Cloud Search Service

Troubleshooting

Issue	01
Date	2026-08-21

HUAWEI CLOUD COMPUTING TECHNOLOGIES CO., LTD.

Trademarks and Permissions


and other Huawei trademarks are the property of Huawei Technologies Co., Ltd.
All other trademarks and trade names mentioned in this document are the property of their respective holders.

Notice

The purchased products, services and features are stipulated by the contract made between Huawei Cloud and the customer. All or part of the products, services and features described in this document may not be within the purchase scope or the usage scope. Unless otherwise specified in the contract, all statements, information, and recommendations in this document are provided "AS IS" without warranties, guarantees or representations of any kind, either express or implied.

The information in this document is subject to change without notice. Every effort has been made in the preparation of this document to ensure accuracy of the contents, but all statements, information, and recommendations in this document do not constitute a warranty of any kind, express or implied.

Huawei Cloud Computing Technologies Co., Ltd.

Address: Huawei Cloud Data Center Jiaoxinggong Road
Qianzhong Avenue
Gui'an New District
Gui Zhou 550029
People's Republic of China

Website: https://www.huaweicloud.com/intl/en-us/

Contents

1 Clusters

1.1 Unable to Launch Kibana

Symptom

When accessing Kibana for an Elasticsearch cluster, the page remains stuck on the loading screen and cannot properly enter the Kibana console.

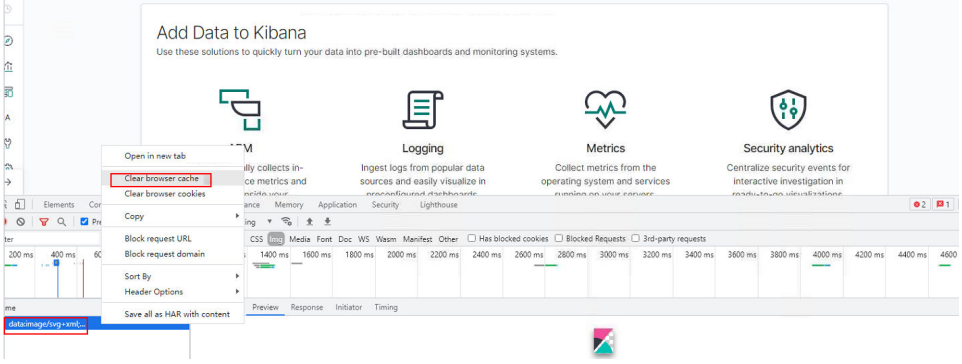
Possible Causes

This issue can be solved by clearing the browser cache.

Procedure

1. Log in to the **CSS management console**.
2. In the navigation pane on the left, choose **Clusters > Elasticsearch**.
3. In the cluster list, find the target cluster, and click **Kibana** in the **Operation** column to log in to the Kibana console.
4. On the **Kibana** console, press **F12**.
5. Click **Network**, right-click **data:image**, and choose **clear browser cache** from the shortcut menu. In the displayed dialog box, click **OK**. Close the Kibana window.

Figure 1-1 Closing the Kibana page



6. In the cluster list, find the target cluster, click **Kibana** in the **Operation** column to try logging in to the Kibana console again.

1.2 How to Improve Filebeat Performance

Symptom

Filebeat is a lightweight, high-performance log shipper that collects and forwards log files to designated centralized systems, such as Elasticsearch. While it typically collects logs within one second, performance can degrade with large files. This happens because Filebeat is allocated only one CPU core by default, limiting its write throughput to less than 1 MB per second. To handle large volumes, you can optimize its performance by adjusting parameters in the **filebeat.yml** file.

Possible Causes

For Filebeat, the default configuration of the **filebeat.yml** file cannot deliver optimal performance when handling large log volumes. In such scenarios, modify parameter settings in the **filebeat.yml** file to meet your demands.

Procedure

1. Optimize the parameters involved in **input** of the **filebeat.yml** configuration file.
   Increase the value of **harvester_buffer_size** based on actual requirements. This parameter defines the buffer size used by every **harvester**.
   harvester_buffer_size: **40,960,000**
   Increase the value of **filebeat.spool_size** based on actual requirements. This parameter defines the number of log records that can be uploaded by the **spooler** at a time.
   filebeat.spool_size: **250,000**
   Adjust the value of **filebeat.idle_timeout** based on actual requirements. This parameter defines how often the **spooler** is flushed. After the **idle_timeout** is reached, the **spooler** is flushed regardless of whether the **spool_size** has been reached.
   filebeat.idle_timeout: **1s**
2. Optimize the parameters involved in **output.elasticsearch** in the **filebeat.yml** configuration file.
   Set the value of **worker** to the number of Elasticsearch clusters based on actual requirements. This parameter indicates the number of Elasticsearch clusters. The default value is **1**.
   worker: **1**
   Increase the value of **bulk_max_size** based on actual requirements. This parameter defines the maximum number of events to bulk in a single Elasticsearch bulk API index request. The default is **50**.
   bulk_max_size: **15,000**
   Adjust the value of **flush_interval** based on the actual requirements. This parameter defines the number of seconds to wait for new events between two bulk API index requests. If **bulk_max_size** is reached before this interval expires, additional bulk index requests are made.
   flush_interval: **1s**

1.3 "Connection reset by peer" Is Returned When Using Elasticsearch with Spring Boot

Symptom

When a Spring Boot application uses the Java High Level REST Client to connect to an Elasticsearch cluster, the error **Connection reset by peer** is reported, the TCP connection is interrupted, and write failures occur.

Possible Causes

There are many causes for connection closure. For example, the server side may have closed the connection. There may be firewalls, switches, VPNs, or other network devices in the network. Or there may be a keepalive configuration issue, a change in the connected server node, network instability, etc.

Procedure

Use one of the following methods to solve the issue:

- Method 1

  Modify the timeout interval of RestHighLevelClient connection requests. The default value is 1000 ms. You can increase the value to 10,000 ms.

  ```
  RestClientBuilder builder = RestClient.builder(new HttpHost(endpoint, port))
          .setHttpClientConfigCallback(httpClientBuilder->
  httpClientBuilder.setDefaultCredentialsProvider(credentialsProvider))
          .setRequestConfigCallback(requestConfigBuilder ->
  requestConfigBuilder.setConnectTimeout(10000).setSocketTimeout(60000));
     return new RestHighLevelClient(builder);
  ```

  Settings for a single request:
  request.timeout(TimeValue.timeValueSeconds(60)).

- Method 2

  Set the RestHighLevelClient keepalive time to 15 minutes.

- Method 3

  Handle the exception captured in the code and retry the request.

Related Information

- TCP connections

  TCP connections are classified into persistent connections and short connections. A short TCP connection is automatically disconnected after data packets are sent. A persistent TCP connection uses the keepalive timer function, and remains open for a certain period of time after data packets are sent.

- TCP keepalive mechanism

  The keepalive mechanism is implemented using a timer. If the timer is activated, the server will send a keepalive probe packet. An ACK message is expected as a response. If the client does not respond, the server will terminate the connection. If the client responds, the keepalive timer will be reset.

  The keepalive duration on the server is set to **30m**. In Linux, three parameters can be used to control the keepalive duration: **tcp_keepalive_time** (idle duration for enabling the keepalive function), **tcp_keepalive_intvl** (interval for sending keepalive packets), and **tcp_keepalive_probes** (the number of times the keepalive packets are sent if no response is received).

- http-keepalive

  The http-keepalive mechanism enables a TCP connection to transmit as many packets as possible. The http-keepalive duration is updated each time a packet is transmitted. If the http-keepalive timer expires, it means that no messages have been exchanged between the client and server during this period, and the local side will actively close and release the connection.

The tcp-keepalive mechanism retains a TCP connection until the connection is deliberately closed.

1.4 Why Does Cluster Creation Fail?

The possible causes of a cluster creation failure are as follows:

- Insufficient resource quota. You are advised to increase the resource quotas.
- The value of **Port Range/ICMP Type** in **Security Group** does not include port **9200**. Modify the security group information or select another available security group.
- For cluster version 7.6.2 and later versions, the communication port 9300 is enabled on the subnet of user VPC by default. When you create a cluster, check whether the selected security group allows traffic from communication port 9300 in the subnet. If it does not allow traffic, modify the security group or select another security group.
- Insufficient permission. You are advised to obtain required permissions. For details, see **Permissions Management**.

1.5 "Bulk Reject" Is Displayed for an Elasticsearch Cluster

Symptom

Under certain circumstances, the cluster may experience an increase in the write rejection rate, known as "Bulk Reject." This is specifically manifested by errors similar to the following during bulk write operations:

```
[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"
}
```

Issue Analysis

The majority of Bulk Reject issues are caused by oversized shard capacity or uneven shard distribution. The specific causes can be identified and analyzed using the following methods.

1. Check whether the data volume in shards is too large.

   The recommended data size in a single shard is 20 GB to 50 GB. You can run the following command on the Kibana console to view the size of each shard of an index:

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

2. Check whether the shards in nodes are unevenly distributed.

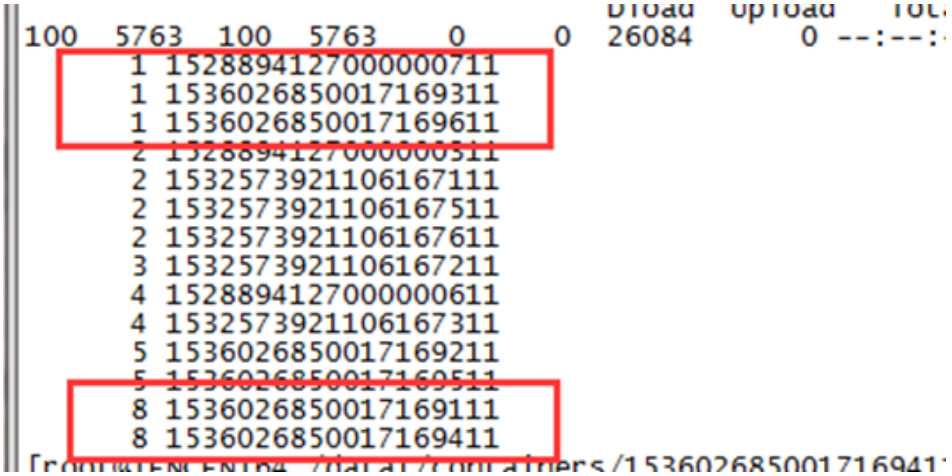
   You can check shard allocation in either of the following ways:

- Method 1: Check the cluster's monitoring metrics on the console, especially those related to index shards. For details, see **Viewing Monitoring Metrics**.
- Method 2: Use the CLI to check the distribution of index shards across cluster nodes.

  For example, run the curl command on the client to check the number of shards on each cluster node.

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

  Figure 1-2 Example of command output

  In the command output, the first column indicates the number of shards, and the second column indicates the node ID. In the result shown in the figure above, some nodes have only one shard, and some have eight shards. This means index shards are unevenly distributed.

Solution

- If the problem is caused by oversized shards:

  Set the value of the **number_of_shards** parameter in the **index** template to limit the shard size.

  A newly created template will be applied immediately to new indexes. Existing indexes cannot be changed.

- If the problem is caused by uneven shard distribution:

  Workarounds:

  a. You can run the following command to set the **routing.allocation.total_shards_per_node** parameter to dynamically adjust an index:

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



```
  }
}
```

CAUTION

When you configure the **total_shards_per_node** parameter, reserve some buffer space to avoid shard allocation failures caused by machine faults. For example, if there are 10 servers and the index has 20 shards, the value of **total_shards_per_node** must be greater than 2.

b. Before creating indexes, configure an index template to specify the quantity of index shards on each node.

```
PUT _template/<template_name>
{
   "order": 0,
   "template": "{index_prefix}*",  //The index prefix to be changed
   "settings": {
      "index": {
         "number_of_shards": "30", //Total number of shards allocated to nodes. The capacity of a shard can be assumed as 30 GB.
         "routing.allocation.total_shards_per_node": 3 //Maximum number of shards on a node
      }
   },
   "aliases": {}
}
```

1.6 Index Patterns Cannot Be Created for an Elasticsearch Cluster

Symptom

On the **Dev Tools** page of Kibana, run the **GET .kibana/_settings** command to query the status of the .kibana index. The value of the parameter **read_only_allow_delete** is **true**, which indicates that the cluster disk usage is too high.

```
{
  ".kibana_1" : {
    "settings" : {
      "index" : {
        "number_of_shards" : "1",
        "auto_expand_replicas" : "0-1",
        "blocks" : {
          "read_only_allow_delete" : "true"
        },
        "provided_name" : ".kibana_1",
        "max_result_window" : "100000",
        "creation_date" : "1602490470096",
        "number_of_replicas" : "0",
        "uuid" : "_T3td16IRt6605J1tAbKOQ",
        "version" : {
          "created" : "7060299"
        }
      }
    }
  }
}
```


Possible Cause

When the .kibana index becomes read-only, new index patterns cannot be created.

Solution

1. Log in to Kibana.
   a. Log in to the **CSS management console**.
   b. In the navigation pane on the left, choose **Clusters > Elasticsearch**.
   c. In the cluster list, find the target cluster, and click **Kibana** in the **Operation** column to log in to the Kibana console.
   d. In the left navigation pane, choose **Dev Tools**.

      The left part of the console is the command input box, and the triangle icon in its upper-right corner is the execution button. The right part shows the execution result.
2. Run the following command to change the **read_only_allow_delete** parameter to **false**.

```
PUT .kibana/_settings
{
    "index": {
        "blocks": {
            "read_only_allow_delete": false
        }
    }
}
```

3. Try creating new index patterns again.

1.7 "The System Is Busy" Is Displayed on the CSS Console

Symptom

In the navigation pane of the CSS console, choose **Clusters**, and the messages "The system is busy. Try again later or xxxx." and "Policy doesn't allow css:cluster:list to be performed." are displayed.

Possible Cause

The current account does not have the read permission on CSS. You need to assign the required permissions to the IAM account.

Solution

Grant permissions to the IAM account. For details, see **Permissions Management**.

1.8 An Elasticsearch Cluster Reports Error Message "unassigned shards all indices"

Symptom

An Elasticsearch cluster reports error message **unassigned shards all indices** and the cluster status is **red**.

Possible Causes

Unallocated shards exist in the current cluster.

Solution

1. Log in to Kibana.
   a. Log in to the **CSS management console**.
   b. In the navigation pane on the left, choose **Clusters > Elasticsearch**.
   c. In the cluster list, find the target cluster, and click **Kibana** in the **Operation** column to log in to the Kibana console.
   d. In the left navigation pane, choose **Dev Tools**.

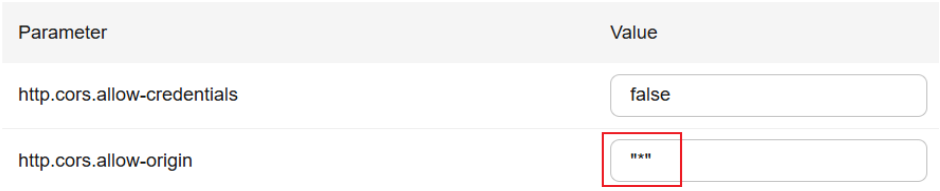
      The left part of the console is the command input box, and the triangle icon in its upper-right corner is the execution button. The right part shows the execution result.
2. Run the following command to re-allocate unallocated shards:
   ```
   POST  /_cluster/reroute?retry_failed=true
   ```

1.9 A Cross-Domain Error Is Returned When I Try to Connect to an Elasticsearch Cluster via the es-head Plugin

Solution

1. Check whether the network is connected on the ECS where the es-head plugin is installed.
2. After the network is connected, Log in to the **CSS management console**.
3. In the navigation pane on the left, choose **Clusters > Elasticsearch**.
4. In the cluster list, click the name of the target cluster. The cluster information page is displayed.
5. Choose **Cluster Settings** > **Parameter Settings** > **Elasticsearch**.
6. Click **Edit** and change the value of **http.cors.allow-origin** under **Cross-domain Access** to **"*"**. Note that the double quotation marks are part of the parameter value.

Figure 1-3 Modifying parameters



7. After the change is complete, click **Save**. In the displayed dialog box, confirm the settings, select the box indicating "I understand that the modification will take effect after the cluster is restarted." and click **Yes**.

   If the **Status** is **Succeeded** in the parameter change list, the change has been saved.

8. Click **Restart** in the upper right corner to restart the cluster, thus making the change take effect.

1.10 An Alarm Is Displayed When I Access Cerebro Through a Single-Node Cluster

Possible Causes

By default, indexes in single-node clusters have copies that cannot deliver requests.

Solution

On the **Dev Tools** page of Kibana, run the following commands to change the number of index copies to **0**:

```
PUT _all/_settings
{
  "index": {
    "number_of_replicas": 0
  }
}
```

1.11 Unable to Connect to a Cluster from an ECS

Perform the following steps to troubleshoot this problem:

1. Check whether the ECS instance and cluster are in the same VPC.
   - If they are, go to 2.
   - If they are not, create an ECS instance and ensure that the ECS instance is in the same VPC as the cluster.
2. View the security group rule setting of the cluster to check whether port **9200** (TCP protocol) is allowed or port **9200** is included in the port range allowed in both the outbound and inbound directions.
   - If it is allowed, go to 3.
   - If it is not allowed, switch to the VPC management console and configure the security group rule of the cluster to allow port **9200** in both the outbound and inbound directions.

3. Check whether the ECS instance has been added to a security group.
   - If a security group has been created, check whether its rules are appropriate. If they are, go to **4**.
   - If the instance has not been added to the security group, go to the VPC page from the ECS instance details page, select a security group, and add the ECS to the group.
4. Check whether the ECS instance can connect to the CSS cluster.

   ```
   ssh <Private network address and port number of a node>
   ```

   If the cluster contains multiple nodes, test connectivity to each node.
   - If the connection is normal, the network is running properly.
   - If the connection still fails, contact technical support.

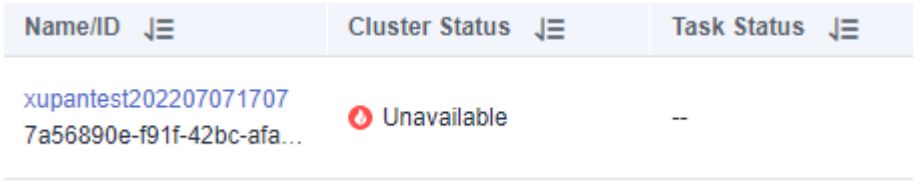
2 Unavailable Clusters

2.1 Cluster Status Is Unavailable

Symptom

A CSS cluster status is **Unavailable**.

Figure 2-1 Unavailable cluster



Cause Analysis and Solution

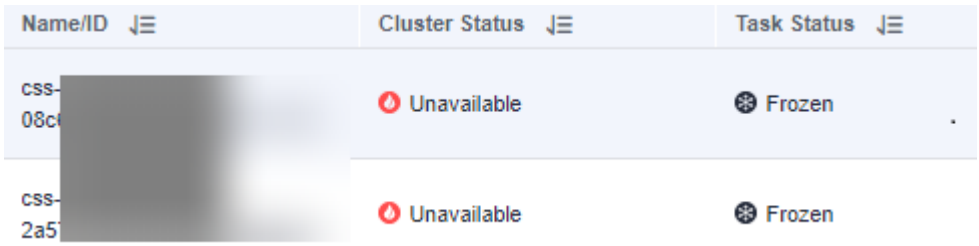
- If the task status is **Frozen**, see **A Cluster Is Frozen and Unavailable**.
- If the task status is **Configuration error. Restart failed**, see **A Cluster Is Unavailable Due to X-pack Parameter Configuration**.
- If the node log contains the error message "master not discovered or elected yet, an election requires at least 2 nodes with ids [xxx, xxx, xxx, ...], have discovered [xxx...] which is not a quorum", see **A Cluster Is Unavailable Due to Improper Security Group Rules**.
- If the node log contains the error message "fatal error in thread [main], exitingjava.lang. NoClassDefFoundError: xxx/xxx/.../xxxPlugin at ...", see **A Cluster is Unavailable Due to Plugin Incompatibility**.
- If the cluster health status is red and the value of **unassigned shards** is not 0, the cluster has index shards that cannot be allocated. For details, see **A Cluster is Unavailable Due to Improper Shard Allocation**.
- If a cluster is unavailable after being restored or migrated, see **A Cluster is Unavailable Due to Incompatible Data Types**.
- If the node log contains the error message "OutOfMemoryError" and warning information "[gc][xxxxx] overhead spent [x.xs] collecting in the last [x.xs]", see **A Cluster is Unavailable Due to Heavy Load**.

2.2 A Cluster Is Frozen and Unavailable

Symptom

The cluster status is **Unavailable**, and the task status is **Frozen**.

