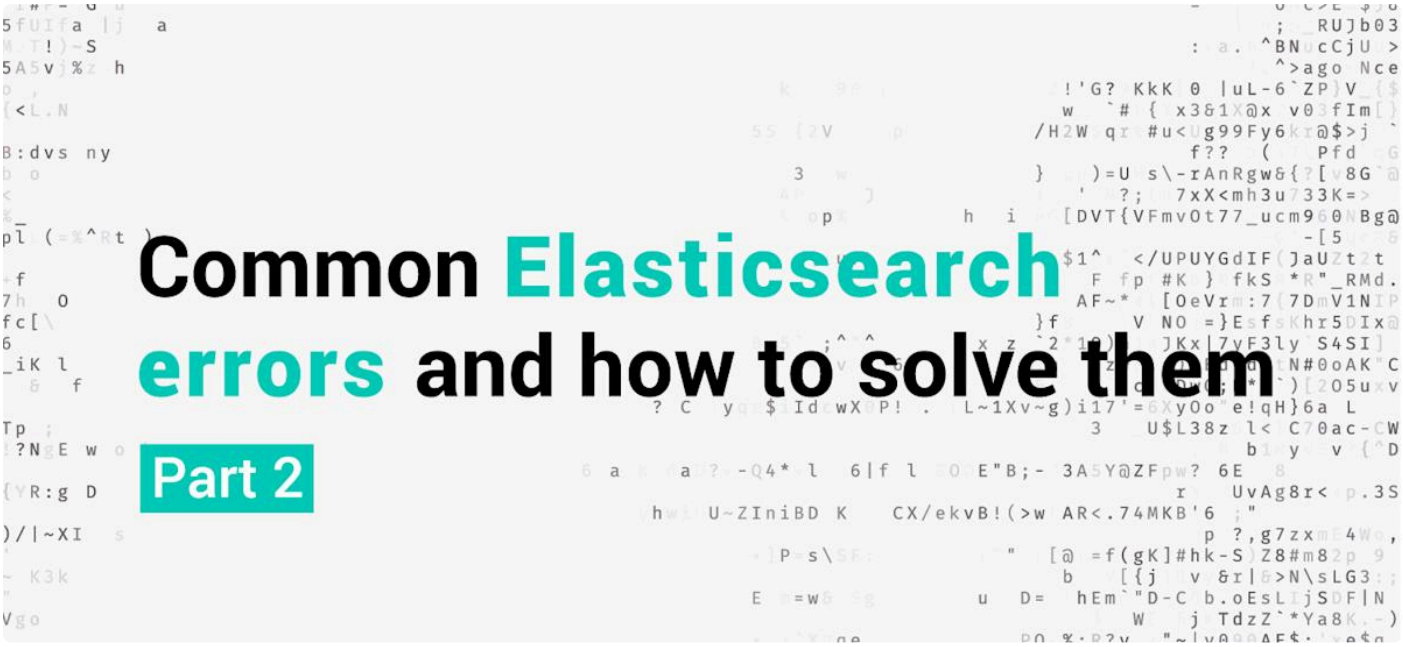
Common Elasticsearch errors and how to solve them - part 2

Jochen Kressin · 2024-04-29



Elasticsearch, being a complex distributed system, can encounter various issues during its operation. Some of the most common error messages in Elasticsearch relate to problems with cluster health, data indexing, querying, and configuration. Understanding these error messages is crucial for effective troubleshooting and maintaining the health of your Elasticsearch environment. Here are some of the most common Elasticsearch error messages:

SearchPhaseExecutionException

The "SearchPhaseExecutionException" in Elasticsearch is an error encountered during the execution of a search query. This exception can be triggered for a variety of reasons and usually indicates an issue with either the query itself or the underlying data it is trying to access.

Source of the Issue:

Query Parsing Errors: The most common cause is a malformed query. This could be due to incorrect syntax, invalid query parameters, or unsupported features in the query.

Shard Failures: The error can occur if one or more shards involved in the search operation fail. Shard failures can be due to issues like hardware problems, network issues, or corrupted indices.

