HTTP Error Troubleshooting

4xx Indexing and Search Errors

It’s important to note that 400/4xx errors indicate that the errors are being caused by client side configurations. Client Side Errors. Common 400 errors that are observed in Amazon Opensearch include 401, 403, 413, 429, 460 HTTP response codes. In this wiki guide, we will be looking into what these common 400 errors mean and how to troubleshoot them.

SECURITY ERRORS:

401 AND 403 ERRORS ARE SECURITY RELATED ERRORS THAT MEAN THAT SECURITY CONFIGURATIONS ON THE CLIENTS THAT ARE RECEIVING THESE ERRORS FROM THE OPENSEARCH ENDPOINT.

401 Unauthorized

This error means that you do not have the correct access configuration or security credentials configured on your client. The 401 Unauthorized error can also be from not signing requests with IAM identity required by the domain’s access policy configuration. Consider the following measures with regards to your client access:

- If you are using a client that doesn't support request signing (such as a browser), then consider using an IP-based access policy. IP-based policies allow unsigned requests to an OpenSearch Service domain.
  - Note: VPC Domains do not support IP-Based Access Policy. VPC Security Groups are the Network Access Control for your domain.
- Use a client that supports request signing with Signature Version 4 signing to add authentication headers to your requests.
  - Requests from clients that aren't compatible with Signature Version 4 are rejected with a "User: anonymous is not authorized" error. For examples of correctly signed requests to OpenSearch Service, see Making and signing OpenSearch Service requests.
- Opensearch Dashboards doesn’t support request signing. If your access control policy has required IAM users or roles domain access, then configure the domain to use Amazon Cognito authentication for OpenSearch Dashboards. Otherwise, your IAM role or user receives an unauthorized error when accessing the OpenSearch Dashboards domain.

References:
https://docs.aws.amazon.com/opensearch-service/latest/developerguide/ac.html
https://repost.aws/knowledge-center/anonymous-not-authorized-opensearch

403 Forbidden

This error means that you do not have the appropriate permissions to do the requested operation or run the API. This can happen while you have other permissions to run other API’s to the domain. When this error occurs, please check your IAM Access Policy to ensure that IAM User/Role have the required permissions attached through IAM Policy. With Fain Grained Access Control added, make sure you have checked the Permissions and Role Mappings to ensure those are set accordingly as well.

Reference:

https://docs.aws.amazon.com/opensearch-service/latest/developerguide/ac.html

https://docs.aws.amazon.com/opensearch-service/latest/developerguide/fgac.html#fgac-access-policies
https://docs.aws.amazon.com/opensearch-service/latest/developerguide/fgac.html

https://opensearch.org/docs/latest/security/access-control/permissions/

403 HTTP Errors - Part 2

The 403 HTTP Error can also occur during non-security related issues. This particular includes:

- ***"Cluster_Block_Exception"*** error when write blocks occur due to low disk space on or more data nodes. It can also be caused by high JVMMemoryPressure exceeding 92% for 30 minutes.
  - Check *FreeStorageSpace* & *JVMMemoryPressure* CloudWatch metrics.
- ***"Request throttled due to too many requests"*** as a 403 error is request throttling that is indicating that the cluster's compute is under scaled for the incoming requests at that time.
  - Increase log verbosity in client to debug requests or scale the cluster based on the compute bottleneck.

Reference:
https://docs.aws.amazon.com/opensearch-service/latest/developerguide/handling-errors.html
https://docs.aws.amazon.com/opensearch-service/latest/developerguide/handling-errors.html#troubleshooting-cluster-block
https://docs.aws.amazon.com/opensearch-service/latest/developerguide/handling-errors.html#troubleshooting-throttle-api

CLIENT PAYLOAD ERRORS:

THE MOST COMMON CLIENT PAYLOAD ERROR THAT WILL BE INCURRED THE CLIENTS REQUEST SIZES AND NETWORK CONFIGURATIONS SUCH AS MTU AND USAGE OF JUMBO FRAMES WILL HAVE TO BE REVIEWED AND CONFIGURED ACCORDINGLY WITH DOCUMENTED LIMITS FOR THE AMAZON OPENSEARCH SERVICE.

