# Elasticsearch - Week 6

Assignment Report

Siddharth Kumar

September 15, 2025

# Contents

# Chapter 1

# Task 1: Elasticsearch 3-Node Cluster Setup

## 1.1 Cluster Setup and Configuration

```
#node1,2,3
wget https://artifacts.elastic.co/downloads/elasticsearch/elasticsearch-9.1.3-linux-x86_64.tar.gz
wget https://artifacts.elastic.co/downloads/elasticsearch/elasticsearch-9.1.3-linux-x86_64.tar.gz.sha512
shasum -a 512 -c elasticsearch-9.1.3-linux-x86_64.tar.gz.sha512
tar -xzf elasticsearch-9.1.3-linux-x86_64.tar.gz
cd elasticsearch-9.1.3/

#node1
#/home/sre/elasticsearch-9.1.3/config/elasticsearch.yml
cluster.name: ppe_elasticsearch
node.name: node1
node.roles: [ master, data ]
network.host: 0.0.0.0
transport.host: 0.0.0.0
http.port: 9200
cluster.initial_master_nodes: ["node1"]

#shell
./bin/elasticsearch -d -p pid
./bin/elasticsearch-create-enrollment-token -s node
$ES_HOME/bin/elasticsearch-reset-password -u elastic
```

Listing 1.1: Elasticsearch node configuration

### Explanation

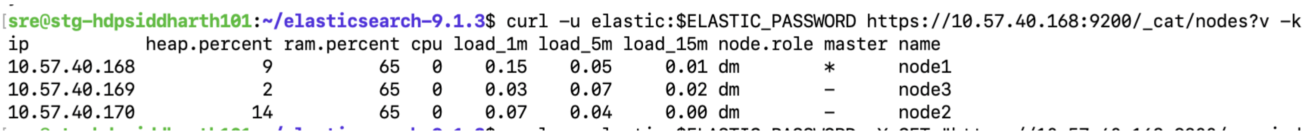
- Line 1: Indicates that these steps are to be repeated on node1, node2, and node3.
- Lines 2–3: Download the Elasticsearch 9.1.3 package and its checksum file from Elastic's repository.
- Line 5: Extract the Elasticsearch archive.
- Line 6: Navigate into the extracted installation directory.
- Lines 9–16: Edit `elasticsearch.yml`: set cluster name, unique node name, roles (master+data), bind network/transport to all interfaces, configure HTTP port, and define initial master node(s).
- Line 19: Start Elasticsearch as a background service and save its PID.
- Line 20: Generate an enrollment token so additional nodes can securely join the cluster.
- Line 21: Reset the built-in `elastic` user password for authentication.

```
#node2-3
#/home/sre/elasticsearch-9.1.3/config/elasticsearch.yml
cluster.name: ppe_elasticsearch
node.name: node1
node.roles: [ master, data ]

#shell
./bin/elasticsearch --enrollment-token <enrollment-token>
```

Listing 1.2: Elasticsearch node2-3 configuration

```
curl -u elastic:$ELASTIC_PASSWORD https://10.57.40.168:9200/
    _cat/nodes?v -k
```

Listing 1.3: API call to check cluster nodes



```
[sre@stg-hdpsiddharth101:~/elasticsearch-9.1.3$ curl -u elastic:$ELASTIC_PASSWORD https://10.57.40.168:9200/_cat/nodes?v -k
ip           heap.percent ram.percent cpu load_1m load_5m load_15m node.role master name
10.57.40.168            9          65   0    0.15    0.05     0.01 dm        *      node1
10.57.40.169            2          65   0    0.03    0.07     0.02 dm        -      node3
10.57.40.170           14          65   0    0.07    0.04     0.00 dm        -      node2
```

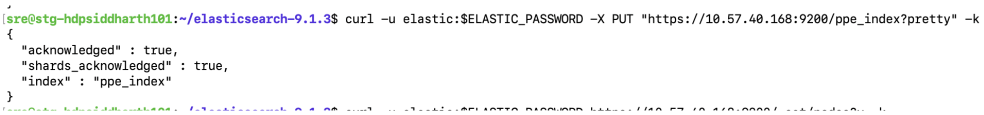
Figure 1.1: Cluster nodes

## 1.2 Sub Tasks Execution

### 1.2.1 Create an Index

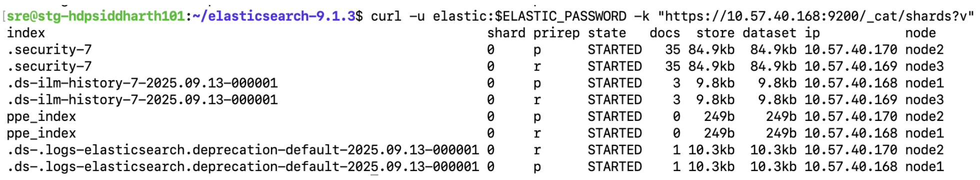
```
curl -u elastic:<password> -X PUT "https://10.57.40.168:9200/ppe_index?pretty" -k
```

