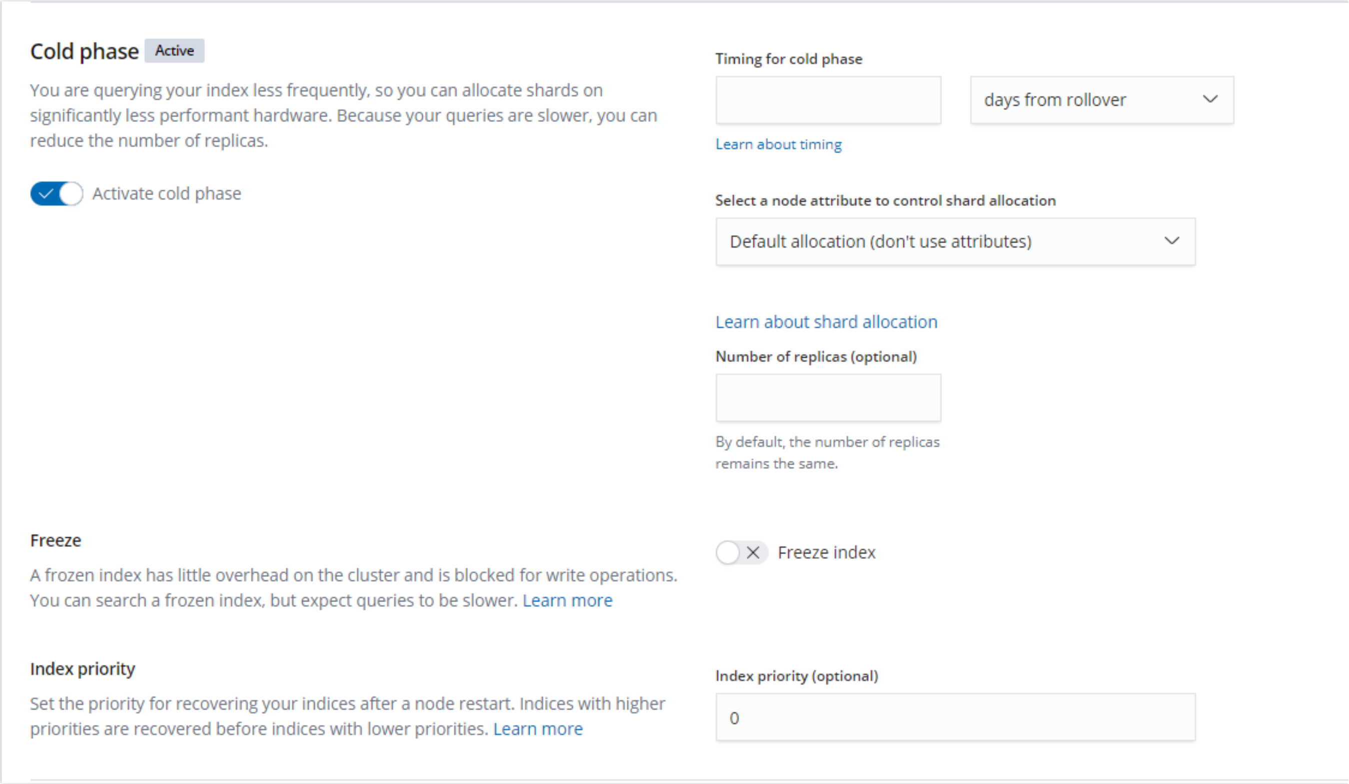
Set the priority for recovering your indices after a node restart. Indices with higher priorities are recovered before indices with lower priorities. Learn more

Index priority (optional)

50

Cold phase: you can set an index to roll over for a certain period of time and then enter the `Cold phase` . You can also set an attribute for this phase. It can be seen from the hot-warm architecture that the hot/warm attribute is extensible. Not only can you specify hot and warm attributes, but you can also expand and add multiple attributes such as `hot` , `warm` , `cold` , and `freeze` . If you want to use three layers of data separation, you can specify `temperature: cold` . You can also `freeze` the index. For more information, please see freeze API.

Delete phase: you can set the index to roll over for a certain period of time and then enter the `Delete phase` . Once in this phase, the index will be automatically deleted.



Cold phase Active

You are querying your index less frequently, so you can allocate shards on significantly less performant hardware. Because your queries are slower, you can reduce the number of replicas.

Activate cold phase

Timing for cold phase

days from rollover

Learn about timing

Select a node attribute to control shard allocation

Default allocation (don't use attributes)

Learn about shard allocation

Number of replicas (optional)

By default, the number of replicas remains the same.