413 Payload Too Large:

This error is encountered when 1 or more request is larger than the server is willing or able to process. When this response code is encountered, resolution will require you to consider the following steps and measures:

-
  - Adjust the request size in your client
  - Check the Network limits documentation which shows the maximum request payload size by instance type.
  - Use instances with 100MB network limits

References:
https://repost.aws/questions/QUa7261SUURa2k45nv4w1ixQ/opensearch-request-entity-too-large

PERFORMANCE ERRORS:

429 AND 460 ERRORS ARE PERFORMANCE ERRORS CAUSED BY THE CLIENT CONFIGURATION OR SCALE OF THE CLUSTER. TO ADDRESS EACH OF THESE TWO ERRORS, THE CLIENT SIDE WILL REQUIRE MORE TROUBLESHOOTING THAN THE SERVER SIDE.

WHEN TAKING THE SERVER SIDE INTO CONSIDERATION FOR ADDRESSING 429 AND 460 ERRORS, THESE ERRORS MEAN THAT THE DOMAIN RETURNING THESE ERRORS IS NOT PROCESSING REQUESTS FAST ENOUGH FOR YOUR CLIENTS AND INCOMING TRAFFIC. MEANING, IF YOU ARE NOT ABLE TO TWEAK AND OPTIMIZE THE CLIENT SIDE CONFIGURATIONS WITH REGARDS TO TRAFFIC, PAYLOAD/BUFFER SIZES, RETRIES AND TIMEOUTS; THEN YOUR DOMAIN WILL NEED TO BE SCALED OUT OR SCALED UP IN ORDER TO PROCESS TRAFFIC WITHIN TIME WITHOUT COMPROMISING THE CLUSTERS COMPUTE RESOURCES AND PERFORMANCE.

429 Too Many Requests - Part 1

This error occurs when the clients are sending more requests than the cluster's CPU Core count can handle at that time. The core count of each individual data node in the cluster determines how big your cluster's threadpool size when all cores (threads) are added up. And each core/thread has a queue that will hold up to 1000 requests on each core per node. When the threadpool queue of each core gets to 1000 requests waiting in queue, the server will start to reject requests and clients sending requests to that node will start to see 429 errors.

Once 429 errors are observed, two measures need to be taken. The first that does not affect compute costs is to optimize clients where possible to not overwhelm the cluster. However, the flexibility and option to opitimize the clients is determined by the use-case and how soon data needs to be ingested and searched.

The second option is to scale out the cluster to reduce shard allocation per node, where more shards can result in more operations that fill up the threadpool queue during live traffic. It is important to note that when scaling out, it is done in order to avoid hot spots and data/storage skew that can lead to hot spots in the cluster based on shard allocation decider.

References:

https://docs.aws.amazon.com/opensearch-service/latest/developerguide/handling-errors.html#troubleshooting-search-backpressure

https://docs.aws.amazon.com/opensearch-service/latest/developerguide/handling-errors.html#handling-errors-index-skew
https://docs.aws.amazon.com/opensearch-service/latest/developerguide/handling-errors.html#troubleshooting-throttle-api

429 Too Many Requests - Part 2

What makes 429 errors complex in troubleshooting is that they tie back to how the cluster and data were designed. When designing Opensearch cluster before deployment and first use, it is important to do this cluster sizing math to avoid threadpool shortage and exhausting threadpoolqueues during high search and indexing traffic to the cluster.

The most important factors to consider during cluster and data design are:

1) Storage calculation of current data and expected growth.

- This determines sharding strategy.
- How many shards do I need?
- How big should my shards be
- This answer determines the number of compute nodes needed.

2) Number of data nodes for compute and storage type.

- Primary shards need even/parallel distributed compute.
- Consider NVME SSD for Write Heavy/Big Data (Lake) use cases.

3) Expected traffic (requests)

- Traffic will ultimately determine number of threads (threadpoolsize) needed.
    - Maximum Write Threads = # of available_processors
    - Maximum Search Threads = ((# of available_processors * 3) / 2) + 1
    - available_processors is the vCPU count of EC2 instance type
- For traffic load testing It's important to run benchmark tests of expected traffic before production deployment.