Figure 2-2 Frozen



Possible Causes

The cluster is frozen because the account is in arrears or the subscription period of a yearly/monthly cluster has expired.

Procedure

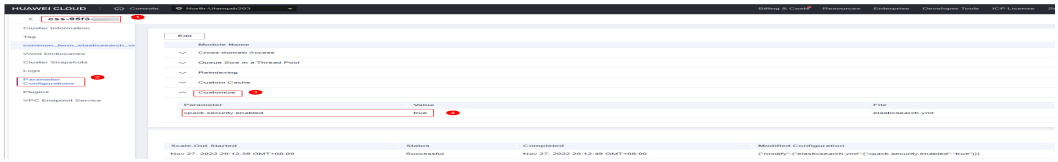
- **Pay-per-use clusters**
  a. Go to **Billing Center**.
  b. Choose **Overview** to view the outstanding amount of the account.
     - If you are an IAM user, log in to the account and top up the account.
     - If you are an account user, click **Top Up** to go to the top-up page.
  c. After confirming that the account balance is sufficient, wait for the cluster to automatically recover to an available state.
- **Yearly/Monthly clusters**
  a. Go to **Billing Center**.
  b. Choose **Orders** > **Renewals**.
  c. On the **Renewals** page, set **Expires** to **Frozen** and **Service Type** to **Cloud Search Service** to search for the frozen cluster yearly/monthly package.
  d. In the **Operation** column of the frozen cluster yearly/monthly package, click **Renew**. On the displayed page, renew the package as required.
  e. After the status of the yearly/monthly package changes from **Frozen** to **Provisioned**, wait until the status of the cluster becomes **Available**.

2.3 A Cluster Is Unavailable Due to X-pack Parameter Configuration

Symptom

The cluster status is **Unavailable**, and the task status of the cluster is **Configuration error. Restart failed**.

Figure 2-3 Incorrect cluster configuration



Possible Causes

You have configured custom X-pack parameters. CSS does not support the X-pack function.

Procedure

1. On the **Clusters** page, click the name of the unavailable cluster. The cluster information page is displayed.
2. Choose **Cluster Settings** > **Parameter Settings**.
3. Click the **OpenSearch Settings** or **Elasticsearch Settings** tab, depending on your cluster type.
4. Expand the **Custom** module, and check whether custom X-pack parameters exist.

   NOTE

   Examples of the custom X-pack parameters:

   - **xpack.security.enabled**: **true**
   - **xpack.security.http.ssl.enabled**: **true**
   - **xpack.security.http.ssl.keystore.path**: **http.p12**

   – If yes, go to the next step.
   – If no, contact technical support to locate the parameter configuration problem.
5. Delete the custom X-pack parameters.
   a. Click **Edit** and click **Delete** on the right of the parameter you want to delete.
   b. Click **Save**. In the displayed dialog box, select **I understand that the modification will take effect after the cluster is restarted** and click **OK**.

      If the **Status** is **Succeeded** in the parameter change list, the change has been saved.

c. Click **Restart** in the upper right corner to restart the cluster, thus making the change take effect.

d. After the cluster is restarted, the cluster becomes available.

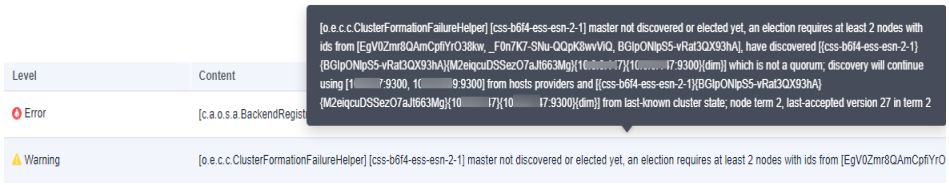
2.4 A Cluster Is Unavailable Due to Improper Security Group Rules

Symptom

The cluster status is **Unavailable**.

Click the cluster name to go to the cluster details page. Choose **Logs** > **Log Search**. The following error message is displayed: "master not discovered or elected yet, an election requires at least 2 nodes with ids [xxx, xxx, xxx, ...], have discovered [xxx...] which is not a quorum".

Figure 2-4 Node error logs



Possible Causes

In the preceding error log, nodes in the cluster cannot communicate with each other and the cluster cannot select the active node. A possible cause is that the security group selected for the cluster does not enable the port 9300.

NOTE

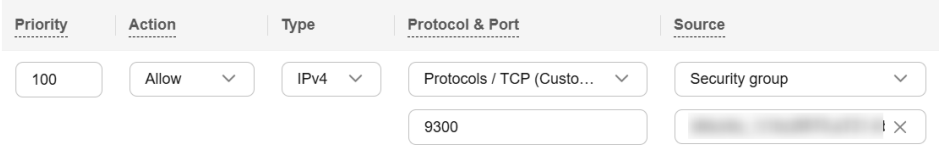
If the Elasticsearch cluster version is 7.6.2 or later, port 9300 is enabled on the subnet of user VPC by default. The security group selected for the cluster must enable the port 9300 in the subnet to ensure communication between nodes.

Procedure

1. On the **Clusters** page, click the name of the unavailable cluster. The cluster information page is displayed.
2. Click the **Overview** tab.
3. In the **Network Information** area, click the security group name. The security group details page is displayed.
4. On the **Inbound Rules** and **Outbound Rules** tabs, check whether there is a security group rule whose **Action** is **Allow**, **Protocol & Port** is **TCP:9300**, and **Type** is **IPv4**.
   - If yes, contact technical support to locate the problem.
   - If no, go to the next step.

5. Modify the cluster's security group to allow port 9300.
   a. Click the **Inbound Rules** tab.
   b. Click **Add Rule**. In the **Add Inbound Rule** dialog box, set **Priority** to **100**, **Action** to **Allow**, **Protocol & Port** to **Protocols/TCP (Custom)** and **9300**, **Type** to **IPv4**, and **Source** to the name of the current security group.

      Figure 2-5 Adding a security group rule

   c. Click **OK** to enable port 9300.
   d. Repeat the preceding steps to enable the port 9300 on the **Outbound Rules** tab page.
6. After the port 9300 is enabled for the security group, wait until the cluster becomes available.

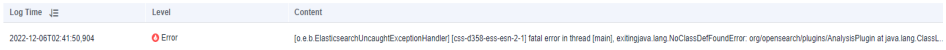
2.5 A Cluster is Unavailable Due to Plugin Incompatibility

Symptom

After a custom plugin is installed and the cluster is restarted, the cluster status changes to **Unavailable**.

Click the cluster name to go to the cluster details page, and choose **Logs** > **Log Search**. The following plug-in error message is displayed: "fatal error in thread [main], exitingjava.lang. NoClassDefFoundError: xxx/xxx/.../xxxPlugin at ...".

Figure 2-6 Node error logs



CAUTION

CSS has discontinued the custom plugin function. Clusters of earlier versions that have installed custom plugins may become unavailable.

Possible Causes

An installed custom plugin is incompatible with the current CSS cluster version. As a result, the Elasticsearch process cannot be started properly.

Procedure

1. On the **Clusters** page, click the name of the unavailable cluster. The cluster information page is displayed.
2. Choose **Cluster Settings** > **Plugins**.
3. Click the **Custom Plugins** tab, find the custom plugin, and determine whether to continue to it.
   - If yes, delete the plugin and reinstall it.
     i. In the custom plugin operation list, uninstall and delete the target plugin.
     ii. Rectify the plugin fault based on the error information in the node logs. If the fault persists, contact technical support.
     iii. After the plugin fault is rectified, upload and install the target plugin in the custom plugin operation list. If the plugin status is **Installed and to be effective upon cluster restart**, the plugin is successfully installed.
   - If no, delete the plugin.

     In the custom plugin operation list, uninstall and delete the target plugin.
4. In the upper right corner, click **Restart**.
5. After the cluster is restarted, the cluster becomes available.

2.6 A Cluster is Unavailable Due to Improper Shard Allocation

Symptom

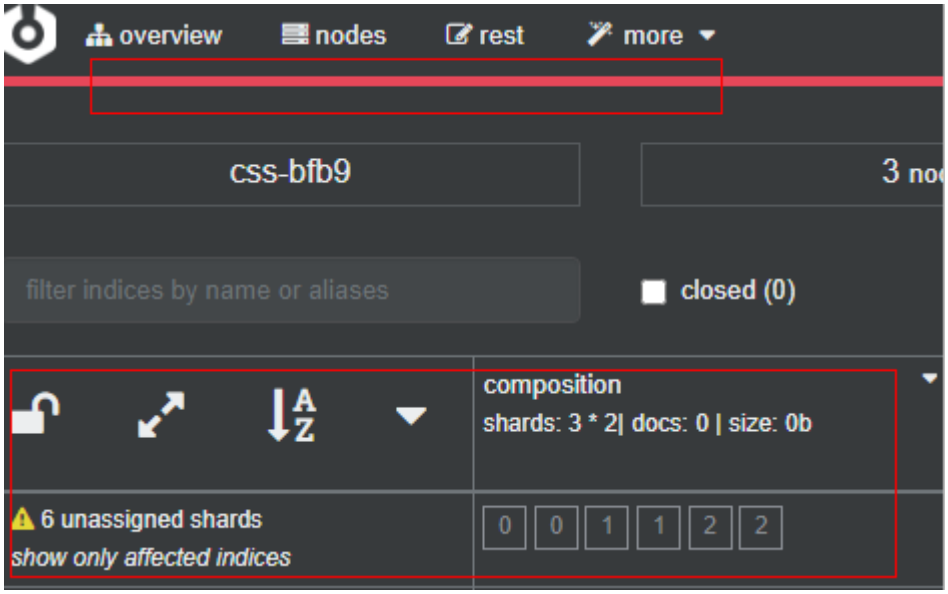
The cluster status is **Unavailable**.

On the **Dev Tools** page of Kibana, run the **GET _cluster/health** command to check the cluster health status. In the output, the value of **status** is **red** and the value of **unassigned_shards** is not **0**. Alternatively, on the **Cerebro** page, click **overview** to view the index shard allocation on each data node. If the cluster status is **red** and the value of **unassigned shards** is not **0**, there are index shards that cannot be allocated in the cluster.

Figure 2-7 Cluster health status


```
{
  "cluster_name" : "css-bfb9",
  "status" : "red",
  "timed_out" : false,
  "number_of_nodes" : 3,
  "number_of_data_nodes" : 3,
  "active_primary_shards" : 19,
  "active_shards" : 38,
  "relocating_shards" : 0,
  "initializing_shards" : 0,
  "unassigned_shards" : 6,
  "delayed_unassigned_shards" : 0,
  "number_of_pending_tasks" : 0,
  "number_of_in_flight_fetch" : 0,
  "task_max_waiting_in_queue_millis" : 0,
  "active_shards_percent_as_number" : 86.36363636363636
}
```


Figure 2-8 Cerebro page



Possible Causes

Some index shards in the cluster are not properly allocated.

Procedure

Step 1: Determine the reason why the cluster is unavailable.

1. Use Kibana to access the faulty cluster. On the **Dev Tools** page of Kibana, run the **GET /_recovery?active_only=true** command to check whether the cluster is restoring the backup.
   - If **{"index_name":{"shards":[{"id":25,"type":"...** is returned, there are indexes being restored using backup. Wait until the backup restoration is complete. If the cluster status is still **Unavailable**, go to the next step.

- If **{ }** is returned, the cluster is not performing backup restoration. Go to the next step.

2. Run the **GET _cluster/allocation/explain?pretty** command to check why some index shards are not allocated based on the returned information.

Table 2-1 Parameter description

Parameter	Description
index	Index name
shard	Shard ID
current_state	Shard status
allocate_explanation	Shard allocation explanation
explanation	Explanation

Table 2-2 Faults description

Symptom	Causes	Procedure
explanation: **no allocations are allowed due to cluster setting [cluster.routing.allocation.enable=none]**	The cluster allocation policy forbids the allocation of all shards.	For details, see **cluster.routing.allocation.enable** in **Incorrect Shard Allocation Policy Configuration**.
explanation: **too many shards [3] allocated to this node for index [write08]index setting [index.routing.allocation.total_shards_per_node=3]**	The number of shards that can be allocated to each data node from a single index in the cluster is too small, which does not meet the index shard allocation requirements.	For details, see **index.routing.allocation.total_shards_per_node** in **Incorrect Shard Allocation Policy Configuration**.
explanation: **too many shards [31] allocated to this node, cluster setting [cluster.routing.allocation.total_shards_per_node=30]**	The number of shards that can be allocated to each data node in the cluster is too small.	For details, see **cluster.routing.allocation.total_shards_per_node** in **Incorrect Shard Allocation Policy Configuration**.

Symptom	Causes	Procedure
explanation: **node does not match index setting [index.routing.allocation.include] filters [box_type:"hot"]**	Index shards can only be allocated to data nodes labeled with **hot**. If a cluster has no nodes labeled with **hot**, shards cannot be allocated.	For details, see **index.routing.allocation.include** in **Incorrect Shard Allocation Policy Configuration**.
explanation: **node does not match index setting [index.routing.allocation.require] filters [box_type:"xl"]**	Index shards can only be allocated to data nodes with specified labels. If a cluster has no such nodes, shards cannot be allocated.	For details, see **index.routing.allocation.require** in **Incorrect Shard Allocation Policy Configuration**.
explanation: **[failed to obtain in-memory shard lock]**	Generally, this problem occurs when a node is removed from a cluster for a short time and then added back to the cluster. In addition, a thread is performing a long-term data writing to a shard, such as bulk or scroll. When the node is added to the cluster again, the master node cannot allocate the shard because the shard lock is not released.	For details, see **shard lock error**.
explanation: **node does not match index setting [index.routing.allocation.include] filters [_tier_preference:"data_hot OR data_warm OR data_cold"]**	The configuration of an index does not match the cluster version.	For details, see **Inconsistent index parameter version**.
explanation: **cannot allocate because all found copies of the shard are either stale or corrupt**	The data on index shards is damaged.	For details, see **Damaged primary shard data**.

Symptom	Causes	Procedure
explanation: the node is above the high watermark cluster setting [cluster.routing.allocation.disk.watermark.high=90%], using more disk space than the maximum allowed [90.0%], actual free: [6.976380997419324%]	The node disk usage reaches the upper limit.	For details, see **Excessive disk usage**.

Step 2: Rectify the fault.

- **Incorrect shard allocation policy**
  - **cluster.routing.allocation.enable**
    i. If the value of **explanation** in the output is as follows, the current allocation policy of the cluster forbids the allocation of all shards.

    Figure 2-9 Incorrect configuration of **allocation.enable**

```
"deciders" : [
  {
    "decider" : "enable",
    "decision" : "NO",
    "explanation" : "no allocations are allowed due to cluster setting
      [cluster.routing.allocation.enable=none]"
  }
```

    ii. On the **Dev Tools** page of Kibana, run the following command to set **enable** to **all** to allow all shards to be allocated:

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

    NOTE

    The index-level configuration overwrites the cluster-level configuration. The parameters are described as follows:

    - **all**: Default value. All types of shards can be allocated.
    - **primaries**: Only the primary shards can be allocated.
    - **new_primaries**: Only the primary shards of the newly created index can be allocated.
    - **none**: No shards can be allocated.

    iii. Run the **POST _cluster/reroute?retry_failed=true** command to manually allocate shards. Wait until all index shards are allocated and the cluster status changes to **Available**.

- **index.routing.allocation.total_shards_per_node**

  i. If the value of **explanation** in the output is as follows, the value of **index.routing.allocation.total_shards_per_node** is too small and does not meet the index shard allocation requirements.

  Figure 2-10 Incorrect configuration of **index total_shards_per_node**

  ```
  "deciders" : [
    {
      "decider" : "shards_limit",
      "decision" : "NO",
      "explanation" : "too many shards [3] allocated to this node for index [write08],
        index setting [index.routing.allocation.total_shards_per_node=3]"
    }
  ]
  ```

  ii. On the **Dev Tools** page of Kibana, run the following command to change the number of index shards that can be allocated to each node:

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

  NOTE

  Value of **index.routing.allocation.total_shards_per_node** = Number of **index_name** index shards/(Number of data nodes - 1)

  Set this parameter to a relative large value. Assume that a cluster has 10 nodes, including five data nodes, two client nodes, and three master nodes. The number of shards of an index is 30. If **total_shards_per_node** is set to **4**, the total number of shards that can be allocated is: 4 x 5 = 20. Not all shards cannot be allocated. To allocate all shards in this index, at least six shards should be allocated to each data node (30 shards in total). In case a data node is faulty, at least eight shards should be allocated to each node.

  iii. Run the **POST _cluster/reroute?retry_failed=true** command to manually allocate shards. Wait until the index shards are allocated and the cluster status changes to **Available**.

- **cluster.routing.allocation.total_shards_per_node**

  i. If the value of **explanation** in the output is as follows, the number of shards that can be allocated to each data node in the cluster is too small.

  Figure 2-11 Incorrect configuration of **cluster total_shards_per_node**

  ```
  "deciders" : [
    {
      "decider" : "shards_limit",
      "decision" : "NO",
      "explanation" : "too many shards [31] allocated to this node, cluster setting
        [cluster.routing.allocation.total_shards_per_node=30]"
    }
  ```

  ii. The value of **cluster.routing.allocation.total_shards_per_node** indicates the maximum number of shards that can be allocated to

each data node in a cluster. The default value of this parameter is **1000**. On the **Dev Tools** page of Kibana, run the following command to specify the **cluster.routing.allocation.total_shards_per_node** parameter:

```
PUT _cluster/settings
{
  "persistent": {
    "cluster": {
      "routing": {
        "allocation.total_shards_per_node": 1000
      }
    }
  }
}
```

iii. In most cases, the problem occurs because **index.routing.allocation.total_shards_per_node** is mistakenly set to **cluster.routing.allocation.total_shards_per_node**. Run the following command to specify the **index.routing.allocation.total_shards_per_node** parameter:

```
PUT index_name/_settings
{
  "index": {
    "routing": {
      "allocation.total_shards_per_node": 30
    }
  }
}
```

NOTE

Both of the following parameters are used to limit the maximum number of shards that can be allocated to a single data node:

- **cluster.routing.allocation.total_shards_per_node** is used to limit shard allocation at the cluster level.
- **index.routing.allocation.total_shards_per_node** is used to limit shard allocation at the index level.

iv. Run the **POST _cluster/reroute?retry_failed=true** command to manually allocate shards. Wait until the index shards are allocated and the cluster status changes to **Available**.

– **index.routing.allocation.include**

i. If the value of **explanation** in the output is as follows, index shards can only be allocated to data nodes with the **hot** label. If no data nodes in the cluster are labeled with **hot**, shards cannot be allocated.

Figure 2-12 Incorrect configuration of **include**

```
"deciders" : [
  {
    "decider" : "filter",
    "decision" : "NO",
    "explanation" : """node does not match index setting [index.routing.allocation.include]
      filters [box_type:"hot"]"""
  }
]
```

ii. On the **Dev Tools** page of Kibana, run the following command to cancel the configuration:

```
PUT index_name/_settings
{
  "index.routing.allocation.include.box_type": null
}
```


iii. Run the **POST _cluster/reroute?retry_failed=true** command to manually allocate shards. Wait until the index shards are allocated and the cluster status changes to **Available**.

– **index.routing.allocation.require**

i. If the value of **explanation** in the output is as follows, shards can only be allocated to data nodes with specified labels. If no nodes in the cluster have such labels, the shards cannot be allocated.

Figure 2-13 Incorrect configuration of **require**

```
"deciders" : [
  {
    "decider" : "filter",
    "decision" : "NO",
    "explanation" : """node does not match index setting [index.routing.allocation.require]
      filters [box_type:"xl"]"""
  },
```

ii. On the **Dev Tools** page of Kibana, run the following command to cancel the configuration:

```
PUT index_name/_settings
{
  "index.routing.allocation.require.box_type": null
}
```

iii. Run the **POST _cluster/reroute?retry_failed=true** command to manually allocate shards. Wait until the index shards are allocated and the cluster status changes to **Available**.

