Comprehensive Strategy for the Storage and Monitoring of Transaction Logs in OpenSearch

This document describes the best strategy for storing, indexing, and visualizing transaction logs in OpenSearch, optimizing traceability and facilitating real-time error detection.

1 Creation of Indices by correlationId

Instead of storing all logs in a single index (`transactions-logs`), each transaction will have its own index, which improves query speed and facilitates error tracking.

1.1 Index Format

Each transaction is stored in an index with the following format:

transactions–logs–<correlationId>

Example of generated indices:

transactions–logs–4B1D9F02912746CCA7455FBDC5
transactions–logs–7A6C2E94F0A34C5DB4C7D7A9D2E8A3F1

If the `correlationId` already exists, the log is inserted into its corresponding index. If the `correlationId` is new, an index for the transaction is automatically created.

2 Indexing Logs in the Corresponding Index

Upon receiving a log with a `correlationId`, it is indexed in the corresponding index:

```
POST transactions-logs-4B1D9F02912746CCA7455FBDC5/_doc/
{
  "timestamp": "2025-02-12T10:41:12.798Z",
  "stepNumber": 1,
  "logLevel": "INFO",
  "correlationId": "4B1D9F02912746CCA7455FBDC5",
  "messageId": "1d6b8f37646440ee991f6e60b9cc73eb",
  "transactionId": "9133c07b193f40bb9e2a0d02dfc755e1",
  "eventType": "REQUEST_RECEIVED",
  "serviceName": "agent",
  "message": "Payment request received",
  "errorDetails": null
}
```

Each transaction has its own index, avoiding searches through large volumes of data. A `stepNumber` is assigned to improve the sequential reading of events.

3 Handling Logs Without correlationId or messageId

Some logs do not include `correlationId` or `messageId`, but they contain relevant information. To avoid losing traceability:

1. Creation of the message-to-correlation Index. This index will act as a dictionary where each document will store the relationship between a messageId and its correlationId. Index creation:

```
PUT message-to-correlation
{
  "settings": {
    "number_of_shards": 1,
    "number_of_replicas": 1
  },
  "mappings": {
    "properties": {
      "messageId": { "type": "keyword" },
      "correlationId": { "type": "keyword" },
      "transactionId": { "type": "keyword" },
      "timestamp": { "type": "date" }
    }
  }
}

```

This allows for instantaneous queries to retrieve a correlationId based on a messageId.

2. Implementation of the correlation-to-message Map.
   This index will store the inverse relationship, allowing for the quick retrieval of all messageIds associated with a correlationId. Index creation:

```
PUT correlation-to-message
{
  "settings": {
    "number_of_shards": 1,
    "number_of_replicas": 1
  },
  "mappings": {
    "properties": {
      "correlationId": { "type": "keyword" },
      "messageIds": { "type": "keyword" },
      "transactionId": { "type": "keyword" },
      "timestamp": { "type": "date" }
    }
  }
}

```

In this index, messageIds will be an array to store multiple messageIds associated with the same transaction.

4 Query to Retrieve All Logs of a Transaction

When an operator receives the correlationId and messageId of an error, they can directly access the transaction logs:

```
GET transactions-logs-4B1D9F02912746CCA7455FBDC5/_search
{
  "query": {
    "bool": {
      "should": [
        { "match": { "messageId": "1d6b8f37646440ee991f6e60b9cc73eb" } },
        { "match": { "correlationId": "4B1D9F02912746CCA7455FBDC5" } }
      ]
    }
  },
  "sort": [
    { "timestamp": { "order": "asc" } }
  ]
}
```

- The logs appear sorted by `timestamp`, ensuring a clear chronological reading.
- The `stepNumber` helps to interpret the sequence of events.

5 Creation of a Summarized View of the Transaction

To facilitate monitoring, a summary is generated with the key events of the transaction:

```
{
  "size": 0,
  "query": {
    "bool": {
      "must": [
        { "match": { "correlationId": "4B1D9F02912746CCA7455FBDC5" } }
      ]
    }
  },
  "aggs": {
    "event_types": {
      "terms": {
        "field": "eventType.keyword",
        "size": 10
      },
      "aggs": {
        "latest_log": {
          "top_hits": {
            "size": 1,
            "sort": [{ "timestamp": { "order": "desc" } }],
            "_source": ["timestamp", "eventType", "message"]
          }
        }
      }
    }
  }
}
```

- It allows a quick overview of the key events of the transaction.
- It facilitates navigation in OpenSearch Dashboards.

6 Generation of Real-Time Alerts

When a transaction fails (`logLevel: ERROR`), the operator receives an automatic alert.

```
POST _plugins/_alerting/monitors
{
  "name": "Transaction Error Monitor",
  "type": "monitor",
  "enabled": true,
  "schedule": {
    "period": { "interval": 1, "unit": "MINUTES" }
  },
  "inputs": [
    {
      "search": {
        "indices": ["transactions-logs-*"],
        "query": {
          "bool": {
            "must": [
              { "match": { "logLevel": "ERROR" } }
            ]
          }
        }
      }
    }
  ],
  "triggers": [
    {
      "name": "Error Trigger",
      "severity": "HIGH",
      "condition": {
        "script": {
          "source": "ctx.results[0].hits.total.value > 0"
        }
      }
```

```
    }
  ]
}
```

- It allows a quick overview of the key events of the transaction.
- It facilitates navigation in OpenSearch Dashboards.

7 Visualization in OpenSearch Dashboards

To facilitate the exploration of transactions in OpenSearch Dashboards:

- **Ordered table with the transaction logs**
  - Filtered by `correlationId`
  - Sorted by `timestamp`
  - Displays `logLevel`, `eventType`, `message`
- **Event Flow Chart**
  - X Axis → `timestamp`
  - Y Axis → `eventType`
  - Errors highlighted in red
- **Alerts Panel**
  - Displays recent errors
  - Allows clicking on the `correlationId` to open the detailed view.

8 Conclusion

- **Clear tracking of transactions.**
- **Optimization of queries and storage.**
- **Automatic alerts for errors.**
- **Intuitive visualization in OpenSearch Dashboards.**