Freeze

A frozen index has little overhead on the cluster and is blocked for write operations. You can search a frozen index, but expect queries to be slower. Learn more

Freeze index

Index priority

Set the priority for recovering your indices after a node restart. Indices with higher priorities are recovered before indices with lower priorities. Learn more

Index priority (optional)

0

SQL Support

Last updated：2022-08-17 11:29:01

Tencent Cloud Elasticsearch Service (ES) supports SQL instead of DSL as the query language. For those engaged in product operations and data analysis and new ES users, using SQL for queries can reduce their learning costs 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"
}
```

For more information on the API of the native SQL parser and how to use it, please see SQL REST API.

Open-Source SQL Parsing Plugin

v7.5.1:

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

Other versions:

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

For more information on the API of the SQL plugin and how to use it, please see here.

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 here or by adding the following dependencies in Maven:

```
<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：2024-12-02 11:42:47

Tencent Cloud Elasticsearch Service (ES) Platinum edition supports the X-Pack Watcher feature. After you configure triggers, actions, etc., specific actions will be triggered when specified conditions are met. For example, when an error log is detected in the index, an alert will be automatically sent. This document describes how to configure the WeCom bot to receive alerts from Watcher.

Note:

X-Pack Watcher is only available in the Platinum edition.

Due to the adjustment of the Tencent Cloud ES network architecture, only instances created in June 2020 or later support configuring WeCom bot to receive alerts from Watcher.

Background

With X-Pack Watcher, you can create watches. A watch consists of the following four parts:

Trigger: defines when to start executing a watch. A trigger must be configured for each watch. For more information about supported triggers, see Schedule trigger.

Input: specifies the query criteria for the monitored index execution. When a watch is triggered, its input loads data into the execution context. This context is accessible during subsequent watch execution phases. For more information, see Inputs.

Condition: defines the condition that needs to be met to execute the actions.

Actions: defines the actions to execute when the specified condition is met. For example, the webhook action described in this document.

Directions

1. Prepare a CVM instance that is in the same VPC as the ES cluster and can access the webhook address (via a public network).

2. Install Nginx on the CVM instance. For detailed installation steps, see here.

3. Configure Nginx agent forwarding. Replace the configuration in the `Server` section of the `nginx.conf` file with the following configuration:

The default port of the Nginx service is 80. If you need to change the port, log in to the Tencent Cloud console and go to Security Group to allow this port.