- **Shard lock error**

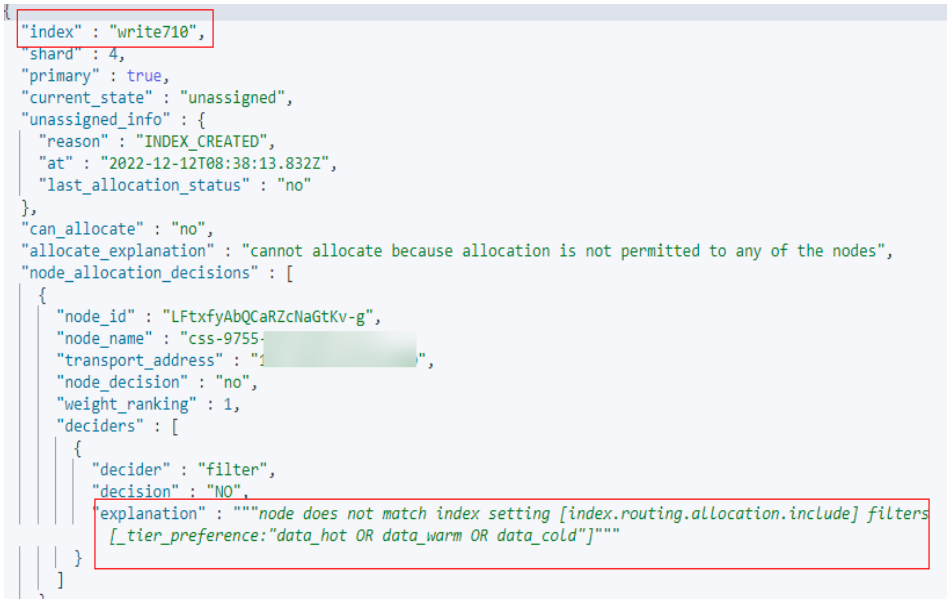
a. In the output, **explanation** contains **[failed to obtain in-memory shard lock]**. This problem usually occurs when a node is removed from a cluster for a short time and then added back to the cluster, and a thread is performing a long-term data writing to a shard, such as bulk or scroll. When the node is added to the cluster again, the master node cannot allocate the shard because the shard lock is not released.

b. This problem does not cause fragment data loss. You only need to allocate the shard again. On the **Dev Tools** page of Kibana, run the **POST /_cluster/reroute?retry_failed=true** command to manually allocate the unallocated shard. Wait until the index shards are allocated and the cluster status changes to **Available**.

- **Inconsistent index setting and node version**

a. The value of **index** and **explanation** in the output are as follows, indicating that the parameter configuration of an index does not match the node version.

Figure 2-14 Inconsistent index configuration



b. Run the **GET index_name/_settings** command to check the index configuration. In the output, check whether the index features match the node version.

Figure 2-15 Index configuration

For example, assume that a cluster version is **7.9.3**. The index feature **index.routing.allocation.include._tier_preference** is supported by clusters of the version later than **7.10**. If you use this feature in a cluster of the version earlier than **7.10**, index shards cannot be allocated. As a result, the cluster is unavailable.

c. Determine whether the inapplicable feature is mandatory for the cluster.

- If yes, create a cluster of the required version and restore the data of the old cluster to the new cluster using the backup.
- If no, go to the next step.

d. Run the following command to remove the inapplicable index feature:

```
PUT /index_name/_settings
{
  "index.routing.allocation.include._tier_preference": null
}
```

e. Run the **POST /_cluster/reroute?retry_failed=true** command to manually allocate the unallocated shards. Wait until the index shards are allocated and the cluster status changes to **Available**.

- **Damaged primary shard data**

a. The values of **index**, **shard**, **allocate_explanation**, and **store_exception** in the output are as follows, indicating that the data of a shard in an index is damaged.

Figure 2-16 Damaged primary shard data

b. When the index data is damaged or the primary backup of a shard is lost, run the following command to define an empty shard and specify the node to be allocated:

```
POST /_cluster/reroute
{
   "commands" : [
         {
         "allocate_empty_primary" : {
            "index" : "index_name",
            "shard" : 2,
            "node" : "node_name",
            "accept_data_loss":true
         }
      }
   ]
}
```

NOTICE

Data in the corresponding shard will be completely cleared. Exercise caution when performing this operation.

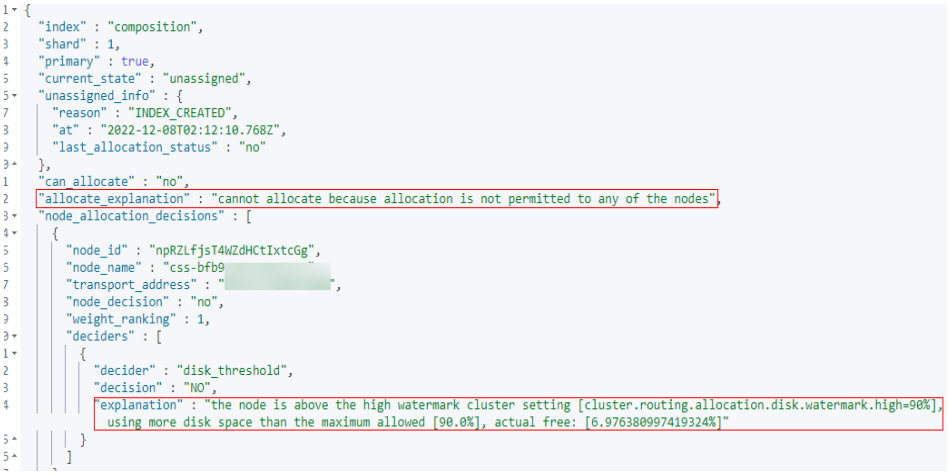
c. After index shards are reallocated, the cluster status becomes **Available**.

- **Excessive disk usage**

a. The output is as follows. The value of **allocate_explanation** indicates that shards of an index cannot be allocated to any data node, and the

value of **explanation** indicates that node disk usage reaches the upper limit.

Figure 2-17 Query result



NOTE

- If the disk usage of a node exceeds 85%, new shards will not be allocated to this node.
- If the disk usage of a node exceeds 90%, the cluster attempts to migrate the shards of this node to other data nodes with low disk usage. If data cannot be migrated, the system forcibly sets the **read_only_allow_delete** attribute for each index in the cluster. In this case, data cannot be written to indexes, and the indexes can only be read or deleted.
- A node may be disconnected due to high disk usage. After the node automatically recovers, once the cluster is overloaded, the cluster may fail to respond when you call the Elasticsearch API to query the cluster status. If the cluster status cannot be updated in a timely manner, the cluster becomes **Unavailable**.

b. Increase the available disk capacity of the cluster.

- On the **Dev Tools** page of Kibana, run the **DELETE index_name** command to clear invalid data in the cluster to release disk space.
- Temporarily reduce the number of index copies. After the disk or node capacity is expanded, change the number of index copies back to the original value.

  1) On the **Dev Tools** page of Kibana, run the following command to temporarily reduce the number of index copies:

  ```
  PUT index_name/_settings
  {
    "number_of_replicas": 1
  }
  ```

  The output is as follows.

  Figure 2-18 Index status of **read-only-allow-delete**

  ```
  {
    "type" : "cluster_block_exception",
    "reason" : "index [log07] blocked by: [TOO_MANY_REQUESTS/12/disk usage
       exceeded flood-stage watermark, index has read-only-allow-delete block];"
  }
  ```


The disk usage exceeds the maximum value allowed by the disk space. The system forcibly sets the **read_only_allow_delete** attribute for all indexes in the cluster. Run the following command to set the attribute value to **null**, and then run the command in **b.1)** to reduce the number of index copies:

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

2) Increase the number of nodes or node storage capacity of the cluster by referring to **Scaling Out a Cluster**.
3) After the scale-out is complete, run the command in step **b.1)** to change the number of index copies back. After all index shards are allocated, the cluster status changes to **Available**.

2.7 A Cluster is Unavailable Due to Incompatible Data Types

Symptom

After a cluster is restored using backup or migrated, the cluster status changes to **Unavailable**.

Possible Causes

Some types of the restored data are not supported by the destination cluster. For example, some plugins or settings are installed in the old cluster, which are not supported by the new cluster. As a result, index shards cannot be allocated.

Procedure

1. On the **Dev Tools** page of Kibana, run the **GET _cluster/allocation/explain?pretty** command to check the reason why index shards are not allocated.
2. In the returned result, the **index** and **explanation** values are **primary shard for this replica is not yet active**, indicating that the primary shard corresponding to this replica is not yet active.

Figure 2-19 Unallocated index shards

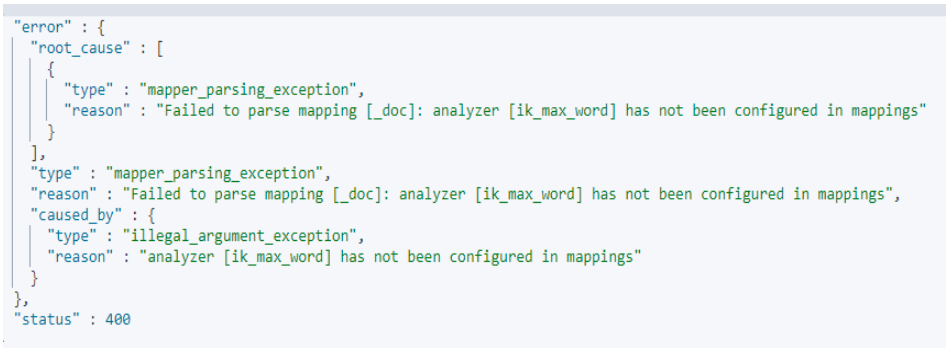
```
{
  "index" : "write24",
  "shard" : 3,
  "primary" : false,
  "current_state" : "unassigned",
  "unassigned_info" : {
    "reason" : "NEW_INDEX_RESTORED",
    "at" : "2022-12-12T07:12:58.652Z",
    "details" : "restore_source[restore_repo_auto/snapshot-8033]",
    "last_allocation_status" : "no_attempt"
  },
  "can_allocate" : "no",
  "allocate_explanation" : "cannot allocate because allocation is not permitted to any of the nodes",
  "node_allocation_decisions" : [
    {
      "node_id" : "1xQ9pmVqSPqVTTlIRAB-wA",
      "node_name" : "css-bfb
      "transport_address" :
      "node_decision" : "no",
      "deciders" : [
        {
          "decider" : "replica_after_primary_active",
          "decision" : "NO",
          "explanation" : "primary shard for this replica is not yet active"
```


3. Modify the configuration of the index and run the following command to set the number of copies to **0**:

```
PUT /index_name/_settings
{
  "number_of_replicas": 0
}
```

The value of **reason** indicates that some types of the data that is being restored are not supported by the CSS cluster.

Figure 2-20 Incompatible data



```
"error" : {
  "root_cause" : [
    {
      "type" : "mapper_parsing_exception",
      "reason" : "Failed to parse mapping [_doc]: analyzer [ik_max_word] has not been configured in mappings"
    }
  ],
  "type" : "mapper_parsing_exception",
  "reason" : "Failed to parse mapping [_doc]: analyzer [ik_max_word] has not been configured in mappings",
  "caused_by" : {
    "type" : "illegal_argument_exception",
    "reason" : "analyzer [ik_max_word] has not been configured in mappings"
  }
},
"status" : 400
```

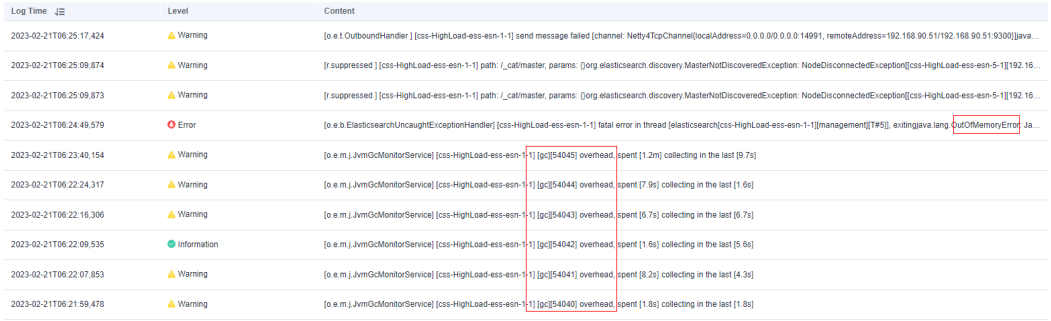

4. According to the root cause, delete the data type that is not supported by the current CSS cluster or select a CSS cluster version that supports the data type, and then restore the backup or migrate the data.

2.8 A Cluster is Unavailable Due to Heavy Load

Symptom

The cluster status is **Unavailable**. Click the cluster name to go to the cluster details page, and choose **Logs** > **Log Search**. The error message "OutOfMemoryError" and alarm "[gc][xxxxx] overhead spent [x.xs] collecting in the last [x.xs]" are displayed.

Figure 2-21 Out-of-memory caused by frequent garbage collection



Log Time	Level	Content
2023-02-21T06:25:17,424	Warning	[o.e.t.OutboundHandler] [css-HighLoad-ess-esn-1-1] send message failed [channel: Netty4TcpChannel{localAddress=0.0.0.0/0.0.0.0:14991, remoteAddress=192.168.90.51/192.168.90.51:9300}]java...
2023-02-21T06:25:09,874	Warning	[r.suppressed] [css-HighLoad-ess-esn-1-1] path: /_cat/master, params: {}org.elasticsearch.discovery.MasterNotDiscoveredException: NodeDisconnectedException[[css-HighLoad-ess-esn-5-1][192.16...
2023-02-21T06:25:09,873	Warning	[r.suppressed] [css-HighLoad-ess-esn-1-1] path: /_cat/master, params: {}org.elasticsearch.discovery.MasterNotDiscoveredException: NodeDisconnectedException[[css-HighLoad-ess-esn-5-1][192.16...
2023-02-21T06:24:49,579	Error	[o.e.b.ElasticsearchUncaughtExceptionHandler] [css-HighLoad-ess-esn-1-1] fatal error in thread [elasticsearch[css-HighLoad-ess-esn-1-1][management][T#5]], exiting java.lang.OutOfMemoryError: Ja...
2023-02-21T06:23:40,154	Warning	[o.e.m.j.JvmGcMonitorService] [css-HighLoad-ess-esn-1-1] [gc][54045] overhead, spent [1.2m] collecting in the last [9.7s]
2023-02-21T06:22:24,317	Warning	[o.e.m.j.JvmGcMonitorService] [css-HighLoad-ess-esn-1-1] [gc][54044] overhead, spent [7.9s] collecting in the last [1.6s]
2023-02-21T06:22:16,306	Warning	[o.e.m.j.JvmGcMonitorService] [css-HighLoad-ess-esn-1-1] [gc][54043] overhead, spent [6.7s] collecting in the last [6.7s]
2023-02-21T06:22:09,535	Information	[o.e.m.j.JvmGcMonitorService] [css-HighLoad-ess-esn-1-1] [gc][54042] overhead, spent [1.6s] collecting in the last [5.6s]
2023-02-21T06:22:07,853	Warning	[o.e.m.j.JvmGcMonitorService] [css-HighLoad-ess-esn-1-1] [gc][54041] overhead, spent [8.2s] collecting in the last [4.3s]
2023-02-21T06:21:59,478	Warning	[o.e.m.j.JvmGcMonitorService] [css-HighLoad-ess-esn-1-1] [gc][54040] overhead, spent [1.8s] collecting in the last [1.8s]

Possible Causes

The cluster is overloaded due to too many pending queries and writes. When the heap memory is insufficient, tasks cannot be allocated and garbage collection is frequently triggered. As a result, the Elasticsearch process exits abnormally.

Procedure

NOTE

If a cluster is overloaded for a long time, data writes and queries may be slow. You are advised to upgrade the node specifications, add new nodes, or scale out node capacities. For details about how to upgrade the node specifications, increase the number of nodes, or expand the storage capacity of nodes, see **Scaling Out a Cluster**.

1. Check whether tasks are stacked in the cluster.
   - Method 1: On the **Dev Tools** page of Kibana, run the following commands to check whether tasks are being delayed:

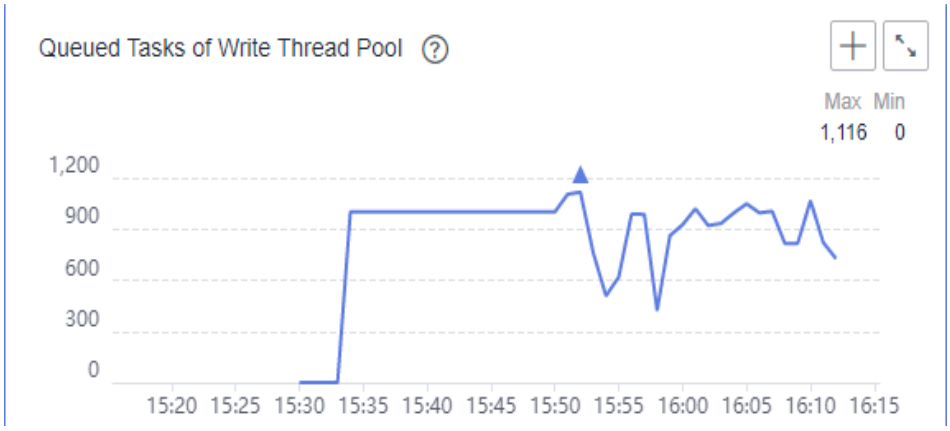
     ```
     GET /_cat/thread_pool/write?v
     GET /_cat/thread_pool/search?v
     ```

     If the value of **queue** is not 0, tasks are stacked.

     ```
     node_name              name  active queue rejected
     css-0323-ess-esn-2-1   write      2   200     7662
     css-0323-ess-esn-1-1   write      2   188     7660
     css-0323-ess-esn-5-1   write      2   200     7350
     css-0323-ess-esn-3-1   write      2   196     8000
     css-0323-ess-esn-4-1   write      2   189     7753
     ```

   - Method 2: In the cluster list, find the target cluster, and click **More** > **Monitoring Metrics** in the **Operation** column. On the displayed page, check the total number of queued tasks in the search thread pool and write thread pool. If the number of queued tasks is not 0, tasks are being delayed.

Figure 2-22 Queued Tasks of White Thread Pool

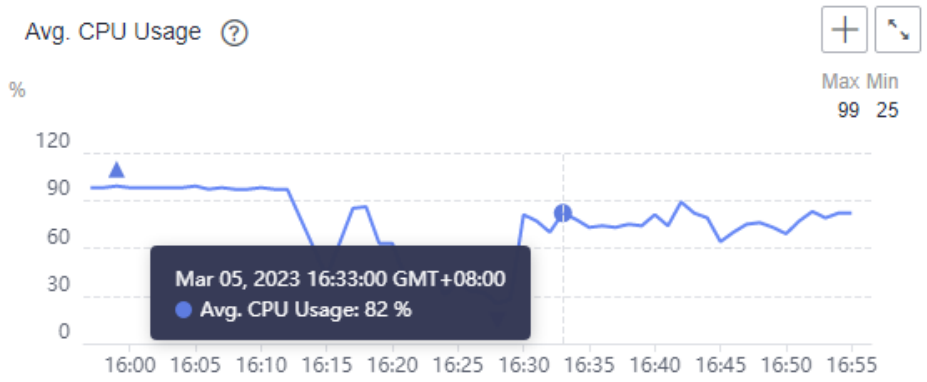


- If a large number of tasks are stacked in the cluster, perform the following steps to optimize the cluster:
  - Choose **Logs** > **Log Search**. If a large number of slow query logs has been generated before the node is out of memory, optimize the queries based on site requirements.
  - Choose **Logs** > **Log Search**. If the error message "Inflight circuit break" or "segment can't keep up" can be found, the circuit breaker may have been triggered due to heavy write pressure. If the amount of written data (write rate) increases sharply recently, you need to properly arrange the write peak time window based on service requirements.
- If no task is stacked in the cluster or the cluster is still unavailable after optimization, go to the next step to check whether the cluster is overloaded.

2. Check whether the cluster is overloaded.

   In the cluster list, find the target cluster, and click **More** > **Monitoring Metrics** in the **Operation** column. On the displayed page, check metrics related to CPU and heap memory usage, such as average CPU usage and average JVM heap usage. If the average CPU usage exceeds 80% or the average JVM heap usage exceeds 70%, the cluster is under heavy pressure.

Figure 2-23 Avg. CPU Usage



- If the cluster is overloaded, reduce the client request sending rate or expand the cluster capacity.
- If the cluster pressure is not overloaded or the cluster is still unavailable after the request sending rate is reduced, go to the next step to check whether a large amount of cache exists in the cluster.

3. On the **Dev Tools** page of Kibana, run the following command to check whether the cluster's cache has cached a large number of requests:
```
GET /_cat/nodes?v&h=name,queryCacheMemory,fielddataMemory,requestCacheMemory
```
   - If the value of **queryCacheMemory**, **fielddataMemory**, or **requestCacheMemory** in the output exceeds 20% of the heap memory, run the **POST _cache/clear** command to clear the cache. The cached data is generated during data query to speed up the query. After the cached data is cleared, the query latency may increase.
```
name                 queryCacheMemory fielddataMemory requestCacheMemory
css-0323-ess-esn-1-1             200mb          1.6gb             200mb
```
     NOTE

     You can run the following command to query the maximum heap memory of each node:
```
GET _cat/nodes?v&h=name,ip,heapMax
```
     name indicates the node name and *ip* indicates the IP address of the node.
   - If the cluster is still overloaded after the optimization, contact technical support.