Resource Constraints: If the cluster is under heavy load or running out of resources (like memory or CPU), it might fail to execute the query, resulting in this exception.

Mapping Issues: Inconsistencies or conflicts in the field mappings of the index being queried can also trigger this error.

Version Incompatibilities: If you're using features in your query that are not supported by the version of Elasticsearch you are running, it can lead to this exception.

Remedies:

Validate and Correct the Query: Check your query syntax and structure. Tools like Kibana's Dev Tools can help test and format your queries. Ensure that the fields and data types used in the query match those defined in your index mappings.

Address Shard Failures: Investigate the health of your shards using the `_cat/shards` API or Kibana. Look into logs to find any indications of why the shard might be failing (like hardware issues, disk errors, etc.). If a shard is corrupted, you might need to restore it from a backup or reindex the data.

Manage Cluster Resources: Monitor the cluster's performance metrics to identify resource bottlenecks. Scale your cluster resources (add more nodes, increase CPU/memory) as needed. Optimize your queries to be more resource-efficient and use filters to narrow down results.

Resolve Mapping Conflicts: Review and update your index mappings to resolve inconsistencies or conflicts. Reindex your data if necessary to apply the new mappings.

Ensure Version Compatibility: Check the Elasticsearch documentation to ensure that your query syntax and features are compatible with your cluster's version. Upgrade your Elasticsearch version if you need features that are not available in your current version.

NoNodeAvailableException

The "NoNodeAvailableException" in Elasticsearch is an error indicating that the client is unable to connect to any nodes in an Elasticsearch cluster. This error typically suggests issues with network connectivity, cluster configuration, or the health of the Elasticsearch nodes themselves.

Source of the Issue:

Network Connectivity Problems: The most common cause is network issues between the client and the Elasticsearch cluster. This could be due to network outages, firewall rules, or incorrect network configurations.

Cluster Configuration Issues: Incorrectly configured Elasticsearch nodes, especially concerning network settings (like `network.host` in `elasticsearch.yml`), can lead to this exception.

Client Configuration Errors: The client might be configured with the wrong address, port, or protocol for connecting to the Elasticsearch nodes.

Elasticsearch Service Down: If the Elasticsearch service is not running on the expected nodes, the client will not be able to connect.

Version Mismatch: A mismatch in versions between the client and the Elasticsearch cluster can sometimes lead to connectivity issues.

Remedies:

Check Network Connectivity: Verify network connectivity between the client and the Elasticsearch nodes. Ensure that the correct ports are open and not blocked by firewalls or network security groups.

Validate Cluster Configuration: Review the `elasticsearch.yml` configuration file on your Elasticsearch nodes. Pay particular attention to settings like `network.host` and `cluster.initial_master_nodes`. Ensure that the Elasticsearch service is running on the nodes and is accessible.

Review Client Configuration: Check the client's configuration to ensure it's pointing to the correct cluster nodes, ports, and protocol (HTTP or HTTPS). Make sure that any authentication credentials or certificates required by the cluster are correctly configured in the client.

Restart Elasticsearch Services: If the Elasticsearch service is down, restart it on the affected nodes. Monitor the logs for any startup errors or issues.

Cluster Health Check: Use the `_cluster/health` API to check the health of your cluster. Look for any reported issues with nodes or shards.

EsRejectedExecutionException

The "EsRejectedExecutionException" in Elasticsearch is an error that occurs when the Elasticsearch cluster rejects an operation because it's unable to handle it at the given moment. This exception is usually related to the cluster being overwhelmed with tasks, leading to an overload of its internal queues.

Source of the Issue:

Thread Pool Queue Overload: Elasticsearch uses thread pools for different types of operations like searching, indexing, etc. When a thread pool's queue fills up, additional tasks get rejected, causing this exception.

High Cluster Load: High load on the cluster, either due to heavy indexing or search operations, can lead to the saturation of thread pools.

