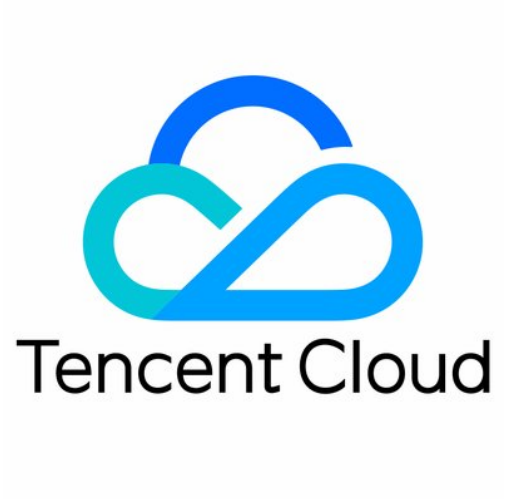
Elasticsearch Service

FAQs

Product Documentation



Contents

FAQs
Product

Last updated : 2020-11-10 16:24:29

What is Elasticsearch?

Elasticsearch is a distributed, scalable, and real-time search and data analysis engine based on Apache Lucene™, a leading full-text search engine library. It supports arbitrary expansion from a single node to hundreds of nodes and enables storage and query of petabytes of structured or unstructured data.

What business scenarios is ES suitable for?

ES fully retains the features of Elasticsearch in terms of massive data search such as full-text search, quasi-real-time search, and structured search. It is widely used in business scenarios like log analysis and in-site search. It enables you to build log service systems for other Tencent Cloud services in use such as CVM and TKE and can also integrate search services into your existing business service frameworks.

What is Elasticsearch Service?

Tencent Cloud ES is a cloud-based cluster service built on the open-source search engine Elasticsearch. It inherits the openness, compatibility, and usability of Elasticsearch and provides a sufficiency of hardware resources, user-friendly graphical creation and management tools, and comprehensive technical and OPS support.

I have an unpaid switch order. Will the order still be valid if I upgrade the cluster configuration?

No. A new purchase order will be generated when the billing mode is switched from pay-as-you-go to monthly subscription, but if you change the configuration of the cluster before paying the order, the amount of the new purchase order will not match the cluster, and the unpaid order cannot be paid.

If you need to switch the billing mode of the cluster, cancel the unpaid order in the Order Center before proceeding with the switch.

Can I change the cloud disk type after a successful purchase?

Currently, switching types of cloud disks is not supported. You can create a snapshot for backup and then use the snapshot to create a cloud disk of your desired type.

How do I choose the appropriate node and disk when creating a cluster?

To evaluate the node specification of the ES service, please see Evaluation of Cluster Specification and Capacity Configuration

Read-only Disks

Last updated : 2020-08-14 11:05:02

Elasticsearch 5.6.4 does not have a disk capacity threshold that helps prevent the disk capacity from being used up. Therefore, if you use Elasticsearch 5.6.4, you are recommended to pay close attention to the disk utilization and delete useless indices or expand the disk capacity in a timely manner.

Elasticsearch 6.4.3 introduces the disk watermark threshold. When the disk utilization reaches `99%`, the indices on the disk that have shards will become read-only, and you won't be able to write data to these indices. You are recommended to clear indices or expand the disk capacity promptly when you find that the disk utilization is high.

Issues

Creation of an index pattern timed out

After your cluster's disk status changes to read-only, you cannot create an index pattern on Kibana. This is because when the disk capacity is used up, ES will automatically set the block level of the index to `readonly_allow_delete`.

Error writing to an index

After your cluster's disk status changes to read-only, you cannot write data to an index even after clearing indices or expanding the disk capacity. The error is as follows:

- Index read-only error:

```
Elasticsearch exception [
type=cluster_block_exception,
reason=blocked by: [FORBIDDEN/12/index read-only / allow delete (api)];
]
```

- Cluster read-only error:

```
Elasticsearch exception [
type=cluster_block_exception,
reason=blocked by: [FORBIDDEN/13/cluster read-only / allow delete (api)];
]
```

Solutions

You need to clear useless indices to free up storage space or expand the disk capacity first, and then run the following command in **Dev Tools** in the Kibana UI:

- Turn off index read-only status:

```
PUT _all/_settings
{
"index.blocks.read_only_allow_delete": null
}
```

- Turn off cluster read-only status:

```
PUT _cluster/settings
{
"persistent": {
"cluster.blocks.read_only_allow_delete": null
}
}
```

Write Rejection (Bulk Reject)

Last updated : 2020-05-12 15:01:54

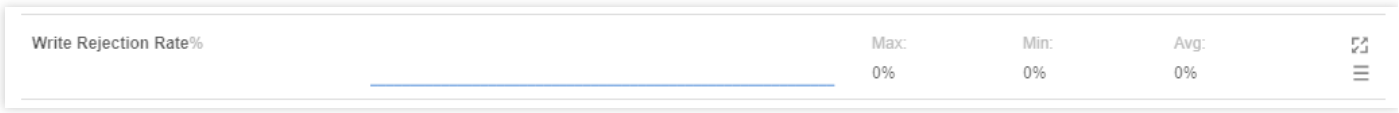
How do I fix the error with write rejection (bulk reject)?