2.9 Client Node Overload

Symptom

A client node overload in an Elasticsearch/OpenSearch cluster can cause the following issues:

- Kibana (or OpenSearch Dashboards) and Cerebro become inaccessible.
- Write and query latency increases significantly.
- Monitoring data shows high CPU, memory, or JVM usage on the client nodes, and some nodes may disconnect from the cluster.

Possible Causes

- Unbalanced load: Requests are unevenly distributed to the client nodes.
- Abnormal or malicious traffic: Sudden traffic spikes from malicious IP addresses or due to abnormal services (such as crawlers and unthrottled write scripts) are overwhelming the client nodes.
- Heavy query or write load: Large quantities of result sets from different shards may need to be merged for complex queries, leading to CPU bottlenecks. (This can be common in the case of unoptimized aggregation queries.) Or the client nodes may be overwhelmed in the face of sustained high-concurrency writes.
- Exhaustion of JVM resources: Continuously high JVM memory usage triggers frequent garbage collection (GC).

Solutions

Scenario 1: Unbalanced load (some of the nodes are overloaded)

- Solution 1: Modify the connection settings by including all client node IP addresses to ensure balanced load across all the nodes.
- Solution 2: Use ELB to access your CSS cluster and improve the cluster's availability and performance through load balancing.

Scenario 2: Abnormal traffic or heavy load (CPU overload)

- Solution 1: Disallow or throttle IP addresses with abnormal traffic.
  a. Identify IP addresses with abnormal traffic.

     On the cluster's Intelligent O&M page, select **Intelligent Diagnostics**, and check **Client Connection Check (Source IP Address Analysis)** to identify abnormal IP addresses.

  b. Forbid or throttle these IP addresses.

     The following is a command for your reference (supported only in Elasticsearch 7.6.2, Elasticsearch 7.10.2, OpenSearch 2.19.0, and OpenSearch 3.4.0):

```
PUT /_cluster/settings
{
  "persistent": {
    "flowcontrol.http.enabled": true,
    "flowcontrol.http.deny": "192.168.1.100,192.168.1.101" // Replace it with the IP address generating abnormal traffic.
  }
}
```

- Solution 2: If it is not practical to disallow or throttle IP addresses with abnormal traffic, temporarily close problematic indexes instead.

  CAUTION

  Closing an index will make it completely inaccessible for both read and write operations.

  a. Identify indexes responsible for abnormal resource usage.

     Run the following command, and in the command output, look for records in which **action** is **indices:data/read/search**, and identify indexes whose **running_time** is abnormal:

```
GET _cat/tasks?v&detailed=true
```

     In the example output shown below, index_20251210 is showing abnormal resource usage.

```
action                          task_id                    parent_task_id              type      start_time    timestamp running_time ip              node                description
cluster:monitor/tasks/lists     16oEhzTLSxOmpOAKcDWiRA:4405923 -                          transport 1765329559256 01:37:39  84micros     192.168.77.164  css-test-ess-esn-3-1
indices:data/read/search        GbveasILSxOutqupHepB4Q:1511246 -                          transport 1765329590043 01:19:50  1.7h         192.168.127.133 css-test-ess-client-esn-1-1
indices[index_20251210], types[], search_type[QUERY_THEN_FETCH], source[{"size":10000,"query":{"ids":{"values":["99998-786522498053054743"],"boost":1.0}}}]
```

  b. Temporarily close the abnormal index.

```
POST /index_20251210/_close
```


- Solution 3: Add more client nodes or upgrade their specifications. If the ELB service is used, restart it for the node changes to take effect.

Scenario 3: Exhaustion of JVM resources (JVM overload)

1. Restart client nodes that are unresponsive.
2. Wait for 5 minutes before checking JVM usage.
3. If the load remains high, handle it by using the same methods used for the CPU overload situation (disallowing or throttling abnormal IP addresses, increasing capacity, etc).

2.10 Master Node Overload

Symptom

A master node overload in an Elasticsearch/OpenSearch cluster can cause the following issues:

- The cluster status changes to yellow or red, and metadata operations cannot be performed.
- Multiple nodes are disconnected from the cluster.
- **master node failed** and **restarting discovery** are generated in the log.
- When APIs are called, a message is displayed indicating that the master node does not exist.

Possible Causes

- Too many pending tasks: Pending tasks can accumulate, for example, during service changes that generate a large number of PUT mapping tasks, or due to task priority conflicts where urgent tasks block routine tasks like index creation.
- Too many shards: An excessively large number of shards can cause high memory and CPU usage on master nodes and lead to increased task processing delays.
- Metadata processing overload: Frequent metadata operations (such as index creation and shard allocation) or expired cluster state updates can cause the master nodes' CPU usage to stay above 90%, affecting cluster stability.

Solutions

Scenario 1: Too many pending tasks

- Solution to accumulation of pending tasks:
  a. Restart the cluster (all nodes) to release pending tasks and thereby restore the cluster.
  b. Temporarily block index writes and mapping updates to prevent further accumulation of pending tasks.

```
PUT my_index/_settings
{
  "index.blocks.metadata": true,
  "index.blocks.write": true,
```



```
  "index.blocks.read_only": true
}
```

c. After the cluster recovers, restore index writes and mapping updates to restore service availability.

```
PUT my_index/_settings
{
  "index.blocks.metadata": false,
  "index.blocks.write": false,
  "index.blocks.read_only": false
}
```

d. Verify the cluster status. If the cluster status is green, the fault has been rectified.

```
GET _cluster/health?pretty
```

- Solution to task priority conflicts:

  a. Identify the cause of index creation failures. For example, check the index template configuration, check whether the number of shards has exceeded the cluster capacity, and check whether the cluster has sufficient storage space.

  b. Rectify configuration issues. For example, modify index template parameters (such as the number of shards and replicas), reclaim storage space, and modify shard allocation policies.

  c. Resubmit index creation requests to verify fault rectification.

Scenario 2: Too many shards

- Solution 1: Upgrade master node specifications so they can handle more shards.

  Recommended heap memory-to-shards ratio: 200 shards per GB of heap memory. For example, if there are 4,000 shards, there should be at least 20 GB of heap memory.

- Solution 2: Optimize shard policies to reduce the pressure of the master nodes in metadata processing.

  For example, merge small shards (using the _shrink API), adjust the number of shards in the index template, and increase the number of replicas to dilute the load.

Scenario 3: Metadata processing overload

- Solution 1: Upgrade master node specifications to increase their capacity and prevent CPU overload.

  Increase the number of vCPUs to at least 8, and increase the memory capacity to at least 32 GB.

- Solution 2: Reduce the frequency and complexity of metadata processing.

  For example, avoid frequent metadata changes, merge batch operations (such as batch index creation), and optimize shard allocation policies.

2.11 Node Overload During Cluster Upgrade or Node Specifications Change

Symptom

During a cluster upgrade (for example, a rolling upgrade) or node specifications change, an inappropriate recovery setting may cause the load of some or all of the nodes in the cluster to increase drastically. Typical symptoms include:

- High node load: During a cluster upgrade or node specifications change, the CPU and JVM memory usage of the cluster nodes increases drastically.
- Node disconnection: Some nodes are removed from the cluster due to resource exhaustion, and the cluster status changes to red.
- Task interruption: The execution time of the upgrade or node specifications change task increases significantly, and some tasks expire and fail.
- Resource bottlenecks: The monitoring data shows the CPU usage of some nodes exceeds 90% and the JVM memory usage keeps increasing.

Possible Causes

- Inappropriate recovery settings: The indices.recovery.max_bytes_per_sec setting is too high. This leads to high data migration speed, causing high CPU and memory usage and node overload.
- Too many pending tasks: During a cluster upgrade or node specifications change, automatically triggered shard migration and replica reallocation tasks may be delayed due to inappropriate parallelism settings, further increasing resource consumption.
- Insufficient resources: The cluster nodes have limited capacities, disk I/O, or network bandwidth and are unable to support high concurrency, leading to low task execution efficiency.

Solutions

1. Adjust recovery settings to reduce the communication overhead between nodes and improve cluster stability.

   Reduce the data migration speed to reduce the node load and ensure cluster stability. The downside is that the upgrade or node specifications change may take longer.

   ```
   PUT _cluster/settings
   {
     "transient": {
       "indices.recovery.max_bytes_per_sec": "40mb",
       "cluster.routing.allocation.node_concurrent_outgoing_recoveries": 2,
       "cluster.routing.allocation.cluster_concurrent_rebalance": 2,
       "cluster.routing.allocation.node_initial_primaries_recoveries": 2
     }
   }
   ```

   Wait for 5 to 10 minutes. Then check metrics like CPU usage and JVM memory usage to see if the load has dropped.

   – Yes: Check whether the cluster status is normal and that all nodes are healthy. If they are, the issue is fixed.

  - No: Further reduce indices.recovery.max_bytes_per_sec. If the load is still high or the task has expired or failed, go to the next step.
2. Terminate the task. If it is an upgrade task, you can terminate it on the CSS console. However, you can only do that if none of the cluster nodes has been successfully upgraded. If the task cannot be terminated, contact technical support.
3. Add more nodes or upgrade the node specifications to ensure there are sufficient resources to meet current service requirements.

   After the capacity expansion is complete, wait for 10 to 15 minutes and make sure the new nodes have started functioning.
4. After the capacity expansion is successful, run the upgrade or specifications change task again.
   - If the task succeeds, the issue is fixed.
   - Otherwise, contact technical support.

2.12 Elasticsearch Cluster Overload Due to Excessive Logstash Resource Consumption During Data Migration

Symptom

During Elasticsearch data migration, Logstash pipelines can consume excessive CPU and memory resources during the ingestion and output phases. This contention for resources, often due to suboptimal Logstash configuration or aggressive migration throughput targets, can degrade performance or cause outages in the source or destination Elasticsearch clusters. Typical symptoms include:

- In the source cluster: increased query response times, more queued queries, delayed indexing operations, overloaded cluster nodes, and cluster status in red.
- Destination cluster: rejected write requests, cluster status in red, and interrupted data migration.

Possible Causes

- High Logstash resource consumption: During data migration, Logstash operates its input, filter, and output stages concurrently in a single data pipeline. This could lead to excessive CPU and memory usage in the source and destination clusters if two key Logstash parameters are misconfigured: batchSize (the number of events processed per batch) and pipeline.workers (the number of parallel threads).
- Elasticsearch cluster overload: The data migration overloads the clusters due to unmanaged resource contention: the source cluster is overwhelmed by high-concurrency read requests (search/scans) from the Logstash pipeline, while the destination cluster is overwhelmed by a write throughput that exceeds its indexing capacity.
- Inappropriate migration policy: The migration uses a high-throughput policy that does not adapt to real-time cluster load. Instead of migrating data in

controlled, sequential batches to smooth resource consumption, it pushes data too fast and in too large volumes, overwhelming the processing capacity of both clusters.

Solutions

1. Stop Logstash processes to release resources in the source and destination clusters.
   a. Go to the **Configuration Center** page.
      i. Log in to the **CSS management console**.
      ii. In the navigation pane on the left, choose **Clusters > Logstash**.
      iii. In the cluster list, click the name of the target cluster. The cluster information page is displayed.
      iv. Click the **Configuration Center** tab.
   b. Select the target pipeline and click **Hot Stop** above the pipeline list.
   c. In the displayed dialog box, click **OK**.

      If the hot stop is successful, the task is removed from the pipeline list and data migration is stopped.
2. Restore the source and destination clusters.

   If the node load is too high, restart the high-low nodes or wait for the clusters to recover by themselves. Wait for 5 to 10 minutes before checking the cluster status again.
3. Adjust the Logstash configuration to reduce resource contention.
   a. Select the target Logstash cluster, and enter its **Configuration Center** page.
   b. In the configuration file list, locate the target configuration file, and click **Edit** in the **Operation** column.

      Modify **Configuration File Content** to reduce the migration task from batch indexes to a single index or same-type indexes, and migrate data in batches based on workload priorities.
   c. Click **Next** to modify runtime parameters.

      Reduce the value of **pipeline.workers** to reduce concurrent requests. The default value is the number of vCPUs. Set this parameter based on available resources.

      Reduce the value of **pipeline.batch.size** to reduce the amount of data processed for each batch. The default value is 125. You are advised to set this parameter to 50 to smooth the load.

      Increase the value of **pipeline.batch.delay** to introduce a longer wait time to fill each batch, thus reducing sudden resource spikes. The default value is 50. You can set it to 500 ms.
   d. Click **Save**.
4. Restart Logstash processes.
   a. In the configuration file list, select the target configuration file and click **Start Logstash**.
   b. In the **Start Logstash** dialog box, select **Keepalive** if necessary.

c. Click **OK** to start the configuration file.

You can check the started configuration file in the pipeline list.

5. Check whether the data migration is successful by comparing the number of documents between the source and destination clusters.
   - If the migration is successful, this issue is fixed.
   - Otherwise, contact technical support.

2.13 Logstash Write Failures

Symptom

Logstash is a data processing pipeline that ingests data from designated sources (such as Kafka or a file system), transforms the data, and writes the results to destinations like Elasticsearch or OpenSearch. Failures during the write phase can occur for various reasons, with typical symptoms including:

- HTTP status codes 403 and 429, "out of disk" errors, or out-of-memory (OOM) exceptions are recorded in Logstash logs.
- The Logstash pipeline fails to start.
- The destination cluster contains frozen indexes, full disks, or nodes that are unavailable.

When Logstash write failures occur, data availability is affected for downstream applications.

Possible Causes

- Destination cluster issues: abnormal index status (for example, frozen or creation failure); insufficient cluster resources (for example, the number of shards has reached the upper limit, or insufficient disk space); authentication failures or account lockout; loss of network connectivity; or abnormal node status (red or yellow).
- Logstash issues: incorrect settings (for example, a large batch.size that leads to OOM); insufficient disk space (/opt/logstash/log is full); resource exhaustion (contention for CPU/memory/I/O resources); or pipeline startup or health check failures.
- Issues in the data source or an intermediate link: unstable data source (for example, Kafka exception); incorrect data formats; or invalid output caused by a filter plugin failure.

Solutions

1. Check whether the Logstash pipeline is started successfully.
   a. Log in to the **CSS management console**.
   b. In the navigation pane on the left, choose **Clusters > Logstash**.
   c. In the cluster list, click the name of the target cluster. The cluster information page is displayed.
   d. Click the **Configuration Center** tab.

On the **Configuration Center** page, check whether the Logstash pipeline is **Running**.

- Yes: Go to the next step.
- No: There might be a network connectivity issue. Handle it by referring to **Scenario 1: Loss of network connectivity**.

2. Check the run logs of the Logstash configuration file.

   On the **Configuration Center** page, click **Run Logs**. Check the log file to analyze error information. Pay attention to error messages containing **elasticsearch**, **action**, **pipeline**, **cluster_block_exception**, **disk**, and **out**.

   For example:

   - **retrying failed action with response code: 403 ... index preparing to close.** : This indicates a frozen index. Handle it by referring to **Scenario 3: Frozen indexes**.
   - **Blocked user <username>**: The password is incorrect or the account is locked out. Handle it by referring to **Scenario 5: Authentication failure or account lockout**.
   - **max retries** or **cluster block**: The disk is full or the index is read-only. Handle it by referring to **Scenario 4: Read-only indexes (full disks)**.
   - **Java Heap space** or **Out of Memory**: Logstash is out of memory. Handle it by referring to **Scenario 6: Logstash node OOM**.

3. Check the status of the destination cluster and determine whether the destination cluster is responsible for the failure.

   Commonly used commands for troubleshooting:

```
GET /_cluster/health?pretty   //Check the cluster's health status (Green/Yellow/Red).
GET /_cat/indices?v or Kibana Stack Monitoring   //Check whether the destination index exists, and check its status, health, and disk usage.
GET /_nodes/stats?human&filter_path=**.fs.*   // Check the disk usage of the cluster nodes.
GET /_cluster/settings   //Check the cluster.max_shards_per_node setting.
```

Solutions for different issues:

- **Scenario 1: Loss of network connectivity**

  Configure Logstash cluster routes: Add routes that contain the source and destination clusters' private IP addresses or CIDR blocks for the Logstash cluster to ensure that Logstash can connect to these clusters.

- **Scenario 2: The number of existing shards has reached the upper limit in the destination cluster, and new indexes cannot be created.**

  Increase the value of **cluster.max_shards_per_node** in the destination cluster.

  CAUTION

  This is a cluster-level setting that remains effective until it is overwritten or cleared. Ensure that the new value is within the range supported by the hardware capacity of all nodes.

```
PUT /_cluster/settings
{
  "persistent": {
    "cluster.max_shards_per_node": "< New greater value > "
  }
}
```


After the change is made, wait for Logstash to try to create new indexes again.

- **Scenario 3: Frozen indexes**

  Delete the frozen indexes. Logstash will automatically try to recreate the indexes and continue writing data into them.

  ```
  DELETE /<frozen_index_name>
  ```

- **Scenario 4: Read-only indexes (full disks)**

  a. Identify nodes with full disks.

     On the cluster's **Intelligent O&M** page, check the diagnostic item **Data Node Disk Usage Check** to identify nodes with full disks.

  b. Expand the disk capacity: Switch to larger disks for these nodes.

  c. Clear the disks: Delete obsolete snapshots and indexes, or manually delete files.

  d. After the disk space is reclaimed, the index status will be restored to green.

  e. The indexes are no longer read-only. Wait for Logstash to write data to them again.

- **Scenario 5: Authentication failure or account lockout**

  For security-mode clusters, change the cluster accounts and passwords in the Logstash configuration file to ensure that Logstash can access these clusters, and then restart the Logstash configuration file. If relevant accounts have been locked out, contact technical support to unlock them.

- **Scenario 6: Logstash node OOM**

  Method 1: Expand the Logstash cluster. Add more nodes to the Logstash cluster to rebalance the write load and prevent OOM on individual nodes.

  Method 2: Modify the Logstash configuration file and tune relevant runtime parameters. For example, reduce the value of **pipeline.batch.size**. Reducing **pipeline.batch.size** can reduce the amount of data written per batch, thereby reducing the peak memory usage. The downside is that this may increase the needed write batches and reduce the throughput.

2.14 Insufficient Disk Space

Symptom

Insufficient disk space occurs when one or more disk partitions of an Elasticsearch/OpenSearch cluster approach or exceed a predefined usage threshold. This condition can degrade performance, cause service malfunctions, or lead to a complete outage for workloads dependent on those partitions. Typical symptoms include:

- The disk usage of cluster data nodes (including cold data nodes) exceeds 90%. Nodes exceeding this threshold cannot allocate new shards. If all nodes exceed the 90% threshold, the cluster is unable to allocate new shards, and creating an index will cause the cluster status to turn red.
- The disk usage of cluster data nodes (including cold data nodes) exceeds 95%. When the disk usage of any node hosting a shard exceeds 95%, the index enters read-only protection mode and cannot accept data writes.

- A large volume of data merging or excessively fast writes cause the index to fail to enter read-only protection mode in time, resulting in 100% disk usage. This leads to process failures. 100% disk usage is a critical issue. In this case, expand the disk capacity.

Possible Causes

- Uneven shard distribution: Index shards are not evenly distributed across the cluster nodes, causing significantly higher disk usage on some nodes compared to others.
- Insufficient initial disk capacity: The storage capacity originally allocated to the cluster nodes is inadequate for the actual data volumes or workloads.

Solutions

1. Check the number of shards on each node to determine whether they are balanced across all nodes.

   ```
   GET _cat/shards?v
   ```

   If there is unbalanced shard distribution, perform the following steps to fix the issue:

   a. Use the _cluster/reroute API to manually migrate shards to balance the load.

   ```
   POST _cluster/reroute
   {
     "commands": [
       {
         "move": {
           "index": "index_name",    //Destination index name
           "shard": 0,        //Destination shard ID
           "from_node": "node1",    //Source node ID
           "to_node": "node2" //Destination node ID, indicating a node with relatively low load
         }
       }
     ]
   }
   ```

   b. Reduce the number of replicas to lower storage overhead.

   ```
   PUT /index_name/_settings
   {
     "number_of_replicas": 1
   }
   ```

2. If shards are balanced across the cluster nodes, try to reclaim storage by deleting obsolete data.

   Delete obsolete indexes.

   ```
   DELETE {index_name}
   ```

3. If these steps do not work, expand the cluster's storage capacity.

3 Data Import and Export

3.1 Logs Cannot Be Written In Due to High CPU Usage of Elasticsearch

Symptom

The CPU usage of an Elasticsearch cluster is high, an error message **Elasticsearch Unreachable** is reported by Logstash, and logs cannot be written to Elasticsearch.

Possible Causes

The indexes have only one shard. This could easily overload individual nodes. When the job queue is full, later jobs are rejected.