  - Opensearch Benchmark Tool available for load testing

References:
https://repost.aws/knowledge-center/opensearch-resolve-429-error
https://opensearch.org/docs/latest/benchmark/

- **460 Timeout Error**

The 460 timeout occurs when an Opensearch client closes a connection with the Opensearch Server + Load Balancer before a processed response is returned back to the client.Troubleshooting 460 timeouts requires a few considerations that include:

1) Increase client timeout setting in the client configuration to allow connections to be open longer.

- This can sometimes be a “keep-alive“ setting.

2) Adjust the request sizes in your client for faster processing of smaller batches.

3) Increase or optimize compute to process requests faster. *It's important to consider the optimizations and impact on costs they could have* when scaling up vertically for faster processing of requests is require

- Scaling up CPU will increase CPU cores for multi thread processing. This will ensure lower CPU Utilizations.
- Scaling up Memory will increase JVM Heap for a bigger request cache with less garbage collection cycles.
- Scaling up Disks will increase Read performance for your search requests and Write performance for your indexing requests.
- EBS GP3 for Provisioned IOPS where EBS Throttling is occurring in GP2 volumes
- NVME SSD options for write latency and heavy disk I/O use cases.

References:
https://docs.aws.amazon.com/elasticloadbalancing/latest/application/load-balancer-troubleshooting.html#http-460-issues
https://repost.aws/knowledge-center/opensearch-timeout-issues

5xx Errors

HTTP 503 – SERVICE UNAVAILABLE

This error occurs when the server cannot handle the request (because it is overloaded or down for maintenance).

STEPS TO REMEDIATE

- Check for performance issues and consider scaling the cluster.
- Reduce the number of concurrent requests to the domain.
- Check shard configuration for skew, maybe a single node is overloaded and has become unresponsive.
- Reduce the scope of your query.
- Avoid running **select** * wildcard queries on large indices.
- If there is no dedicated master node for domain , it needs to add it
- If dedicated master node is under heavy load , consider scaling up the master pool
- Check the nodes to see if ES is running or if there is OOM on the nodes
- Check the cluster to be balanced for indexes with heavy load
- Check the number of shards per node and the size of shards (not too big not too small)

Troubleshooting Metrics: JVMMemoryPressure, CPUUtilization, MasterReachableFromNode, Nodes

Troubleshooting Tooling :

- Tumbler for checking AOS metrics
- TMS issues
- sending API requests to the domain,
- Timber for checking reason for 5xx in ES logs.

Escalation:
If there are no visible performance issues , if there is OOM on the node, or if AOS service has been killed with indications of a service issue, then an escalation to support ops or Service Team is required.
Be sure to check for TMS issues for clues that can lead to the cause and relevant data to add to your escalation ticket. Timestamps are very important here.

References:
https://aws.amazon.com/premiumsupport/knowledge-center/elasticsearch-http-503-errors/

HTTP 504 – GATEWAY TIMEOUT

Back end failed to respond in the time out period. The ALB closes the connection after the idle timeout period which is 300 sec. This code return by the back-end instances when the ES service doesn't even have enough resources to accept the incoming requests

Steps to remediate

- Consider scaling up the domain
- Check the snapshot , if it is working and not completed yet , the error might be due to that
- Use bulk to reduce the request overhead
- Check the size of the request response. The big payload can cause this error
- Adopt exponential back off and retry mechanism for the client side.
- Complicated queries need longer time and more resources to be done, consider tuning index and search to reduce this erorr. Slow logs can help if a particular query is taking a long time to complete. On identifying the query from slow logs, the customers can consider further query optimization. They can also use the "profile" API to identify which phase of the query takes longer to execute, and optimize it accordingly.

Troubleshooting Metrics: JVMMemoryPressure, CPUUtilization, ELB Logs and Metrics

Troubleshooting Tooling:
Review metrics in Tumbler or CloudWatch Impersonate.

Reference:
https://aws.amazon.com/premiumsupport/knowledge-center/elasticsearch-http-504-gateway-timeout/