Issue

In some cases, the bulk rejection rate of a cluster increases. Specifically, an error message such as the following will appear when bulk writes are performed:

```
[2019-03-01 10:09:58][ERROR]rspItemError: {"reason":"rejected execution of org.elasticsearch.transport.TransportService$7@5436e129 on EsThreadPoolExecutor[bulk, queue capacity = 1024, org.elasticsearch.common.util.concurrent.EsThreadPoolExecutor@6bd77359[Running, pool size = 12, active threads = 12, queued tasks = 2390, completed tasks = 20018208656]]","type":"es_rejected_execution_exception"}
```

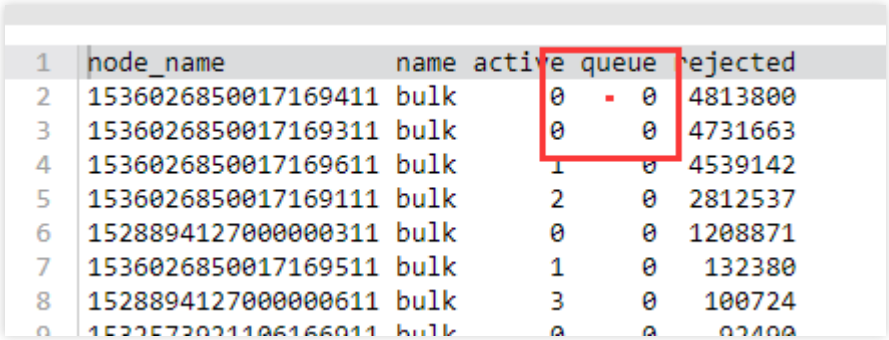
- You can see that the bulk rejection rate has increased in Cloud Monitor.



- You can also view the number of bulk writes that are being rejected or have been rejected by running the following command in the Kibana Console.

  ```
  GET _cat/thread_pool/bulk?s=queue:desc&v
  ```

  Generally, the default value for a queue is 1024. If there is 1024 under a queue, rejections have occurred on the node.



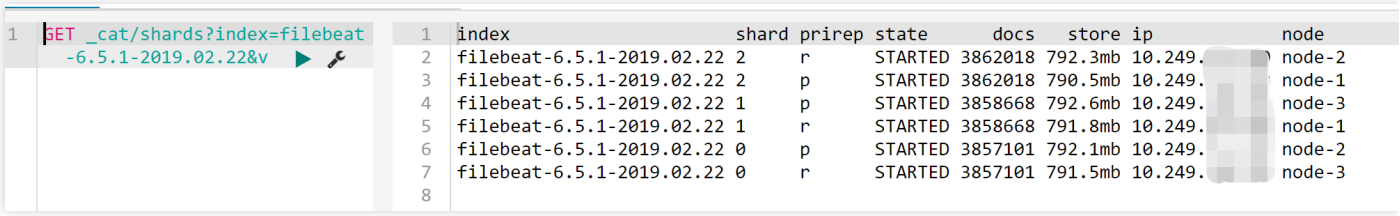
	node_name	name	active	queue	rejected
2	1536026850017169411	bulk	0	0	4813800
3	1536026850017169311	bulk	0	0	4731663
4	1536026850017169611	bulk	1	0	4539142
5	1536026850017169111	bulk	2	0	2812537
6	1528894127000000311	bulk	0	0	1208871
7	1536026850017169511	bulk	1	0	132380
8	1528894127000000611	bulk	3	0	100724

Cause

Bulk rejections are typically caused by a too large shard capacity or uneven allocation of shards. The specific cause can be identified and analyzed by following the steps below.

1. Check whether the shard size is too large
   A too large shard size may result in bulk rejections; therefore, you are recommended to limit the size of a shard to 20-50 GB. You can view the size of each shard in the index by running the following command in the Kibana console.

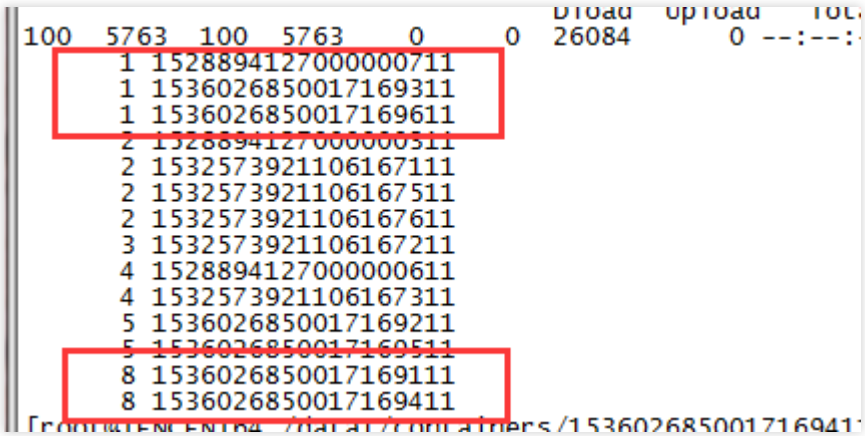
```
GET _cat/shards?index=index_name&v
```