Procedure

1. Log in to the **CSS management console**.
2. In the navigation pane on the left, choose **Clusters > Elasticsearch**.
3. In the cluster list, find the target cluster, and choose **More** > **Cerebro** in the **Operation** column. Log in to Cerebro.
4. In Cerebro, check the number of shards in the cluster and metrics such as the CPU, load, heap, and disk of each node.
5. Analyze the possible causes based on metrics and tune your system accordingly.
   a. Increase the number of queues and reduce rejected jobs by changing the value of **write.queue_size**.
      i. Click the name of the target cluster. The cluster information page is displayed.
      ii. Choose **Cluster Settings** > **Parameter Settings** > **Elasticsearch**.
      iii. Expand **Custom**, find the **write.queue_size** parameter and increase its value.

      If this parameter does not exist, add it under **Custom**.

For more information, see **Parameter Settings**.

b. Rebuild the indexes to ensure that the number of shards is greater than that of nodes in the cluster.

6. If the number of shards and queues are appropriate but the CPU usage and load are still high, you are advised to scale out the cluster.

3.2 An Error Is Returned When Logstash Deployed on an ECS Pushes Data to CSS

Symptom

After Logstash is deployed on an ECS, an error is reported when data is pushed to CSS. The error message is as follows:

```
LogStash::Outputs::ElasticSearch::HttpClient::Pool::BadResponseCodeError: Got response code '500' contacting Elasticsearch at URL 'https://192.168.xx.xx:9200/_xpack'
```

Possible Causes

CSS has not integrated the x-pack plugin. When you connect to CSS after deploying Logstash, the system will check whether x-pack is enabled for CSS.

Procedure

1. Delete the x-pack directory in Logstash.
2. Add **ilm_enabled => false** for Elasticsearch in the output module of the Logstash configuration file.
3. Push data to the Elasticsearch cluster again.

3.3 "Could not write all entries" Is Reported When I Use ES-Hadoop to Import Data

Issue Analysis

The Elasticsearch backend bulk thread pool queue supports a maximum of 200 requests. Requests exceeding this limit will be rejected.

Solution

1. Set a proper number of the concurrent write requests from the client as required. ES-Hadoop has a retry mechanism for the rejected HTTP requests. You can modify the following parameters:
   - **es.batch.write.retry.count**: The default value for retry times is **3**.
   - **es.batch.write.retry.wait**: The waiting time for each attempt is 10s.
2. If you do not require real-time query, you can adjust the shard refresh time (once per second by default) to improve the write speed.

```
PUT /my_logs
{
"settings": {
```



```
"refresh_interval": "30s"
}
}
```

3.4 OpenSearch Dashboards Sample Data Import Failure

Symptom

When importing sample data on the OpenSearch Dashboards page of an OpenSearch 2.11.0 or OpenSearch 2.17.1 cluster, the installation fails with the following error message:

```
Unable to install sample data set: Sample flight data
```

NOTE

Sample data is an example dataset provided by OpenSearch Dashboards for quickly experiencing query and visualization features. Import failure does not affect existing business data in the cluster.

Possible Causes

In OpenSearch Dashboards versions 2.11.0 and 2.17.1, certain fields are missing from the **.kibana** index and index templates, resulting in incomplete data writes when importing sample flight data.

Solution

Execute commands on the OpenSearch Dashboards Dev Tools page to repair the **.kibana** index field mapping.

Step 1 Log in to OpenSearch Dashboards and navigate to the command execution page.

1. Log in to the **CSS management console**.
2. In the navigation pane on the left, choose **Clusters > OpenSearch**.
3. In the cluster list, find the target cluster, and click **Dashboards** in the **Operation** column to log in to OpenSearch Dashboards.
4. In the left navigation pane, choose **Dev Tools**.

   The left part of the console is the command input box, and the triangle icon in its upper-right corner is the execution button. The right part shows the execution result.

Step 2 Run the following commands sequentially to repair the field mapping of the **.kibana** index and its associated index templates, ensuring that newly created **.kibana** indexes also contain the complete set of fields.

CAUTION

The following commands will modify the field mappings of the **.kibana** index and index templates. Verify the command content before execution, and wait for the returned result to include **"acknowledged": true** before executing the next command. You are advised to perform this operation during off-peak hours.

- Command 1: Repair the **.kibana** index field mapping.

```
PUT /.kibana*/_mapping
{
  "_meta": {
    "migrationMappingPropertyHashes": {
      "augment-vis": "88f930f6b43c089a12bfc276c359d1eb",
      "application_usage_daily": "43b8830d5d0df85a6823d290885fc9fd",
      "visualization": "f819cf6636b75c9e76ba733a0c6ef355",
      "observability-search": "638ae3ab28e5faf2e02f00dd8091abaf",
      "references": "7997cf5a56cc02bdc9c93361bde732b0",
      "integration-template": "9033ba0b281fe1c88c3c56d8fd9d089d",
      "type": "2f4316de49999235636386fe51dc06c1",
      "sample-data-telemetry": "7d3cfeb915303c9641c59681967ffeb4",
      "search": "43012c7ebc4cb57054e0a490e4b43023",
      "originId": "2f4316de49999235636386fe51dc06c1",
      "application_usage_totals": "3d1b76c39bfb2cc8296b024d73854724",
      "updated_at": "00da57df13e94e9d98437d13ace4bfe0",
      "dql-telemetry": "d12a98a6f19a2d273696597547e064ee",
      "search-telemetry": "3d1b76c39bfb2cc8296b024d73854724",
      "integration-instance": "17ab914d1f92a5b03bbb194651c23b1c",
      "observability-panel": "df07b1a361c32daf4e6842c1d5521dbe",
      "visualization-visbuilder": "7a6fadcdaaf97b2b8b65d290fd8a3ba7",
      "map": "32fcfaba6c580bf048359fcb48ad5301",
      "dashboard": "40554caf09725935e2c02e02563a2d07",
      "query": "531cf50c8e0b1c9f610e263f878d124c",
      "ui-metric": "0d409297dc5ebe1e3a1da691c6ee32e3",
      "application_usage_transactional": "3d1b76c39bfb2cc8296b024d73854724",
      "observability-visualization": "638ae3ab28e5faf2e02f00dd8091abaf",
      "url": "c7f66a0df8b1b52f17c28c4adb111105",
      "migrationVersion": "4a1746014a75ade3a714e1db5763276f",
      "index-pattern": "45915a1ad866812242df474eb0479052",
      "namespace": "2f4316de49999235636386fe51dc06c1",
      "observability-notebook": "638ae3ab28e5faf2e02f00dd8091abaf",
      "config": "c63748b75f39d0c54de12d12c1ccbc20",
      "tsvb-validation-telemetry": "3a37ef6c8700ae6fc97d5c7da00e9215",
      "namespaces": "2f4316de49999235636386fe51dc06c1",
      "homepage": "a8d79cfe4f79546aa5bbee73facc51dc"
    }
  },
  "dynamic": "strict",
  "properties": {
    "augment-vis": {
      "properties": {
        "visLayerExpressionFn": {
          "properties": {
            "args": { "dynamic": "true", "type": "object" },
            "name": { "type": "text" },
            "type": { "type": "text" }
          }
        },
        "pluginResource": {
          "properties": {
            "id": { "type": "text" },
            "type": { "type": "text" }
          }
        },
        "originPlugin": { "type": "text" },
        "description": { "type": "text" },
        "title": { "type": "text" },
        "version": { "type": "integer" },
        "visName": { "type": "keyword" }
      }
    },
    "application_usage_daily": {
      "dynamic": "false",
      "properties": {
        "timestamp": { "type": "date" }
      }
    },
```

```
"visualization": {
  "properties": {
    "savedSearchRefName": { "type": "keyword" },
    "description": { "type": "text" },
    "uiStateJSON": { "index": true, "type": "text" },
    "title": { "type": "text" },
    "version": { "type": "integer" },
    "kibanaSavedObjectMeta": {
      "properties": {
        "searchSourceJSON": { "index": true, "type": "text" }
      }
    },
    "visState": { "type": "text" }
  }
},
"observability-search": {
  "dynamic": "false",
  "properties": {
    "description": { "type": "text" },
    "title": { "type": "text" },
    "version": { "type": "integer" }
  }
},
"references": {
  "type": "nested",
  "properties": {
    "name": { "type": "keyword" },
    "id": { "type": "keyword" },
    "type": { "type": "keyword" }
  }
},
"integration-template": {
  "dynamic": "false",
  "properties": {
    "components": { "type": "nested" },
    "author": { "type": "text" },
    "displayName": { "type": "text" },
    "description": { "type": "text" },
    "type": { "type": "text" },
    "version": { "type": "text" },
    "statics": { "type": "nested" },
    "labels": { "type": "text" },
    "sourceUrl": { "type": "text" },
    "license": { "type": "text" },
    "assets": { "type": "nested" },
    "sampleData": { "type": "nested" },
    "name": { "type": "text" }
  }
},
"type": { "type": "keyword" },
"sample-data-telemetry": {
  "properties": {
    "installCount": { "type": "long" },
    "unInstallCount": { "type": "long" }
  }
},
"search": {
  "properties": {
    "hits": { "index": false, "type": "integer", "doc_values": true },
    "columns": { "index": false, "type": "keyword", "doc_values": true },
    "description": { "type": "text" },
    "sort": { "index": false, "type": "keyword", "doc_values": true },
    "title": { "type": "text" },
    "version": { "type": "integer" },
    "kibanaSavedObjectMeta": {
      "properties": {
        "searchSourceJSON": { "index": false, "type": "text" }
      }
    }
```



```
    }
  },
  "application_usage_totals": { "dynamic": "false", "type": "object" },
  "originId": { "type": "keyword" },
  "dql-telemetry": {
    "properties": {
      "optInCount": { "type": "long" },
      "optOutCount": { "type": "long" }
    }
  },
  "updated_at": { "type": "date" },
  "search-telemetry": { "dynamic": "false", "type": "object" },
  "integration-instance": {
    "dynamic": "false",
    "properties": {
      "assets": { "type": "nested" },
      "templateName": { "type": "text" },
      "name": { "type": "text" },
      "creationDate": { "type": "date" },
      "dataSource": { "type": "text" }
    }
  },
  "observability-panel": {
    "dynamic": "false",
    "properties": {
      "description": { "type": "text" },
      "title": { "type": "text" }
    }
  },
  "visualization-visbuilder": {
    "properties": {
      "styleState": { "index": false, "type": "text" },
      "uiState": { "index": false, "type": "text" },
      "description": { "type": "text" },
      "visualizationState": { "index": false, "type": "text" },
      "title": { "type": "text" },
      "version": { "type": "integer" },
      "kibanaSavedObjectMeta": {
        "properties": {
          "searchSourceJSON": { "index": true, "type": "text" }
        }
      }
    }
  },
  "map": {
    "properties": {
      "uiState": { "index": false, "type": "text" },
      "mapState": { "index": false, "type": "text" },
      "description": { "type": "text" },
      "layerList": { "index": false, "type": "text" },
      "title": { "type": "text" },
      "version": { "type": "integer" },
      "kibanaSavedObjectMeta": {
        "properties": {
          "searchSourceJSON": { "index": false, "type": "text" }
        }
      }
    }
  },
  "dashboard": {
    "properties": {
      "hits": { "index": true, "type": "integer", "doc_values": true },
      "timeFrom": { "index": true, "type": "keyword", "doc_values": true },
      "timeTo": { "index": true, "type": "keyword", "doc_values": true },
      "refreshInterval": {
        "properties": {
          "display": { "index": true, "type": "keyword", "doc_values": true },
          "section": { "index": true, "type": "integer", "doc_values": true },
          "value": { "index": true, "type": "integer", "doc_values": true },
```



```
        "pause": { "index": true, "type": "boolean", "doc_values": true }
      }
    },
    "description": { "type": "text" },
    "timeRestore": { "index": true, "type": "boolean", "doc_values": true },
    "title": { "type": "text" },
    "version": { "type": "integer" },
    "kibanaSavedObjectMeta": {
      "properties": {
        "searchSourceJSON": { "index": true, "type": "text" }
      }
    },
    "optionsJSON": { "index": true, "type": "text" },
    "panelsJSON": { "index": true, "type": "text" }
  }
},
"query": {
  "properties": {
    "timefilter": { "type": "object", "enabled": false },
    "isTemplate": { "type": "boolean" },
    "query": {
      "properties": {
        "query": { "index": false, "type": "keyword" },
        "language": { "type": "keyword" },
        "dataset": {
          "properties": {
            "timeFieldName": { "type": "text" },
            "language": { "type": "text" },
            "id": { "type": "text" },
            "title": { "type": "text" },
            "type": { "type": "text" },
            "dataSource": {
              "properties": {
                "meta": {
                  "properties": {
                    "name": { "type": "text" },
                    "sessionId": { "type": "text" }
                  }
                },
                "id": { "type": "text" },
                "title": { "type": "text" },
                "type": { "type": "text" }
              }
            }
          }
        }
      }
    },
    "description": { "type": "text" },
    "filters": { "type": "object", "enabled": false },
    "title": { "type": "text" }
  }
},
"application_usage_transactional": { "dynamic": "false", "type": "object" },
"ui-metric": {
  "properties": {
    "count": { "type": "integer" }
  }
},
"observability-visualization": {
  "dynamic": "false",
  "properties": {
    "description": { "type": "text" },
    "title": { "type": "text" },
    "version": { "type": "integer" }
  }
},
"url": {
  "properties": {
```



```
      "accessCount": { "type": "long" },
      "accessDate": { "type": "date" },
      "url": { "type": "text", "fields": { "keyword": { "ignore_above": 2048, "type": "keyword" } } },
      "createDate": { "type": "date" }
    }
  },
  "migrationVersion": {
    "dynamic": "true",
    "properties": {
      "config": { "type": "text", "fields": { "keyword": { "ignore_above": 256, "type": "keyword" } } }
    }
  },
  "index-pattern": {
    "dynamic": "false",
    "properties": {
      "title": { "type": "text" },
      "type": { "type": "keyword" }
    }
  },
  "namespace": { "type": "keyword" },
  "observability-notebook": {
    "dynamic": "false",
    "properties": {
      "description": { "type": "text" },
      "title": { "type": "text" },
      "version": { "type": "integer" }
    }
  },
  "config": {
    "dynamic": "false",
    "properties": {
      "buildNum": { "type": "keyword" }
    }
  },
  "tsvb-validation-telemetry": {
    "properties": {
      "failedRequests": { "type": "long" }
    }
  },
  "homepage": {
    "properties": {
      "heroes": {
        "properties": {
          "id": { "type": "keyword" }
        }
      },
      "kibanaSavedObjectMeta": {
        "properties": {
          "searchSourceJSON": { "index": false, "type": "text" }
        }
      },
      "sections": {
        "properties": {
          "id": { "type": "keyword" }
        }
      }
    }
  },
  "namespaces": { "type": "keyword" }
 }
}
```

- Command 2: Repair the **tenant_template** index template.

```
PUT /_template/tenant_template
{
 "order": 0,
 "index_patterns": [
  ".kibana*",
  ".kibana_-*_*",
  ".kibana_0*_*",
```

```
  ".kibana_1*_*",
  ".kibana_2*_*",
  ".kibana_3*_*",
  ".kibana_4*_*",
  ".kibana_5*_*",
  ".kibana_6*_*",
  ".kibana_7*_*",
  ".kibana_8*_*",
  ".kibana_9*_*"
 ],
 "settings": {
  "index": {
   "number_of_shards": "1",
   "priority": "1000",
   "auto_expand_replicas": "0-1"
  }
 },
 "mappings": {
  "_meta": {
   "migrationMappingPropertyHashes": {
    "augment-vis": "88f930f6b43c089a12bfc276c359d1eb",
    "application_usage_daily": "43b8830d5d0df85a6823d290885fc9fd",
    "visualization": "f819cf6636b75c9e76ba733a0c6ef355",
    "observability-search": "638ae3ab28e5faf2e02f00dd8091abaf",
    "references": "7997cf5a56cc02bdc9c93361bde732b0",
    "integration-template": "9033ba0b281fe1c88c3c56d8fd9d089d",
    "type": "2f4316de49999235636386fe51dc06c1",
    "sample-data-telemetry": "7d3cfeb915303c9641c59681967ffeb4",
    "search": "43012c7ebc4cb57054e0a490e4b43023",
    "originId": "2f4316de49999235636386fe51dc06c1",
    "application_usage_totals": "3d1b76c39bfb2cc8296b024d73854724",
    "updated_at": "00da57df13e94e9d98437d13ace4bfe0",
    "dql-telemetry": "d12a98a6f19a2d273696597547e064ee",
    "search-telemetry": "3d1b76c39bfb2cc8296b024d73854724",
    "integration-instance": "17ab914d1f92a5b03bbb194651c23b1c",
    "observability-panel": "df07b1a361c32daf4e6842c1d5521dbe",
    "visualization-visbuilder": "7a6fadcdaaf97b2b8b65d290fd8a3ba7",
    "map": "32fcfaba6c580bf048359fcb48ad5301",
    "dashboard": "40554caf09725935e2c02e02563a2d07",
    "query": "531cf50c8e0b1c9f610e263f878d124c",
    "ui-metric": "0d409297dc5ebe1e3a1da691c6ee32e3",
    "application_usage_transactional": "3d1b76c39bfb2cc8296b024d73854724",
    "observability-visualization": "638ae3ab28e5faf2e02f00dd8091abaf",
    "url": "c7f66a0df8b1b52f17c28c4adb111105",
    "migrationVersion": "4a1746014a75ade3a714e1db5763276f",
    "index-pattern": "45915a1ad866812242df474eb0479052",
    "namespace": "2f4316de49999235636386fe51dc06c1",
    "observability-notebook": "638ae3ab28e5faf2e02f00dd8091abaf",
    "config": "c63748b75f39d0c54de12d12c1ccbc20",
    "tsvb-validation-telemetry": "3a37ef6c8700ae6fc97d5c7da00e9215",
    "namespaces": "2f4316de49999235636386fe51dc06c1",
    "homepage": "a8d79cfe4f79546aa5bbee73facc51dc"
   }
  },
  "dynamic": "strict",
  "properties": {
   "augment-vis": {
    "properties": {
     "visLayerExpressionFn": {
      "properties": {
       "args": { "dynamic": "true", "type": "object" },
       "name": { "type": "text" },
       "type": { "type": "text" }
      }
     },
     "pluginResource": {
      "properties": {
       "id": { "type": "text" },
       "type": { "type": "text" }
```

```
        }
      },
      "originPlugin": { "type": "text" },
      "description": { "type": "text" },
      "title": { "type": "text" },
      "version": { "type": "integer" },
      "visName": { "index": false, "type": "keyword", "doc_values": false }
    }
  },
  "application_usage_daily": {
    "dynamic": "false",
    "properties": {
      "timestamp": { "type": "date" }
    }
  },
  "visualization": {
    "properties": {
      "savedSearchRefName": { "index": false, "type": "keyword", "doc_values": false },
      "description": { "type": "text" },
      "uiStateJSON": { "index": false, "type": "text" },
      "title": { "type": "text" },
      "version": { "type": "integer" },
      "kibanaSavedObjectMeta": {
        "properties": {
          "searchSourceJSON": { "index": false, "type": "text" }
        }
      },
      "visState": { "index": false, "type": "text" }
    }
  },
  "observability-search": {
    "dynamic": "false",
    "properties": {
      "description": { "type": "text" },
      "title": { "type": "text" },
      "version": { "type": "integer" }
    }
  },
  "references": {
    "type": "nested",
    "properties": {
      "name": { "type": "keyword" },
      "id": { "type": "keyword" },
      "type": { "type": "keyword" }
    }
  },
  "integration-template": {
    "dynamic": "false",
    "properties": {
      "components": { "type": "nested" },
      "author": { "type": "text" },
      "displayName": { "type": "text" },
      "description": { "type": "text" },
      "type": { "type": "text" },
      "version": { "type": "text" },
      "statics": { "type": "nested" },
      "labels": { "type": "text" },
      "sourceUrl": { "type": "text" },
      "license": { "type": "text" },
      "assets": { "type": "nested" },
      "sampleData": { "type": "nested" },
      "name": { "type": "text" }
    }
  },
  "type": { "type": "keyword" },
  "sample-data-telemetry": {
    "properties": {
      "installCount": { "type": "long" },
      "unInstallCount": { "type": "long" }
```



```
    }
  },
  "search": {
    "properties": {
      "hits": { "index": false, "type": "integer", "doc_values": false },
      "columns": { "index": false, "type": "keyword", "doc_values": false },
      "description": { "type": "text" },
      "sort": { "index": false, "type": "keyword", "doc_values": false },
      "title": { "type": "text" },
      "version": { "type": "integer" },
      "kibanaSavedObjectMeta": {
        "properties": {
          "searchSourceJSON": { "index": false, "type": "text" }
        }
      }
    }
  },
  "application_usage_totals": { "dynamic": "false", "type": "object" },
  "originId": { "type": "keyword" },
  "dql-telemetry": {
    "properties": {
      "optInCount": { "type": "long" },
      "optOutCount": { "type": "long" }
    }
  },
  "updated_at": { "type": "date" },
  "search-telemetry": { "dynamic": "false", "type": "object" },
  "integration-instance": {
    "dynamic": "false",
    "properties": {
      "assets": { "type": "nested" },
      "templateName": { "type": "text" },
      "name": { "type": "text" },
      "creationDate": { "type": "date" },
      "dataSource": { "type": "text" }
    }
  },
  "observability-panel": {
    "dynamic": "false",
    "properties": {
      "description": { "type": "text" },
      "title": { "type": "text" }
    }
  },
  "visualization-visbuilder": {
    "properties": {
      "styleState": { "index": false, "type": "text" },
      "uiState": { "index": false, "type": "text" },
      "description": { "type": "text" },
      "visualizationState": { "index": false, "type": "text" },
      "title": { "type": "text" },
      "version": { "type": "integer" },
      "kibanaSavedObjectMeta": {
        "properties": {
          "searchSourceJSON": { "index": false, "type": "text" }
        }
      }
    }
  },
  "map": {
    "properties": {
      "uiState": { "index": false, "type": "text" },
      "mapState": { "index": false, "type": "text" },
      "description": { "type": "text" },
      "layerList": { "index": false, "type": "text" },
      "title": { "type": "text" },
      "version": { "type": "integer" },
      "kibanaSavedObjectMeta": {
        "properties": {
```



```
          "searchSourceJSON": { "index": false, "type": "text" }
        }
      }
    }
  },
  "dashboard": {
    "properties": {
      "hits": { "index": false, "type": "integer", "doc_values": false },
      "timeFrom": { "index": false, "type": "keyword", "doc_values": false },
      "timeTo": { "index": false, "type": "keyword", "doc_values": false },
      "refreshInterval": {
        "properties": {
          "display": { "index": false, "type": "keyword", "doc_values": false },
          "section": { "index": false, "type": "integer", "doc_values": false },
          "value": { "index": false, "type": "integer", "doc_values": false },
          "pause": { "index": false, "type": "boolean", "doc_values": false }
        }
      },
      "description": { "type": "text" },
      "timeRestore": { "index": false, "type": "boolean", "doc_values": false },
      "title": { "type": "text" },
      "version": { "type": "integer" },
      "kibanaSavedObjectMeta": {
        "properties": {
          "searchSourceJSON": { "index": false, "type": "text" }
        }
      },
      "optionsJSON": { "index": false, "type": "text" },
      "panelsJSON": { "index": false, "type": "text" }
    }
  },
  "query": {
    "properties": {
      "timefilter": { "type": "object", "enabled": false },
      "isTemplate": { "type": "boolean" },
      "query": {
        "properties": {
          "query": { "index": false, "type": "keyword" },
          "language": { "type": "keyword" },
          "dataset": {
            "properties": {
              "timeFieldName": { "type": "text" },
              "language": { "type": "text" },
              "id": { "type": "text" },
              "title": { "type": "text" },
              "type": { "type": "text" },
              "dataSource": {
                "properties": {
                  "meta": {
                    "properties": {
                      "name": { "type": "text" },
                      "sessionId": { "type": "text" }
                    }
                  },
                  "id": { "type": "text" },
                  "title": { "type": "text" },
                  "type": { "type": "text" }
                }
              }
            }
          }
        }
      },
      "description": { "type": "text" },
      "filters": { "type": "object", "enabled": false },
      "title": { "type": "text" }
    }
  },
  "application_usage_transactional": { "dynamic": "false", "type": "object" },
```



```
"ui-metric": {
  "properties": {
    "count": { "type": "integer" }
  }
},
"observability-visualization": {
  "dynamic": "false",
  "properties": {
    "description": { "type": "text" },
    "title": { "type": "text" },
    "version": { "type": "integer" }
  }
},
"url": {
  "properties": {
    "accessCount": { "type": "long" },
    "accessDate": { "type": "date" },
    "url": { "type": "text", "fields": { "keyword": { "ignore_above": 2048, "type": "keyword" } } },
    "createDate": { "type": "date" }
  }
},
"migrationVersion": {
  "dynamic": "true",
  "properties": {
    "config": { "type": "text", "fields": { "keyword": { "ignore_above": 256, "type": "keyword" } } }
  }
},
"index-pattern": {
  "dynamic": "false",
  "properties": {
    "title": { "type": "text" },
    "type": { "type": "keyword" }
  }
},
"namespace": { "type": "keyword" },
"observability-notebook": {
  "dynamic": "false",
  "properties": {
    "description": { "type": "text" },
    "title": { "type": "text" },
    "version": { "type": "integer" }
  }
},
"config": {
  "dynamic": "false",
  "properties": {
    "buildNum": { "type": "keyword" }
  }
},
"tsvb-validation-telemetry": {
  "properties": {
    "failedRequests": { "type": "long" }
  }
},
"homepage": {
  "properties": {
    "heroes": {
      "properties": {
        "id": { "type": "keyword" }
      }
    },
    "kibanaSavedObjectMeta": {
      "properties": {
        "searchSourceJSON": { "index": false, "type": "text" }
      }
    },
    "sections": {
      "properties": {
        "id": { "type": "keyword" }
```



```
          }
        }
      }
    },
    "namespaces": { "type": "keyword" }
  }
 },
 "aliases": {}
}
```