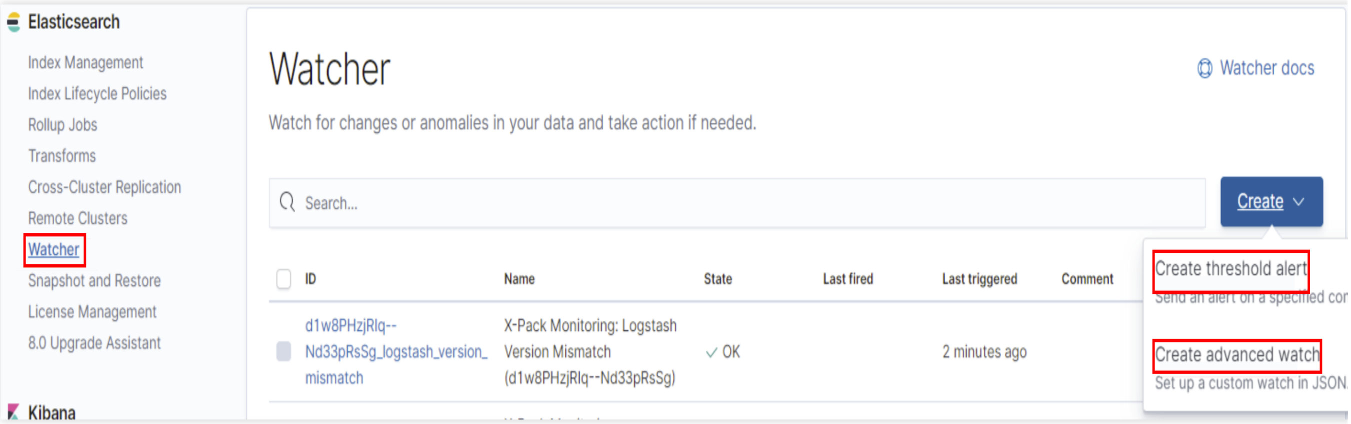
<WeCom bot webhook address>: enter the WeCom bot webhook address that receives alerts.

```
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 <WeCom bot webbook 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 alerting rule for a watch. **This step can be performed in **Management** > **Watcher** in the Kibana UI.



Watcher

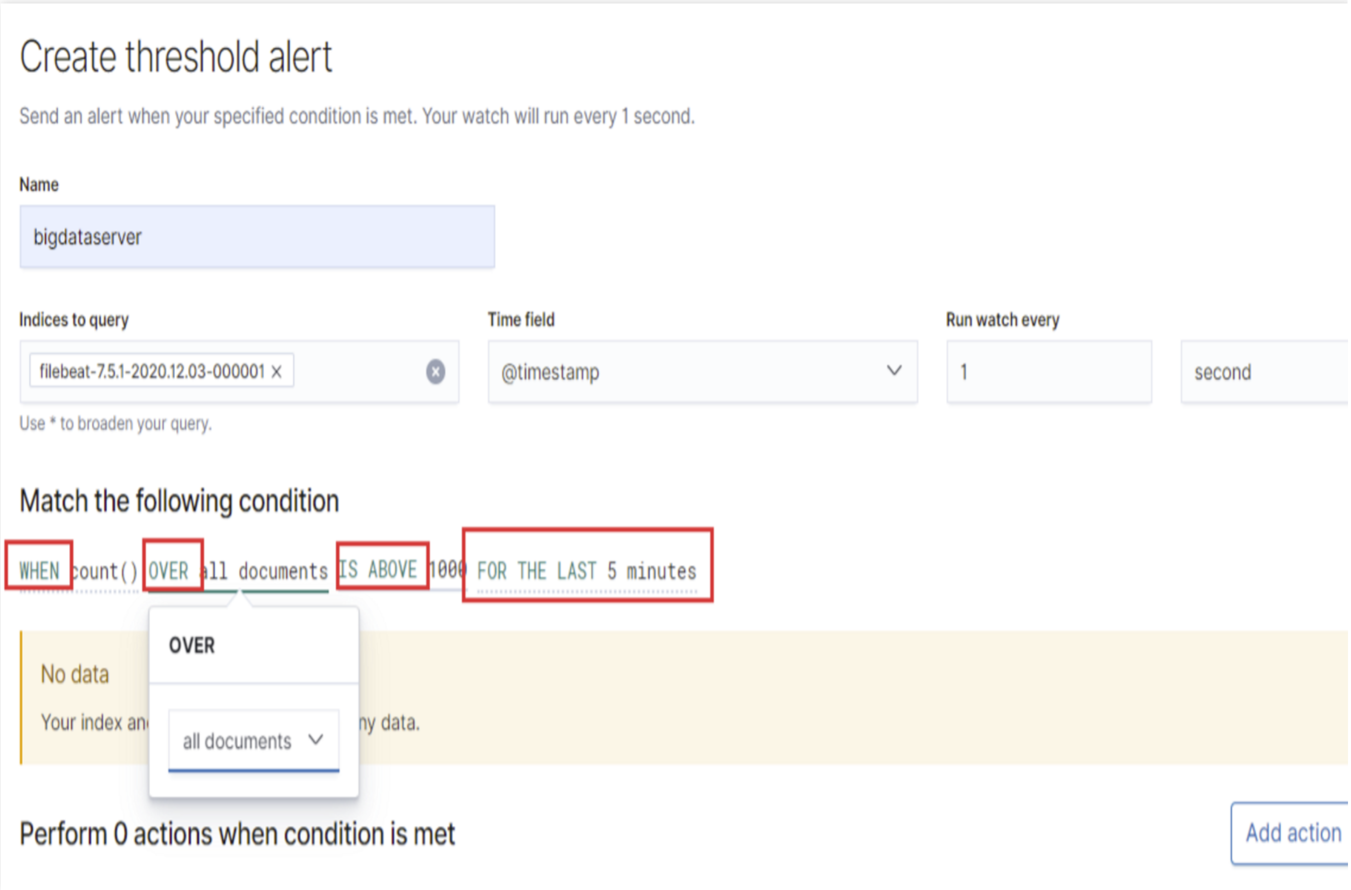
Watch for changes or anomalies in your data and take action if needed.

Send an alert on a specified cor

Set up a custom watch in JSON

ID	Name	State	Last fired	Last triggered	Comment
d1w8PHzjRlq--Nd33pRsSg_logstash_version_mismatch	X-Pack Monitoring: Logstash Version Mismatch (d1w8PHzjRlq--Nd33pRsSg)	✓ OK		2 minutes ago	

`Create threshold alert` : you can set threshold alerts to monitor specific conditions of an index, such as CPU usage, number of documents, etc. You can make more detailed settings in the condition section, as shown in the following figure:



Create threshold alert

Send an alert when your specified condition is met. Your watch will run every 1 second.