Inadequate Resource Allocation: Inadequate hardware resources (CPU, memory) or improperly configured thread pool sizes can contribute to this issue.

Bulk Operations: Bulk indexing operations with too many documents or very large individual documents can easily overwhelm the thread pools.

Remedies

Optimize Bulk Operations: If bulk indexing is causing the issue, optimize the size and frequency of your bulk requests. Smaller, more frequent bulk requests are often more manageable for Elasticsearch. Ensure that the documents themselves are not excessively large.

Adjust Thread Pool Settings: While not generally recommended, as a temporary measure, you can increase the size of thread pool queues. However, this is usually a stop-gap solution and might only delay the problem. It's crucial to understand the implications of changing thread pool settings and to do so cautiously.

Scale Your Cluster: If your cluster is consistently hitting capacity, it might be time to scale up. Adding more nodes or increasing the capacity of existing nodes (more CPU, RAM) can help distribute the load more effectively. Consider using Elasticsearch's autoscaling feature, if available in your version, for dynamic scaling based on workload.

Review and Optimize Queries: Optimize your search queries to be as efficient as possible. Avoid overly broad queries and use filters appropriately. Review your indexing strategy. Frequent small updates or poorly structured documents can add unnecessary load.

Use Back-Off Strategies in Client Applications: Implement back-off strategies in your client applications. When you receive an `EsRejectedExecutionException`, have your application wait and then retry the operation after a delay.

CircuitBreakingException

The "CircuitBreakingException" in Elasticsearch is a mechanism to prevent the system from running out of memory. This exception occurs when a request would cause the memory usage to exceed a predefined limit, triggering the "circuit breaker" to open and reject the request to maintain the stability of the system.

Source of the Issue:

Memory Usage Limits: Elasticsearch has several built-in circuit breakers to prevent OutOfMemory errors. These include the request circuit breaker, the fielddata circuit breaker, and the parent circuit breaker. When the memory used by any of these components exceeds its set threshold, the circuit breaker trips.

Large or Complex Requests: A common trigger for this exception is large or complex queries, especially those involving significant aggregations, fielddata, or sorting operations.

High Cardinality Data: High cardinality fields (fields with a large number of distinct values) used in aggregations or sorting can consume a lot of memory, leading to this error.

Inefficient Data Structures: Certain data structures or queries might be inefficient and use more memory than necessary.

Remedies:

Optimize Queries and Mappings: Review and optimize your queries to be less memory-intensive. For instance, limit the size of aggregations and avoid using high-cardinality fields for sorting or aggregating. Optimize your index mappings to reduce memory usage, such as using `doc_values` for fields involved in sorting and aggregations.

Adjust Circuit Breaker Settings: While not generally recommended, as a short-term fix, you can adjust the settings of the circuit breakers to allow more memory usage. However, this can lead to other issues like OutOfMemory errors. It's crucial to understand the implications of changing these settings and to do so cautiously.

Scale Your Elasticsearch Cluster: If your cluster is consistently reaching memory limits, consider scaling up the cluster. Adding more nodes or increasing the resources (like memory) of existing nodes can help

distribute the load and reduce the risk of hitting circuit breaker limits.

Use Resource Capping on Fields: Implement resource capping strategies, such as limiting the number of terms in aggregations or using filters to narrow down the data involved in memory-intensive operations.

Monitor Memory Usage: Regularly monitor memory usage in your Elasticsearch cluster. Tools like Elasticsearch's node stats API, Kibana monitoring features, or monitoring solutions like Signals can be used for this purpose.

6. **Review Data Volume and Usage Patterns:** Consider the volume of data being processed and the typical usage patterns. Large datasets or spikes in query volume can contribute to memory pressure. Implement strategies to handle large datasets more efficiently, such as using pagination for large result sets.

Where to go next

- Read the previous article in this series
- Ask any question you have on our Search Guard Forum
- Want to learn more about a specific error message? Just drop us a note!

This article was published on the Search Guard blog: https://search-guard.com/blog/common-elasticsearch-errors-and-how-to-solve-them-part-2/