- Command 3: Repair the **kibana_template** index template.

```
PUT /_template/kibana_template
{
  "order": 0,
  "index_patterns": [
    ".kibana",
    ".kibana_*"
  ],
  "settings": {
    "index": {
      "number_of_shards": "1",
      "priority": "1000",
      "auto_expand_replicas": "0-1"
    }
  },
  "mappings": {
    "_meta": {
      "migrationMappingPropertyHashes": {
        "augment-vis": "88f930f6b43c089a12bfc276c359d1eb",
        "application_usage_daily": "43b8830d5d0df85a6823d290885fc9fd",
        "visualization": "f819cf6636b75c9e76ba733a0c6ef355",
        "observability-search": "638ae3ab28e5faf2e02f00dd8091abaf",
        "references": "7997cf5a56cc02bdc9c93361bde732b0",
        "integration-template": "9033ba0b281fe1c88c3c56d8fd9d089d",
        "type": "2f4316de49999235636386fe51dc06c1",
        "sample-data-telemetry": "7d3cfeb915303c9641c59681967ffeb4",
        "search": "43012c7ebc4cb57054e0a490e4b43023",
        "originId": "2f4316de49999235636386fe51dc06c1",
        "application_usage_totals": "3d1b76c39bfb2cc8296b024d73854724",
        "updated_at": "00da57df13e94e9d98437d13ace4bfe0",
        "dql-telemetry": "d12a98a6f19a2d273696597547e064ee",
        "search-telemetry": "3d1b76c39bfb2cc8296b024d73854724",
        "integration-instance": "17ab914d1f92a5b03bbb194651c23b1c",
        "observability-panel": "df07b1a361c32daf4e6842c1d5521dbe",
        "visualization-visbuilder": "7a6fadcdaaf97b2b8b65d290fd8a3ba7",
        "map": "32fcfaba6c580bf048359fcb48ad5301",
        "dashboard": "40554caf09725935e2c02e02563a2d07",
        "query": "531cf50c8e0b1c9f610e263f878d124c",
        "ui-metric": "0d409297dc5ebe1e3a1da691c6ee32e3",
        "application_usage_transactional": "3d1b76c39bfb2cc8296b024d73854724",
        "observability-visualization": "638ae3ab28e5faf2e02f00dd8091abaf",
        "url": "c7f66a0df8b1b52f17c28c4adb111105",
        "migrationVersion": "4a1746014a75ade3a714e1db5763276f",
        "index-pattern": "45915a1ad866812242df474eb0479052",
        "namespace": "2f4316de49999235636386fe51dc06c1",
        "observability-notebook": "638ae3ab28e5faf2e02f00dd8091abaf",
        "config": "c63748b75f39d0c54de12d12c1ccbc20",
        "tsvb-validation-telemetry": "3a37ef6c8700ae6fc97d5c7da00e9215",
        "namespaces": "2f4316de49999235636386fe51dc06c1",
        "homepage": "a8d79cfe4f79546aa5bbee73facc51dc"
      }
    },
    "dynamic": "strict",
    "properties": {
      "augment-vis": {
        "properties": {
          "visLayerExpressionFn": {
            "properties": {
              "args": { "dynamic": "true", "type": "object" },
              "name": { "type": "text" },
```

```
          "type": { "type": "text" }
        }
      },
      "pluginResource": {
        "properties": {
          "id": { "type": "text" },
          "type": { "type": "text" }
        }
      },
      "originPlugin": { "type": "text" },
      "description": { "type": "text" },
      "title": { "type": "text" },
      "version": { "type": "integer" },
      "visName": { "index": false, "type": "keyword", "doc_values": false }
    }
  },
  "application_usage_daily": {
    "dynamic": "false",
    "properties": {
      "timestamp": { "type": "date" }
    }
  },
  "visualization": {
    "properties": {
      "savedSearchRefName": { "index": false, "type": "keyword", "doc_values": false },
      "description": { "type": "text" },
      "uiStateJSON": { "index": false, "type": "text" },
      "title": { "type": "text" },
      "version": { "type": "integer" },
      "kibanaSavedObjectMeta": {
        "properties": {
          "searchSourceJSON": { "index": false, "type": "text" }
        }
      },
      "visState": { "index": false, "type": "text" }
    }
  },
  "observability-search": {
    "dynamic": "false",
    "properties": {
      "description": { "type": "text" },
      "title": { "type": "text" },
      "version": { "type": "integer" }
    }
  },
  "references": {
    "type": "nested",
    "properties": {
      "name": { "type": "keyword" },
      "id": { "type": "keyword" },
      "type": { "type": "keyword" }
    }
  },
  "integration-template": {
    "dynamic": "false",
    "properties": {
      "components": { "type": "nested" },
      "author": { "type": "text" },
      "displayName": { "type": "text" },
      "description": { "type": "text" },
      "type": { "type": "text" },
      "version": { "type": "text" },
      "statics": { "type": "nested" },
      "labels": { "type": "text" },
      "sourceUrl": { "type": "text" },
      "license": { "type": "text" },
      "assets": { "type": "nested" },
      "sampleData": { "type": "nested" },
      "name": { "type": "text" }
```



```
    }
  },
  "type": { "type": "keyword" },
  "sample-data-telemetry": {
    "properties": {
      "installCount": { "type": "long" },
      "unInstallCount": { "type": "long" }
    }
  },
  "search": {
    "properties": {
      "hits": { "index": false, "type": "integer", "doc_values": false },
      "columns": { "index": false, "type": "keyword", "doc_values": false },
      "description": { "type": "text" },
      "sort": { "index": false, "type": "keyword", "doc_values": false },
      "title": { "type": "text" },
      "version": { "type": "integer" },
      "kibanaSavedObjectMeta": {
        "properties": {
          "searchSourceJSON": { "index": false, "type": "text" }
        }
      }
    }
  },
  "application_usage_totals": { "dynamic": "false", "type": "object" },
  "originId": { "type": "keyword" },
  "dql-telemetry": {
    "properties": {
      "optInCount": { "type": "long" },
      "optOutCount": { "type": "long" }
    }
  },
  "updated_at": { "type": "date" },
  "search-telemetry": { "dynamic": "false", "type": "object" },
  "integration-instance": {
    "dynamic": "false",
    "properties": {
      "assets": { "type": "nested" },
      "templateName": { "type": "text" },
      "name": { "type": "text" },
      "creationDate": { "type": "date" },
      "dataSource": { "type": "text" }
    }
  },
  "observability-panel": {
    "dynamic": "false",
    "properties": {
      "description": { "type": "text" },
      "title": { "type": "text" }
    }
  },
  "visualization-visbuilder": {
    "properties": {
      "styleState": { "index": false, "type": "text" },
      "uiState": { "index": false, "type": "text" },
      "description": { "type": "text" },
      "visualizationState": { "index": false, "type": "text" },
      "title": { "type": "text" },
      "version": { "type": "integer" },
      "kibanaSavedObjectMeta": {
        "properties": {
          "searchSourceJSON": { "index": false, "type": "text" }
        }
      }
    }
  },
  "map": {
    "properties": {
      "uiState": { "index": false, "type": "text" },
```

```
      "mapState": { "index": false, "type": "text" },
      "description": { "type": "text" },
      "layerList": { "index": false, "type": "text" },
      "title": { "type": "text" },
      "version": { "type": "integer" },
      "kibanaSavedObjectMeta": {
        "properties": {
          "searchSourceJSON": { "index": false, "type": "text" }
        }
      }
    }
  },
  "dashboard": {
    "properties": {
      "hits": { "index": false, "type": "integer", "doc_values": false },
      "timeFrom": { "index": false, "type": "keyword", "doc_values": false },
      "timeTo": { "index": false, "type": "keyword", "doc_values": false },
      "refreshInterval": {
        "properties": {
          "display": { "index": false, "type": "keyword", "doc_values": false },
          "section": { "index": false, "type": "integer", "doc_values": false },
          "value": { "index": false, "type": "integer", "doc_values": false },
          "pause": { "index": false, "type": "boolean", "doc_values": false }
        }
      },
      "description": { "type": "text" },
      "timeRestore": { "index": false, "type": "boolean", "doc_values": false },
      "title": { "type": "text" },
      "version": { "type": "integer" },
      "kibanaSavedObjectMeta": {
        "properties": {
          "searchSourceJSON": { "index": false, "type": "text" }
        }
      },
      "optionsJSON": { "index": false, "type": "text" },
      "panelsJSON": { "index": false, "type": "text" }
    }
  },
  "query": {
    "properties": {
      "timefilter": { "type": "object", "enabled": false },
      "isTemplate": { "type": "boolean" },
      "query": {
        "properties": {
          "query": { "index": false, "type": "keyword" },
          "language": { "type": "keyword" },
          "dataset": {
            "properties": {
              "timeFieldName": { "type": "text" },
              "language": { "type": "text" },
              "id": { "type": "text" },
              "title": { "type": "text" },
              "type": { "type": "text" },
              "dataSource": {
                "properties": {
                  "meta": {
                    "properties": {
                      "name": { "type": "text" },
                      "sessionId": { "type": "text" }
                    }
                  },
                  "id": { "type": "text" },
                  "title": { "type": "text" },
                  "type": { "type": "text" }
                }
              }
            }
          }
        }
      }
```



```
      },
      "description": { "type": "text" },
      "filters": { "type": "object", "enabled": false },
      "title": { "type": "text" }
    }
  },
  "application_usage_transactional": { "dynamic": "false", "type": "object" },
  "ui-metric": {
    "properties": {
      "count": { "type": "integer" }
    }
  },
  "observability-visualization": {
    "dynamic": "false",
    "properties": {
      "description": { "type": "text" },
      "title": { "type": "text" },
      "version": { "type": "integer" }
    }
  },
  "url": {
    "properties": {
      "accessCount": { "type": "long" },
      "accessDate": { "type": "date" },
      "url": { "type": "text", "fields": { "keyword": { "ignore_above": 2048, "type": "keyword" } } },
      "createDate": { "type": "date" }
    }
  },
  "migrationVersion": {
    "dynamic": "true",
    "properties": {
      "config": { "type": "text", "fields": { "keyword": { "ignore_above": 256, "type": "keyword" } } }
    }
  },
  "index-pattern": {
    "dynamic": "false",
    "properties": {
      "title": { "type": "text" },
      "type": { "type": "keyword" }
    }
  },
  "namespace": { "type": "keyword" },
  "observability-notebook": {
    "dynamic": "false",
    "properties": {
      "description": { "type": "text" },
      "title": { "type": "text" },
      "version": { "type": "integer" }
    }
  },
  "config": {
    "dynamic": "false",
    "properties": {
      "buildNum": { "type": "keyword" }
    }
  },
  "tsvb-validation-telemetry": {
    "properties": {
      "failedRequests": { "type": "long" }
    }
  },
  "homepage": {
    "properties": {
      "heroes": {
        "properties": {
          "id": { "type": "keyword" }
        }
      },
      "kibanaSavedObjectMeta": {
```



```
          "properties": {
            "searchSourceJSON": { "index": false, "type": "text" }
          }
        },
        "sections": {
          "properties": {
            "id": { "type": "keyword" }
          }
        }
      }
    },
    "namespaces": { "type": "keyword" }
  }
 },
 "aliases": {}
}
```

Step 3 Re-import the sample data to verify that the issue is resolved.

1. In the navigation pane on the OpenSearch Dashboards console, choose **Home**.
2. Click **Try sample data** and select **Sample flight data**.
3. Click **Add data** and confirm that the import succeeds without errors.

   If the **Add data** button is not available, first remove the failed sample data import.

----End

4 Functions

4.1 Indexes Cannot Be Backed Up

Index backup is implemented by creating cluster snapshots. If index backup fails, perform the following steps to troubleshoot this problem:

Check Whether the Account or IAM User Has the Index Backup Permissions

1. In the cluster list, click the name of the target cluster. The cluster information page is displayed.
2. Click the **Cluster Snapshots** tab.
3. Click the IAM agency name under **IAM Agency** to go to the agency details page.
4. On the **Basic Information** tab, check that **Cloud Service** is **Cloud Search Service (CSS)**.
5. On the **Permissions** tab, click a permission name. On the displayed page, check that **Policy Content** contains the following actions:

```
"obs:bucket:GetBucketLocation",
"obs:object:GetObjectVersion",
"obs:object:GetObject",
"obs:object:DeleteObject",
"obs:bucket:HeadBucket",
"obs:bucket:GetBucketStoragePolicy",
"obs:object:DeleteObjectVersion",
"obs:bucket:ListBucketVersions",
"obs:bucket:ListBucket",
"obs:object:PutObject"
```

6. If SSE-KMS encryption is enabled for the selected OBS bucket, check whether the agency has the following KMS permissions:

```
"kms:cmk:create",
"kms:dek:create",
"kms:cmk:get",
"kms:dek:decrypt",
"kms:cmk:list"
```

7. If the agency has insufficient permissions, modify its permissions.
   - If a custom agency is used, you can modify the agency to add the missing actions.

- If the default CSS agency is used, click **Modify Settings** on the **Cluster Snapshots** page, and click **Automatically Create IAM Agency** or **One-click Authorization** in the **IAM Agency** area to modify the agency.

8. If indexes still cannot be backed up after the agency permissions are modified, contact technical support.

4.2 Custom Word Dictionaries Cannot Be Configured for a Cluster

Perform the following steps to troubleshoot this problem:

Check When the Target Cluster Was Created

Step 1 Log in to the **CSS management console**.

Step 2 In the navigation pane on the left, expand **Clusters**. Select a cluster type based on the target cluster. The cluster list is displayed.

Step 3 In the cluster list, locate the target cluster for which you want to configure custom word dictionaries and check its creation time.

- If the creation time is earlier than March 10, 2018, no further action is required. The function is not working because the custom word dictionary function was unavailable at that time.
- If the creation time is later than March 10, 2018, check whether the account or IAM user used for logging in to the management console has the custom word dictionary permission. For details, see **Check Whether the Account or IAM User Has the Required Permissions**.

----End

Check Whether the Account or IAM User Has the Required Permissions

1. Check the user group that the current account or IAM user belongs to.

   For details, see **Viewing or Modifying IAM User Information** in *Identity and Access Management User Guide*.

2. Check whether the user group has been granted permissions to configure custom word dictionaries as well as access to the OBS bucket list and read permission on OBS buckets and objects.

   Permissions to configure custom word dictionaries:

   - Loading a custom dictionary: css:IKThesaurus:load
   - Querying a custom word dictionary: css:IKThesaurus:get
   - Deleting a custom dictionary: css:IKThesaurus:delete

   For details, see **Viewing or Modifying User Group Information** in *Identity and Access Management User Guide*.

   - If none of the preceding permissions has been granted to the user group, go to **3**.
   - If all these permissions have been granted to the group, contact technical support.

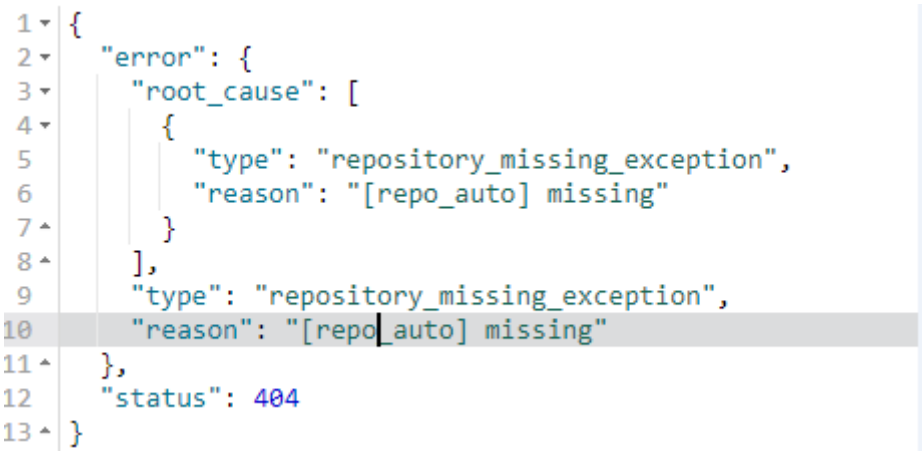
3. Add the required permissions to the user group.