Name: bigdataserver

Indices to query: filebeat-7.5.1-2020.12.03-000001

Use * to broaden your query.

Time field: @timestamp

Run watch every: 1 second

Match the following condition

WHEN count() OVER all documents IS ABOVE 1000 FOR THE LAST 5 minutes

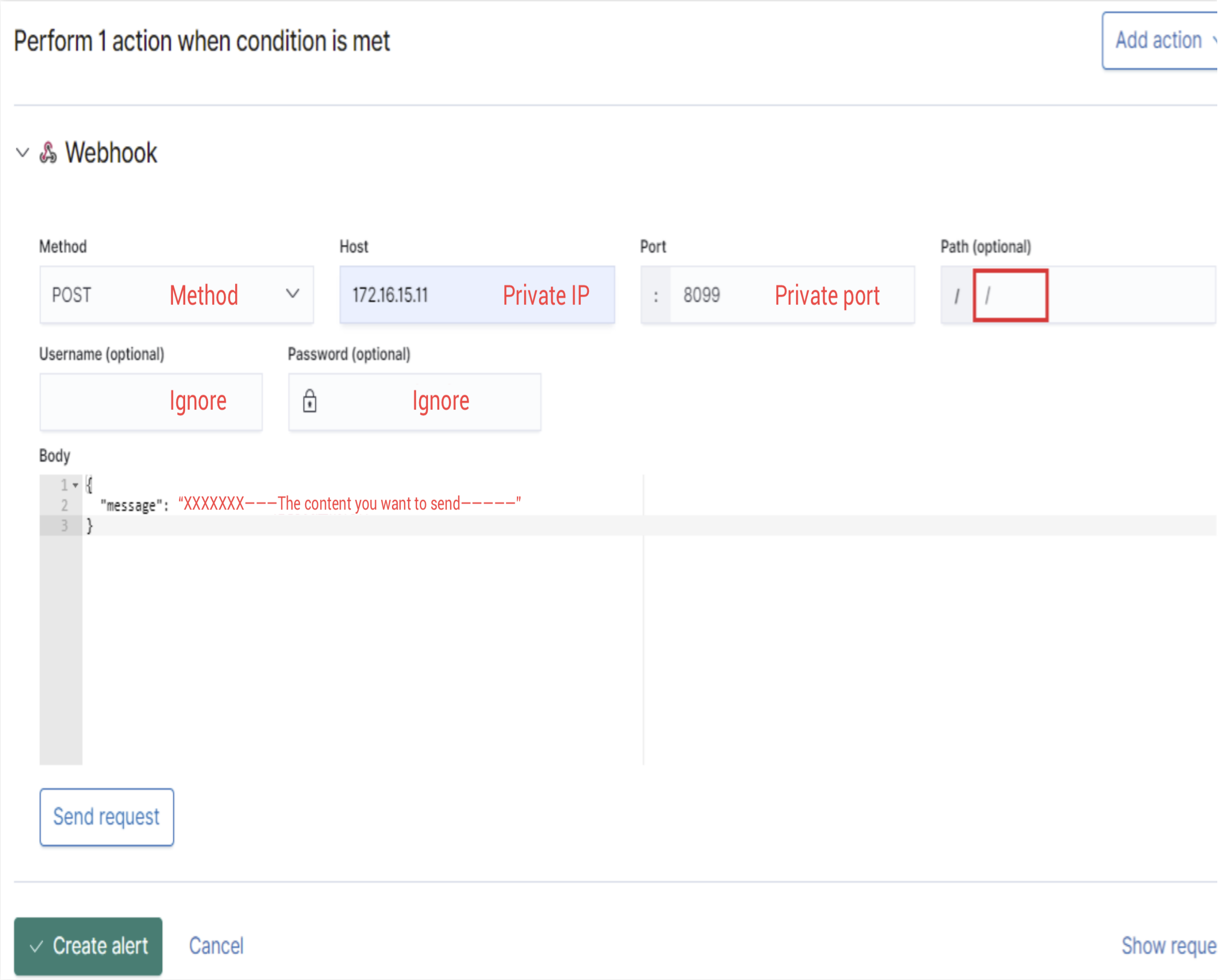
OVER: all documents

No data

Your index an... ny data.

Perform 0 actions when condition is met

Click **Add action** in the upper right corner, select **Webhook**, and configure related settings, as shown in the following figure:



Perform 1 action when condition is met

Webhook

Method: POST Method

Host: 172.16.15.11 Private IP

Port: 8099 Private port

Path (optional): /

Username (optional): Ignore

Password (optional): Ignore

Body

```
{
  "message": "XXXXXXX---The content you want to send-----"
}
```

Click **Send request** to perform a test and then click **Create alert**.

`Create advanced watch` : you can set the parameters for a watch via the PUT watch API.

6. After finishing the above steps, you can receive alerts from WeCom bot in the group you created.