2. Check whether the shards are unevenly distributed
   Sometimes, the shards may be unevenly distributed across all the nodes in the cluster. Some nodes are allocated with too many shards, while some too few.
   - You can check that in **Cluster Monitoring** > **Node Status** on the cluster details page in the ES Console. For more information, please see Viewing Monitoring Data.
   - You can also view the number of shards allocated to each node in the cluster using the curl client.

```
curl "$p:$port/_cat/shards?index={index_name}&s=node,store:desc" | awk '{print $8}' | sort | uniq -c | sort
```

   The results are as follows (the first column shows the number of shards, and the second shows the node ID), where some nodes are allocated with one shard, while some eight.



Solution

1. Set the shard size
   The shard size can be configured using the `number_of_shards` parameter in the index template (after the template is created, it will take effect when you create new indexed, and previous indices will not be adjusted).
2. Fix the uneven distribution of shards
   - Temporary solution:
     If you find that shards are not evenly allocated, you can dynamically adjust a certain index by setting the `routing.allocation.total_shards_per_node` parameter. For more information, please see Total Shards Per Node.

     > A certain buffer should be reserved for `total_shards_per_node` so as to prevent any machine failure from rendering allocation of shards impossible (for example, if there are 10 machines and an index has 20 shards, `total_shards_per_node` should be set to above 2, such as 3).

     Reference command:

     ```
     PUT {index_name}/_settings
     {
     "settings": {
     "index": {
     "routing": {
     "allocation": {
     "total_shards_per_node": "3"
     }
     }
     }
     }
     }
     ```

   - Set an index before production:
     Set the number of shards per node through the index template.

     ```
     PUT _template/{template_name}
     {
     "order": 0,
     "template": "{index_prefix@}*", // Prefix of the index to be adjusted
     "settings": {
     "index": {
     "number_of_shards": "30", // Specify the number of shards allocated to the index based on a shard size of about 30 GB
     "routing.allocation.total_shards_per_node":3 // Specify the maximum number of shards that a
     ```

```
node can accommodate
}
},
"aliases": {}
}
```

ES Circuit Breakers

Last updated : 2020-08-14 11:04:36

Elasticsearch provides multiple circuit breakers to prevent ES cluster errors caused by OutOfMemoryError when the memory utilization is too high. It is equipped with various types of child circuit breakers to specify the limit on memory available to specific requests. In addition, there is a parent circuit breaker that is used to specify the total amount of memory available across all child circuit breakers.

ES Circuit Breakers

One of the drawbacks of Elasticsearch circuit breaker mechanism is that only those requests that are often problematic are tracked to estimate the memory utilization, making it impossible to limit the amount of memory available to requests or trigger a circuit breaker based on the actual memory utilization on the current node. ES has a proprietary circuit breaker to address this problem with JVM OLD memory utilization.

This circuit breaker monitors the JVM OLD memory utilization. When the utilization exceeds `85%`, write requests will be rejected. If GC still cannot recycle the JVM OLD memory, query requests will be rejected when the utilization reaches `90%`. If a request is rejected, the client will receive the following response:

```
{
”status”: 403,
”error”: {
”root_cause”: [{
”reason”: ”pressure too high, (smooth) bulk request circuit break”,
”type”: ”status_exception”
}],
”type”: ”status_exception”,
”reason”: ”pressure too high, (smooth) bulk request circuit break”
}
}
```

The above error message indicates that the JVM OLD memory is currently heavily loaded and you need to free up some memory in JVM before retrying.

Common Methods to Free up Memory

- Clear fielddata cache: When you perform aggregation and sorting operations on fields of text type, the data structure called fielddata will be used, which may take up a lot of memory. You can view

the memory utilization of the fielddata in an index by running the following command in the **Dev Tools** on the Kibana page:

```
GET /_cat/indices?v&h=index,fielddata.memory_size&s=fielddata.memory_size:desc
```

If fielddata takes up a lot of memory, you can clear it by running the following command in the **Dev Tools** on the Kibana page:

```
POST /${Indices whose fielddata takes up a lot of memory}/_cache/clear?fielddata=true
```

- Clear segments: The FST structure of each segment will be loaded into the memory, which will not be recycled by GC. Therefore, a large number of segments in an index will also result in a high memory utilization. You can view the number of segments per node and the amount of memory used by them by running the following command in the **Dev Tools** on the Kibana page:

  ```
  GET /_cat/nodes?v&h=segments.count,segments.memory&s=segments.memory:desc
  ```

  If segments take up a too large amount of memory, you can delete indices that are not used, close indices, or regularly merge indices that are no longer updated.

- Scale out your cluster: If the circuit breaker is still triggered frequently after you clean up the memory, your cluster size may be no longer suitable for your business, and you are recommended to scale it out. For more information, see Scaling out a Cluster.