   For details, see **Viewing or Modifying User Group Information** in *Identity and Access Management User Guide*.

4.3 The Snapshot Repository Cannot Be Found

1. Log in to the **CSS management console**.
2. In the navigation pane on the left, expand **Clusters**. Select a cluster type based on the target cluster. The cluster list is displayed.
3. For an Elasticsearch cluster, click **Kibana** in the **Operation** column to log in to Kibana. For an OpenSearch cluster, click **Dashboards** in the **Operation** column to log in to OpenSearch Dashboards.
4. Expand the menu in the upper-left corner, and choose **Dev Tools**.
5. If no information is returned after the **GET _snapshot/_all** is executed, or if the error message shown in **Figure 4-1** is displayed after the **GET _snapshot/repo_auto/_all** command is executed, no snapshot repository is configured. In this case, configure the snapshot again.

   Figure 4-1 Returned information



6. In the cluster list, click the name of the target cluster. On the cluster information page that is displayed, click the **Cluster Snapshots** tab.
7. Click **Modify Settings** to modify basic snapshot settings, and select an OBS bucket and backup path.
8. After the modification is complete, click **OK**.

   If the repository still cannot be found after the modification, try modifying and restoring the backup path, and save the settings again.

4.4 A Cluster Is Stuck in the Creating Snapshot State

If a cluster's task status is stuck in **Creating snapshot**, there are two possible causes:

1. **The cluster is under heavy load or contains large volumes of data, and snapshots take longer to create**

When a cluster is under heavy load (indexing or query) or contains large volumes of data, snapshots take significantly longer time to create. By default, both Elasticsearch and OpenSearch set an upper limit on the per-node snapshot speed (typically 40 MB/s). Additionally, the overall snapshot creation speed depends on the cluster resource availability. When under heavy load, the snapshot creation process takes longer.

Solution:

a. Query the snapshot progress: Use an API to check the snapshot progress and estimate the remaining time.

```
GET _snapshot/repo_auto/{snapshot_name}
```

Replace {snapshot_name} with the actual snapshot name.

In the response, check the shards_stats field (or a similar field, depending on the version) to learn the number of completed shards and the total number of shards.

b. Choose a solution.

- Wait: If the estimated remaining time is acceptable or the cluster load is expected to decrease, you can wait for the snapshot to complete.
- Terminate: If the estimated remaining time is too long, call the snapshot deletion API to terminate the task.

```
DELETE _snapshot/repo_auto/{snapshot_name}
```

2. **The snapshot status has failed to be updated**

Both Elasticsearch and OpenSearch save the status information of ongoing snapshots in a place called **cluster state**. After a snapshot is created, the task status needs to be updated in **cluster state**. If the update fails (for example, the request for updating the cluster state is rejected by the thread pool or has expired due to high memory usage in the cluster), the snapshot status in **cluster state** will be stuck in the Creating state.

Solution:

Call the snapshot deletion API to delete the snapshot task. This will clear the residual status.

```
DELETE _snapshot/repo_auto/{snapshot_name}
```

4.5 How Do I Back Up Large Volumes of Data Using Snapshots?

When you have large volumes of data and backing up the data may take more than one day, you can use the following methods to improve backup efficiency:

- Back up specified indexes in batches: Manually create snapshots on the console and specify a subset of indexes each time to reduce the amount of data backed up in a single operation.
- Customize the backup rate: Modify the snapshot configuration on the console to adjust the maximum backup rate and improve the efficiency of each backup operation.
- Use a custom snapshot repository: Create a custom repository through Kibana to flexibly configure backup rates and chunk sizes for efficient backup.

Method 1: Backing Up Specified Indexes in Batches

When the data volume is extremely large, you can manually create snapshots by specifying a subset of indexes each time to back up indexes in batches.

1. On the **Dev Tools** page of Kibana or OpenSearch Dashboards, run the following command to query the names of all indexes in the cluster:
   ```
   GET /_cat/indices
   ```
2. Divide indexes into multiple batches based on index size and priority.
3. Log in to the **CSS management console**.
4. In the navigation pane on the left, expand **Clusters**. Select a cluster type based on the target cluster. The cluster list is displayed.
5. In the cluster list, click the name of the target cluster. The cluster information page is displayed.
6. Choose the **Cluster Snapshots** tab and enable snapshots.
7. Under **Cluster Snapshot Tasks**, click **Create Snapshot Manually**. In the displayed dialog box, configure the snapshot policy.

   Table 4-1 Manually creating a snapshot

Parameter	Description
Snapshot Name	Set the snapshot name. You can use names like **snapshot-batch1** or **snapshot-batch2** to distinguish different batches.
Index	Specify the indexes to be backed up in this snapshot. If you do not specify this parameter, all indexes in the cluster are backed up by default.
Snapshot Description	Add a snapshot description.

8. Click **OK** to start creating snapshots.
9. In the cluster snapshot task list, when **Snapshot Task Status** changes to **Available**, backup for the current batch is complete.

   Click the expand icon in front of the snapshot name to view the backed-up shards and indexes and verify successful backup.
10. Repeat steps 7 to 9 until all indexes are backed up.

Method 2: Customizing the Backup Rate

You can modify the snapshot configuration on the console to adjust the maximum backup rate and improve the backup speed of each node.

CAUTION

Increasing the backup rate increases the CPU and disk I/O load on the cluster. It is recommended that you perform this operation during off-peak hours and reduce this rate before peak hours.

1. Log in to the **CSS management console**.
2. In the navigation pane on the left, expand **Clusters**. Select a cluster type based on the target cluster. The cluster list is displayed.
3. In the cluster list, click the name of the target cluster. The cluster information page is displayed.
4. Choose the **Cluster Snapshots** tab, and click **Modify Settings** to increase the maximum backup rate.

Table 4-2 Modifying the backup rate

Parameter	Description
Maximum Backup Rate (/s)	The parameter sets the maximum backup rate per node. When it is exceeded, flow control is triggered to prevent excessive resource usage and ensure system stability. **Value format**: number (0–9999) + unit (KB, MB, GB, TB, PB, or B) **Default value**: 40 MB The value 0MB means there is no rate limit. An excessively high backup rate may lead to excessive resource usage, which may impact cluster stability. Configure this parameter carefully to maintain optimal performance. The actual backup rate may not reach the configured value, as it depends on factors such as OBS performance and disk I/O.
Maximum Recovery Rate (/s)	The parameter sets the maximum recovery rate per node. When it is exceeded, flow control is triggered to prevent excessive resource usage and ensure system stability. **Value format**: number (0–9999) + unit (KB, MB, GB, TB, PB, or B) **Default value**: 40 MB for Elasticsearch 7.6.2 or earlier; and 0 MB (no rate limit) for OpenSearch and Elasticsearch versions later than 7.6.2. An excessively high recovery rate may lead to excessive resource usage, which may impact cluster stability. Configure this parameter carefully to maintain optimal performance. For OpenSearch clusters and Elasticsearch versions later than 7.6.2, the actual recovery rate is also affected by the value of **indices.recovery.max_bytes_per_sec**. Whichever is smaller between the two parameters will be used. Additionally, the actual recovery rate may not reach the configured value, as it depends on factors such as OBS performance and disk I/O.

5. Click **OK** to save the change.
6. Manually create a snapshot or wait for an automatic snapshot to be triggered. The backup will be performed based on the new rate.

Method 3: Using a Custom Snapshot Repository

In addition to the repo_auto repository provided by CSS, you can create a custom snapshot repository to flexibly configure parameters such as backup rate and chunk size for more efficient backup.

1. Log in to the **Dev Tools** page of Kibana or OpenSearch Dashboards.
2. Create a custom snapshot repository.

```
PUT _snapshot/my_backup
{
  "type": "obs",
  "settings": {
    "bucket": "<OBS bucket name>",
    "base_path": "<Backup path>",
    "chunk_size": "2g",
    "endpoint": "obs.<Region name>.example.com",
    "region": "<Region name>",
    "compress": "true",
    "access_key": "<AK>",
    "secret_key": "<SK>",
    "max_restore_bytes_per_sec": "100mb",
    "max_snapshot_bytes_per_sec": "100mb"
  }
}
```

Table 4-3 Custom snapshot repository parameters

Parameter	Mandatory	Description
bucket	Yes	OBS bucket name. The bucket must be in the same region as the cluster, and its storage class must be **Standard**.
base_path	Yes	Path for storing snapshots in the OBS bucket. The path cannot start with **/** or **..**
chunk_size	No	Chunk size for backup. The default value is **2g**. Retain the default value for large data volumes.
endpoint	Yes	OBS endpoint in the format **obs.<Region name>.example.com**.
region	Yes	Region where the cluster is located.
compress	No	Whether to enable data compression. The default value is **true**. Data compression reduces OBS storage usage.
access_key	Yes	AK for accessing OBS. For details about how to obtain the AK, see **How Do I Obtain an Access Key (AK/SK)?**

Parameter	Mandatory	Description
secret_key	Yes	SK for accessing OBS. For details about how to obtain the SK, see **How Do I Obtain an Access Key (AK/SK)?**
max_snapshot_bytes_per_sec	No	Maximum backup rate per node. You can increase this value during off-peak hours (for example, 100 MB) and reduce it during peak hours.
max_restore_bytes_per_sec	No	Maximum restore rate per node. Adjust this value based on cluster performance.

3. Create a snapshot using a custom repository.

```
PUT _snapshot/my_backup/<snapshot name>
{
  "indices": "*",
  "ignore_unavailable": true,
  "include_global_state": false
}
```

Table 4-4 Parameters for creating a snapshot

Parameter	Mandatory	Description
<snapshot name>	Yes	Snapshot name. Use names like **snapshot-batch1** to distinguish different batches.
indices	No	Specifies the indexes to back up. Separate multiple indexes with commas. Wildcards (*) are supported. The default value is *****, meaning to back up all indexes.
ignore_unavailable	No	Whether to ignore indexes that cannot be found. The default value is **false**. • **true**: Ignore indexes that cannot be found and continue backing up other indexes. • **false**: Return a failure when a target index cannot be found.
include_global_state	No	Whether to save the global cluster state. The default value is **false**. You are advised to retain the default value to prevent the global state from affecting restoration.

4. Query the snapshot status.

```
GET _snapshot/my_backup/<snapshot name>/_status
```

If the value of **state** in the returned result is **SUCCESS**, the backup is successful.

4.6 There Is a Sudden Spike in Cluster Load

Symptom

For a long time, many of a cluster's tasks are rejected and a large number of tasks are suspended. The Cerebro console shows a sudden, sharp increase in the cluster's load values.

Possible Causes

Possible causes are as follows:

- Query threads are executed slowly because a large amount of data is obtained.
- Threads are suspended caused by high read pressures.

Troubleshooting Procedure

Method 1: Using Cerebro

1. Log in to the **CSS management console**.
2. In the navigation pane on the left, expand **Clusters**. Select a cluster type based on the target cluster. The cluster list is displayed.
3. Locate the cluster whose load has spiked, and choose **More** > **Cerebro** in the **Operation** column.
4. Check the CPU and heap metrics. If the values of these two metrics are too high, the cluster is overloaded. In this case, reduce the number of requests sent by the client and wait until the cluster load decreases.
5. Check the number and size of shards. Each shard is recommended to be 20 GB to 40 GB and not exceed 50 GB. On a single node, up to five shards can use the same index.

Method 2: Using Kibana

1. Log in to the **CSS management console**.
2. In the navigation pane on the left, expand **Clusters**. Select a cluster type based on the target cluster. The cluster list is displayed.
3. For an Elasticsearch cluster, click **Kibana** in the **Operation** column to log in to Kibana. For an OpenSearch cluster, click **Dashboards** in the **Operation** column to log in to OpenSearch Dashboards.
4. Expand the menu in the upper-left corner, and choose **Dev Tools**.
5. Run the following command to check which threads are having tasks piling up and locate the cause of the sudden spike in cluster load.
   ```
   GET _cat/thread_pool?v
   ```
6. Run the following command to check which threads occupy excessive CPU resources and take a long time to execute, and locate the cause of task delaying.
   ```
   GET /_nodes/hot_threads
   ```


4.7 "I/O Reactor STOPPED" Is Reported When I Use the Elasticsearch HLRC

Symptom

When I use the HLRC (High Level Rest Client) of Elasticsearch, "I/O Reactor STOPPED" is occasionally reported, but no errors are found in Elasticsearch logs.

Figure 4-2 I/O Reactor STOPPED

```
java.lang.RuntimeException: Request cannot be executed; I/O reactor status: STOPPED
        at org.elasticsearch.client.RestClient.extractAndWrapCause(RestClient.java:796)
        at org.elasticsearch.client.RestClient.performRequest(RestClient.java:218)
        at org.elasticsearch.client.RestClient.performRequest(RestClient.java:205)
        at org.elasticsearch.client.RestHighLevelClient.internalPerformRequest(RestHighLevelClient.java:1454)
        at org.elasticsearch.client.RestHighLevelClient.performRequest(RestHighLevelClient.java:1424)
        at org.elasticsearch.client.RestHighLevelClient.performRequestAndParseEntity(RestHighLevelClient.java:1394)
        at org.elasticsearch.client.RestHighLevelClient.search(RestHighLevelClient.java:930)
        at [redacted]
        at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1149)
        at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:624)
        at java.lang.Thread.run(Thread.java:748)
Caused by: java.lang.IllegalStateException: Request cannot be executed; I/O reactor status: STOPPED
        at org.apache.http.util.Asserts.check(Asserts.java:46)
        at org.apache.http.impl.nio.client.CloseableHttpAsyncClientBase.ensureRunning(CloseableHttpAsyncClientBase.java:90)
        at org.apache.http.impl.nio.client.InternalHttpAsyncClient.execute(InternalHttpAsyncClient.java:123)
        at org.elasticsearch.client.RestClient.performRequest(RestClient.java:214)
        ... 9 common frames omitted
```

Checking Why the I/O Reactor Status Changed to STOPPED

In the call stack, the error is found in line 90 in CloseableHttpAsyncClientBase.

Figure 4-3 Error location

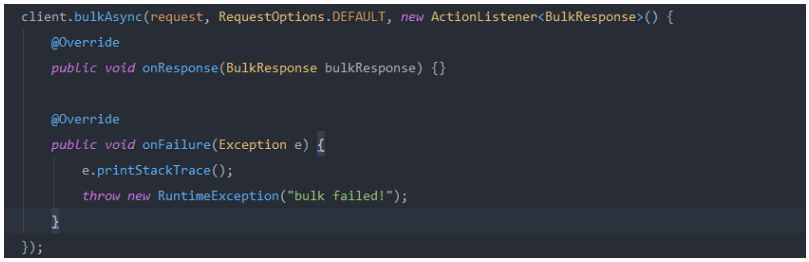
```
88      protected void ensureRunning() {
89          final Status currentStatus = this.status.get();
90          Asserts.check( expression: currentStatus == Status.ACTIVE,  message: "Request cannot be executed; " +
91                  "I/O reactor status: %s", currentStatus);
92      }
```

The **ensureRunning()** method is called at the beginning of each request execution to verify that the Client status is **ACTIVE**. If the status is not **ACTIVE**, an error is reported. Check when the **status** in **CloseableHttpAsyncClientBase** changed to **STOPPED**. Two occurrences are found, as shown in the figures below.

Figure 4-4 First occurrence of **STOPPED**

Figure 4-5 Second occurrence of **STOPPED**

- It can be inferred that only the first occurrence made the status change to **STOPPED**, because generally a user would not close the client before calling an API.
- In the first occurrence, the **reactorThread** thread scheduled schedule I/O events. The status changed to **STOPPED** when an internal exception is thrown. To reproduce this issue, throw an exception and check whether the status will change, as shown in the following figure.



- When a request fails because of the exception, the status changes to **STOPPED**, the I/O Reactor is disabled, and the HLRC instance is suspended.

Then, the HLRC instance fails at any attempt to call a request. Here the exception was manually created to reproduce the issue, but the cause of the I/O Reactor exception in the production environment is still unknown.

Possible Causes

The possible causes of the I/O Reactor exception are as follows:

1. Exception thrown during callback
2. High client concurrency

   After the exception was found in the customer's logs, we checked the Elasticsearch cluster monitoring metrics, including the CPU usage and the number of network connections.

   The customer's cluster used five i3.4xlarge.8 nodes with 16 vCPUs and 128 GB memory. The customer performed a large number of bulk operations at around 05:00 every morning and wrote 100 GB to 200 GB data. Cluster monitoring metrics showed that the CPU usage and network inbound and outbound rates did not bring heavy workloads to nodes, but the number of network connections on each node was large. The network connections on certain nodes nearly reached 9,000, and even 50,000 on five nodes. The code used multiple threads of a single REST client to concurrently invoke the bulkAsync interface of the HLRC. The 40,000 to 50,000 Elasticsearch connections on a single client node could easily exhaust handles or connections of the node.
3. Elasticsearch's REST Client. You are advised to add an Exception Handler to Elasticsearch's REST Client.

   Both HLRC and LLRC use Apache HTTPComponents Async Client. As mentioned in the documentation of Apache, some I/O exceptions during the interaction with sessions are predictable. Such exceptions may cause the termination of a single session, but do not affect I/O Reactor or other sessions. However, if an error occurs in I/O Reactor, such as the I/O exceptions in some classes of the underlying NIO or runtime exceptions that have not been not handled, I/O Reactor will shut down. Apache recommends that the IOReactorExceptionHandler interface be rewritten.

   Figure 4-6 Rewriting the IOReactorExceptionHandler API

```
DefaultConnectingIOReactor ioreactor = <...>

ioreactor.setExceptionHandler(new IOReactorExceptionHandler() {

    public boolean handle(IOException ex) {
        if (ex instanceof BindException) {
            // bind failures considered OK to ignore
            return true;
        }
        return false;
    }

    public boolean handle(RuntimeException ex) {
        if (ex instanceof UnsupportedOperationException) {
            // Unsupported operations considered OK to ignore
            return true;
        }
        return false;
    }

});
```

   We rewrote the ExceptionHandler, placed it in the HLRC configurations, and threw an exception in the callback to simulate the problem. However, the exception in the callback was not captured by the I/O Reactor ExceptionHandler, and no exception was thrown during script execution. It was difficult to verify the function of the I/O Reactor ExceptionHandler. According to verification results from other developers in the Elasticsearch

community, they added the I/O Reactor ExceptionHandler a long time ago and have had no problems since. You are advised to configure the I/O Reactor ExceptionHandler, but do not ignore all exceptions.

The Elasticsearch REST Client should include an Exception Handler, rather than requiring users to handle exceptions by setting an **IOReactorExceptionHandler**. Furthermore, this approach does not resolve all types of exceptions.

Solution

1. Adjust the size of the client connection pool based on the site requirements.
2. You are advised to adjust the number of concurrent clients based on the site requirements or configure multiple nodes to share the pressure.
3. You are advised to configure the I/O Reactor ExceptionHandler to handle some exceptions.
4. After the HLRC is called, catch exceptions and check whether they occurred because the I/O Reactor status had changed to STOPPED. Restart the client to restore the connection.

4.8 The Peak Heap Memory of an Elasticsearch Cluster Remains High (Over 90%)

Symptom

The peak heap memory of an Elasticsearch cluster remains high (over 90%). If the heap memory usage of a node does not remain above 90%, the cluster is normal. If the heap memory usage remains above this rate for a long time, the cluster faces a certain risk of unavailability.

Possible Causes

- Check whether there are many tasks waiting in the write and query queues of the cluster.
  ```
  GET /_cat/thread_pool/write?v
  GET /_cat/thread_pool/search?v
  ```
- View the cluster monitoring information. Check the metrics related to the write and query tasks of the cluster.
- If the heap memory usage of the cluster remains high for a long time, check the cluster size and the number of nodes, and scale out the cluster if necessary.

Solution

- Optimize the write and query programs on the client based on the task queuing statistics.
- If the cluster is heavily loaded for a long time, it may cause slow write, query, and frequent node disconnections. To avoid these problems, you can increase nodes or redesign the cluster.

- If the heap memory fluctuates around 95% and nodes are sometimes disconnected, use the traffic control function as needed. For more information, see **Configuring Flow Control for an Elasticsearch Cluster**.

4.9 Failed to Modify the Elasticsearch Cluster Specifications

Symptom

I failed to modify the cluster specifications and an error message is displayed on the console.

Figure 4-7 CSS.0073 error

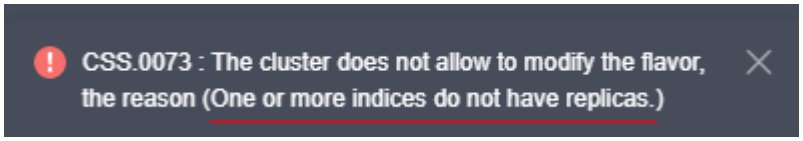
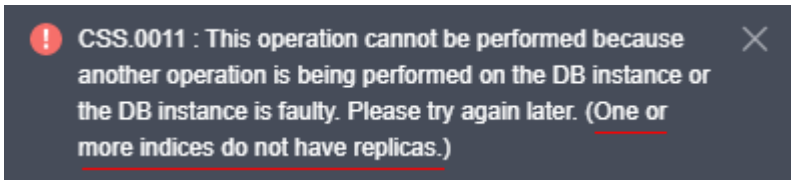


Figure 4-8 CSS.0011 error



Possible Causes

The current index replica count is 0, and the backend has intercepted the specification change request. You need to set the replica count first before performing the specification change operation. Otherwise, there is a risk of shard loss.

Solution

Run the following commands to set backup parameters:

```
PUT /index/_settings
{
   "number_of_replicas" : 1 //Number of replicas
}
```


4.10 An Error Is Returned When I Change the Read-Only Status of an Index

Symptom

When the disk usage of a security cluster exceeds the threshold, all indices will enter a read-only mode (with **read_only_allow_delete** set to **true**), preventing further writes. To restore write access, manually set **read_only_allow_delete** to **false** by running the following command:

```
PUT _settings
{
  "index": {
    "blocks": {
      "read_only_allow_delete": "false"
    }
  }
}
```

The error information is as follows:

```
{
  "error": {
    "root_cause": [
      {
        "type": "security_exception",
        "reason": "no permissions for [] and User [name=admin, roles=[admin], requestedTenant=null]"
      }
    ],
    "type": "security_exception",
    "reason": "no permissions for [] and User [name=admin, roles=[admin], requestedTenant=null]"
  },
  "status": 403
}
```

Possible Causes

By default, a security cluster has an **. opendistro_security** index, which cannot be written. You need to skip this index when changing the status of indexes.

Solution

Use wildcards to match specified indexes. (Use a wildcard to replace the **indexname** in the following example.)

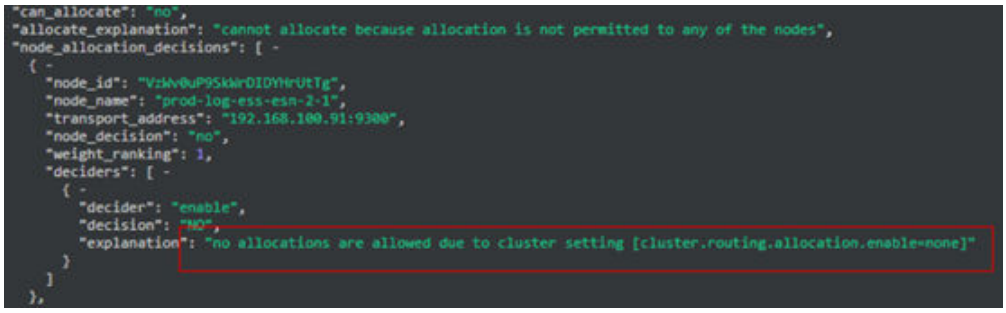
```
PUT indexname/_settings
{
  "index": {
    "blocks": {
      "read_only_allow_delete": "false"
    }
  }
}
```


4.11 A Node in an Elasticsearch Cluster Has No Shards Allocated to It

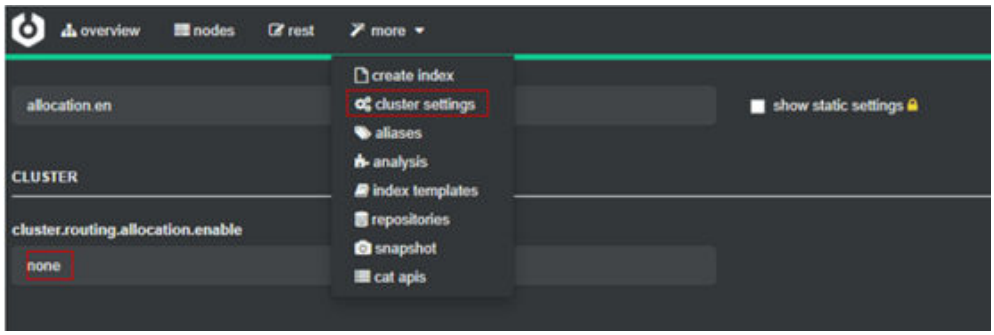
Solution

1. Run the following command to check unallocated shards:
   ```
   GET _cluster/allocation/explain?pretty
   ```

   Figure 4-9 Checking unallocated shards

2. In the cluster list, find the target cluster, and choose **More** > **Cerebro** in the **Operation** column. Log in to Cerebro.
3. On the Cerebro console, choose **more** > **cluster settings**, enter **allocation.enable** in the upper left corner, and change **none** to **all**.

4. If Cerebro is unavailable, run the following command:
   ```
   curl -X PUT "http://Intranet_IP:9200/_cluster/settings" -H 'Content-Type: application/json' -d'{"persistent":{"cluster.routing.allocation.enable": "all"}}'
   ```

4.12 Failed to Write Data to a CSS Cluster Index

Symptom

Failed to write data to an Elasticsearch/OpenSearch cluster index. The following error message is returned:

```
"reason": "blocked by: [FORBIDDEN/12/index read-only / allow delete (api)];"
```

The index has been set to read-only mode (read-only/allow delete), and all write operations (including creating and updating documents) are blocked.

Possible Cause

When the disk usage of the data node reaches 95%, Elasticsearch/OpenSearch automatically sets the index to read-only mode, that is, set **index.blocks.read_only_allow_delete** to **true**, to prevent system breakdown caused by disk space exhaustion. In this case, all write requests are rejected, and only the delete permission is reserved to release space.

The automatic recovery mechanism varies depending on the version:

- Elasticsearch 7.10.2 and OpenSearch: After the disk usage decreases, the read-only mode is automatically disabled.
- Versions earlier than Elasticsearch 7.10.2: After the disk usage decreases, you need to manually disable the read-only mode.

Solution

1. View the cluster monitoring metric to check the disk usage. If the disk usage reaches 95%, clear the disk or expand the disk capacity.
2. Clear the disk or expand the disk capacity.
   - Clear the disk: Delete indexes or expired data that is no longer needed.

     Run the following command to delete a specified index:

     ```
     DELETE /<index-name>
     ```

     CAUTION

     Deleted indexes cannot be restored. Exercise caution.

   - Expand disk capacity: On the cluster management page, locate the target data node or cold data node, click **More** in the **Operation** column, and select **Modify Configuration** to expand the node's disk capacity.
3. Disable the read-only mode for the index.
   - **Elasticsearch 7.10.2 and OpenSearch**

     For these versions, the read-only mode is automatically disabled after the cluster disk usage decreases. Wait for about 1 to 2 minutes and try writing data again to check whether the fault is rectified.

   - **Versions earlier than Elasticsearch 7.10.2**

     For these versions, you need to manually disable the read-only mode after the cluster disk usage decreases.

     i. Run the following commands to disable the read-only mode for all indexes:

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

     CAUTION

     Before running this command, ensure that the disk space is sufficient. Otherwise, read-only protection may be triggered again.

ii. Wait for about 1 to 2 minutes and try writing data again to check whether the fault is rectified.

4. To prevent the fault from re-occurring, you are advised to:
   - Periodically monitor the disk usage of the node and configure an alarm threshold (85% is recommended).
   - Periodically delete expired indexes and documents to release disk space.
   - Plan sufficient storage capacity and reserve 15% to 20% redundant disk space.

4.13 Error Message "maximum shards open" Is Displayed When Users Try to Create an Index

Symptom

When users try to create an index, an error message is displayed: "this action would add [2] total shards, but this cluster currently has [1000]/[1000] maximum shards open".

Possible Cause

By default, each node can support a maximum of 1000 shards. An error is reported when this limit is exceeded.

Solution

- Solution 1: Disable or delete unused indexes to reduce the number of shards.
- Solution 2: Change the limit on the maximum number of shards per node. For details, see **max_shards_per_node**.

```
PUT _cluster/settings
{
  "persistent": {
    "cluster": {
      "max_shards_per_node": 2000
    }
  }
}
```

NOTE

Changing the limit on the maximum number of shards per node is just a temporary fix. To solve this issue in the long term, aim for 20 shards or fewer per GB of JVM heap memory (each node should have a heap memory size that is 50% of the node's available memory, up to 31 GB). For more information, see **shard count recommendation**.

4.14 Error Message "403 Forbidden" Is Displayed When I Delete All Indexes

Symptom

When I run the **curl –i –u admin:password –XDELETE https://ip:9200/_all** command (**password** is the admin account password and **ip** is the private network

address of the cluster) to delete all indexes, the error message "403 Forbidden" is reported.

Solution

The index **.opendistro_security** cannot be deleted from a security cluster. The command for deleting all indexes is invalid for clusters in the security mode. Use the index name or wildcard to delete specified indexes. Do not delete all indexes at a time.

4.15 "Forbidden" Is Returned When I Try to Delete an Index Pattern

Symptom

When I delete an index pattern on Kibana, the error message "Forbidden" is displayed.

Possible Causes

The reason why the index pattern could not be deleted was that the .kibana index was read-only. This typically happens when the disk usage exceeds a preset threshold.

Solution

On the **Dev Tools** page of Kibana, run the following command to cancel the read-only state of the .kibana index:

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

4.16 "Trying to create too many scroll contexts" Is Returned When the update-by-query Command Is Executed

Symptom

When I run the **update-by-query** command in an Elasticsearch cluster, the error message "Trying to create too many scroll contexts." is displayed. The specific error information is as follows:

```
{"error":{"root_cause":[{"type":"exception","reason":"Trying to create too many scroll contexts. Must be less than or equal to: [100000]. This limit can be set by changing the [search.max_open_scroll_context] setting."},{"type":"exception","reason":"Trying to create too many scroll contexts. Must be less than or equal to: [100000]. .....
```


Possible Causes

Each time a cluster calls the scroll creation API, a scroll context is created. When the number of scroll contexts reaches the preset threshold, no more scrolls can be created.

Procedure

Run the following command to check the value of **open_contexts** (the number of scroll contexts):

```
GET /_nodes/stats/indices/search
```

When the value reaches the upper limiter, use either of the following methods to solve the problem.

- Method 1: Delete unnecessary scrolls to release the storage space of scroll contexts.

  ```
  DELETE /_search/scroll
  {
    "scroll_id": "DXF1ZXJ5QW5kRmV0Y2gBAAAAAAAAAD4WYm9laVYtZndUQlNsdDcwakFMNjU1QQ=="
  }
  ```

- Method 2: Change the value of **search.max_open_scroll_context** to increase the storage space for scroll contexts.

  ```
  PUT /_cluster/settings
  {
    "persistent": {
      "search.max_open_scroll_context": 50000
    },
    "transient": {
      "search.max_open_scroll_context": 50000
    }
  }
  ```

  The value used in this example is for reference only. Set this value based on the actual size of your cluster.

4.17 Index Patterns Cannot Be Created for an Elasticsearch Cluster

Symptom

When I click **Create index pattern** on the Kibana console, nothing happens, so I cannot create an index pattern.

Possible Cause

1. Check whether the disk is full, which leads to the read-only state of the .kibana index.
2. Check whether there are multiple Kibana indexes.

Solution

Run the following command to delete unnecessary indexes to release disk space:

```
PUT .kibana/_settings
{
  "index": {
```



```
  "blocks": {
    "read_only_allow_delete": "null"
  }
 }
}
```

4.18 Troubleshooting Method for Logstash Pipeline Hot Stop Failure

Symptom

A Logstash cluster writes data to a target Elasticsearch cluster using the **logstash-output-elasticsearch** plug-in. When updating the pipeline configuration using the hot stop feature, the hot stop fails, and the pipeline becomes blocked and unable to close properly. The following error message appears in the runtime logs: **The shutdown process appears to be stalled due to busy or blocked plugins**

Troubleshooting

Table 4-5 Possible causes and troubleshooting methods

No.	Probable Cause	Key Error Characteristics	Handling Method
1	Target index set to read-only	HTTP 403 + cluster_block_exception	For details, see **Cause 1: Target Index Read-Only**.
2	Target cluster JVM memory circuit breaker triggered	HTTP 429 + circuit_breaking_exception	For details, see **Cause 2: Target Cluster JVM Memory Circuit Breaker**.

1. Log in to the **CSS management console**.
2. In the navigation pane on the left, choose **Clusters** > **Logstash**.
3. In the cluster list, click the target cluster name to enter the cluster details page.
4. Click the **Configuration Center** tab and click **Run Logs** above the pipeline list.
5. Search for error information in the runtime logs.
   - **Typical error log example for target index read-only**
     - Write phase:

       ```
       [2026-07-31T08:30:38,426][ERROR][logstash.outputs.elasticsearch][pipeline.name][plugin.id] Encountered a retryable error. Will Retry with exponential backoff {:code=>403, :url=>"http://xx.xx.xx.xx:9200/_bulk"}
       .
       .
       .
       [2026-07-31T06:39:25,387][INFO ][logstash.outputs.elasticsearch][pipeline.name][plugin.id] retrying failed action with response code: 403
       ```



```
({"type"=>"cluster_block_exception", "reason"=>"index [target_index_name] blocked by: [FORBIDDEN/8/index write (api)];"})
```

- Hot stop phase:

```
[2026-07-31T06:33:21,217][ERROR][org.logstash.execution.ShutdownWatcherExt] The shutdown process appears to be stalled due to busy or blocked plugins. Check the logs for more information.
```

– **Typical error log example for target cluster JVM memory circuit breaker**

- Write phase:

```
[2026-07-31T08:30:38,426][ERROR][logstash.outputs.elasticsearch][pipeline.name][plugin.id] Encountered a retryable error. Will Retry with exponential backoff {:code=>429, :url=>"http://xx.xx.xx.xx:9200/_bulk"}
.
.
.
[2026-07-31T06:39:25,387][INFO ][logstash.outputs.elasticsearch][pipeline.name][plugin.id] retrying failed action with response code: 429 ({"type"=>"circuit_breaking_exception", "reason"=>"fake circuit break here!!!!!!"bytes_wanted"=>0, "bytes_limit"=>0", "durability"=>"TRANSIENT"})
```

- Hot stop phase:

```
[2026-07-31T08:31:13,951][ERROR][org.logstash.execution.ShutdownWatcherExt] The shutdown process appears to be stalled due to busy or blocked plugins. Check the logs for more information.
```

Cause 1: Target Index Read-Only

The **index.blocks.write** property of the target index is set to **true**. When Logstash writes data, it encounters a write block and receives an HTTP 403 rejection error. The **logstash-output-elasticsearch** plug-in treats the 403 error as a retryable error and continuously retries. Since the retry loop does not check the **@stopping** flag, the hot stop process waits for all in-flight events in the pipeline to complete before closing. However, events cannot complete due to continuous retries, causing the pipeline to block and time out.

Solution:

1. Search for the **403** or **cluster_block_exception** keyword in the runtime logs.

   If the runtime logs contain HTTP 403 errors with **cluster_block_exception**, confirm this as cause 1 and proceed with the following steps.

2. Force stop the pipeline.

   WARNING

   Stopping all pipelines will directly interrupt all running pipeline tasks in the cluster. Data in transit may experience brief delays or offset resets. Ensure that the source data is traceable.

   a. On the **Configuration Center** tab, click **Stop All** above the pipeline list.

   b. In the displayed dialog box, click **OK**.

   c. Check the **Status** column in the pipeline list to confirm that all pipelines display a status of **Stopped**.

3. Remove the read-only status of the target index.

Log in to the Kibana page of the target Elasticsearch cluster. In *Dev Tools, run the following command to remove the index read-only status:

```
PUT <index-name>/_settings
{
  "index.blocks.write": false
}
```

4. Restart the pipeline tasks.
   a. In the configuration file list on the **Configuration Center** tab, select the configuration files to start and click **Start Logstash** above.
   b. In the displayed dialog box, confirm the persistent configuration and click **OK**.
   c. In the pipeline list, confirm that the target pipeline's **Status** displays **Running** and the **Events** column data is continuously updating, indicating that the data migration task has resumed.

Cause 2: Target Cluster JVM Memory Circuit Breaker

The JVM memory usage of the target Elasticsearch cluster exceeds 95%, triggering the parent-level circuit breaker mechanism and returning a BadResponseCodeError 429 error. Similar to cause 1, the **logstash-output-elasticsearch** plug-in treats the 429 error as a retryable error and continuously retries. This causes the hot stop process to wait for in-flight events to complete, resulting in pipeline blocking and timeout.

Solution:

1. Search for the **429** or **circuit_breaking_exception** keyword in the runtime logs.

   If the runtime logs contain HTTP 429 errors with **circuit_breaking_exception**, confirm this as cause 2 and proceed with the following steps.

2. Force stop the pipeline.

   WARNING

   Stopping all pipelines will directly interrupt all running pipeline tasks in the cluster. Data in transit may experience brief delays or offset resets. Ensure that the source data is traceable.

   a. On the **Configuration Center** tab, click **Stop All** above the pipeline list.
   b. In the displayed dialog box, click **OK**.
   c. Check the **Status** column in the pipeline list to confirm that all pipelines display a status of **Stopped**.

3. Restore JVM memory in the target cluster.

   Reduce the JVM memory usage of the target Elasticsearch cluster using the following methods:

   - **Reduce write pressure**: Decrease the number of concurrent write requests and reduce the batch write size.
   - **Release memory**: Run the following command in Kibana to trigger cache clearing:

     ```
     POST /_cache/clear
     ```


- **Scale out/up the cluster**: Increase the number of nodes in the target cluster or upgrade node specifications to increase JVM memory capacity.

4. Restart the pipeline tasks.
   a. In the configuration file list on the **Configuration Center** tab, select the configuration files to start and click **Start Logstash** above.
   b. In the displayed dialog box, confirm the persistent configuration and click **OK**.
   c. In the pipeline list, confirm that the target pipeline's **Status** displays **Running** and the **Events** column data is continuously updating, indicating that the data migration task has resumed.

Submitting a Service Ticket

If the problem persists, **submit a service ticket**.

5 Ports

5.1 Port 9200 Cannot Be Reached

Symptom

If a VPN or VPC peering connection is used to access the CSS cluster, no result is returned when the curl command is used to connect to an Elasticsearch cluster.

For example, if you run the following command to connect to the cluster, no result is returned:

```
curl -s 'http://< node private access address >:9200'
```

Possible Causes

If a VPN or VPC peering connection is used to access CSS, that means that the client and CSS are not in the same VPC. Therefore, the subnet of the CSS cluster must be in a different network segment from that of the VPC.

For example, consider a CSS cluster that uses the VPC **vpc-8e28** with a CIDR block of **192.168.0.0/16**. The subnet **subnet-4a81** under this VPC is selected, and the CIDR block of **subnet-4a81** is the same as that of **vpc-8e28**, both being **192.168.0.0/16**. In this case, if a VPN private line or a VPC peering connection is used to access the CSS cluster, there will be no gateway route corresponding to the VPC under this subnet. This affects the configuration of the default route for the CSS service, ultimately causing the port 9200 access failure.

Procedure

If port 9200 cannot be reached but the CSS cluster is available, perform the following steps to rectify the issue:

1. Log in to the **CSS management console**.
2. In the navigation pane on the left, expand **Clusters**. Select a cluster type based on the target cluster. The cluster list is displayed.
3. In the cluster list, click the name of the target cluster. The cluster information page is displayed.

4. Click the **Overview** tab.
5. In the **Network Information** area, check the **VPC** and **Current Subnet** used by the cluster.
6. Click the VPC name to go to the VPC details page. Check the CIDR block of the VPC and subnet.

   If the VPC and the subnet are on the same CIDR block, when a VPN private line or a VPC peering connection is used, access to port 9200 will fail.
7. If the preceding error occurs, create another cluster and this time select a subnet that is different from the VPC subnet. If the subnet does not exist, create another subnet on the VPC management console.

   After a new CSS cluster is created, migrate the data of the old cluster to the new cluster, and then use the VPN or VPC peering connection to access the cluster.

CAUTION

If you require a VPN connection or VPC peering connection to access the CSS cluster, ensure that the VPC and subnet of the newly created CSS are in different network segments.