Listing 1.4: Create index ppe_index



```
sre@stg-hdpsiddharth101:~/elasticsearch-9.1.3$ curl -u elastic:$ELASTIC_PASSWORD -X PUT "https://10.57.40.168:9200/ppe_index?pretty" -k
{
  "acknowledged" : true,
  "shards_acknowledged" : true,
  "index" : "ppe_index"
}
```

Figure 1.2: Index creation successfull



```
sre@stg-hdpsiddharth101:~/elasticsearch-9.1.3$ curl -u elastic:$ELASTIC_PASSWORD -k "https://10.57.40.168:9200/_cat/shards?v"
index                                                         shard prirep state   docs  store dataset ip           node
.security-7                                                   0     p      STARTED   35 84.9kb  84.9kb 10.57.40.170 node2
.security-7                                                   0     r      STARTED   35 84.9kb  84.9kb 10.57.40.169 node3
.ds-ilm-history-7-2025.09.13-000001                           0     p      STARTED    3  9.8kb   9.8kb 10.57.40.168 node1
.ds-ilm-history-7-2025.09.13-000001                           0     r      STARTED    3  9.8kb   9.8kb 10.57.40.169 node3
ppe_index                                                     0     p      STARTED    0   249b    249b 10.57.40.170 node2
ppe_index                                                     0     r      STARTED    0   249b    249b 10.57.40.168 node1
.ds-.logs-elasticsearch.deprecation-default-2025.09.13-000001 0     r      STARTED    1 10.3kb  10.3kb 10.57.40.170 node2
.ds-.logs-elasticsearch.deprecation-default-2025.09.13-000001 0     p      STARTED    1 10.3kb  10.3kb 10.57.40.168 node1
```

Figure 1.3: Default shards allocation

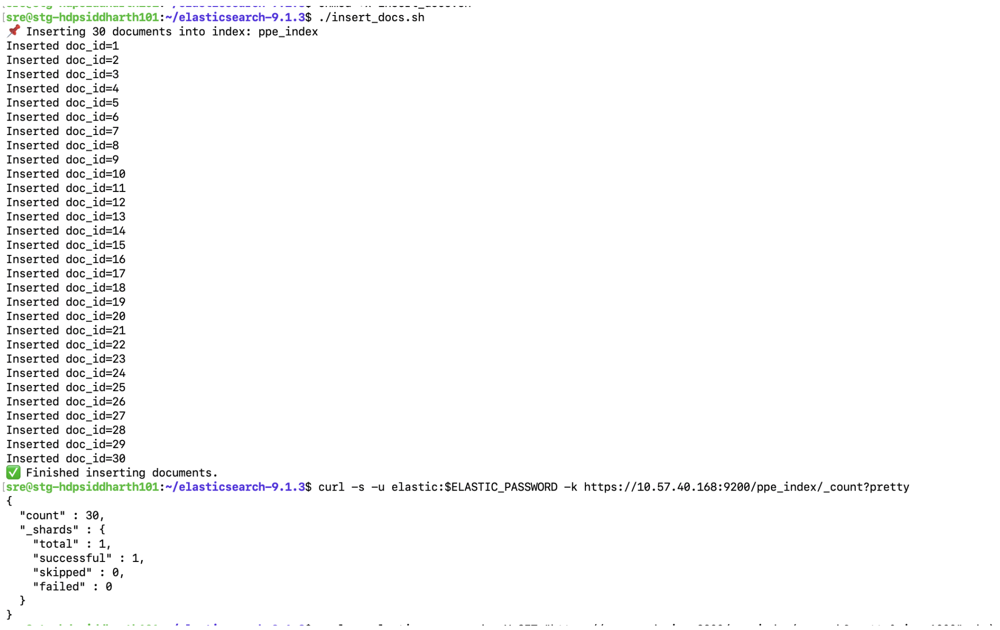
### 1.2.2 Insert Data into Index

```
1  #!/bin/bash
2
3  ES_HOST="https://10.57.40.168:9200"
4  ES_USER="elastic"
5  ES_PASS="4+fgoJIF16HbF4WxAFlR"
6  INDEX="ppe_index"
7
8  echo "Inserting 30 documents into index: $INDEX"
9
10 for i in {1..30}; do
11   curl -s -u $ES_USER:$ES_PASS -k -X POST "$ES_HOST/$INDEX/
       _doc?pretty" \
12        -H 'Content-Type: application/json' \
13        -d "{\"doc_id\": $i,
14             \"user\": { \"id\": \"siddharth\" },
15             \"message\": \"Document number $i\"
16            }" > /dev/null
17   echo "Inserted doc_id=$i"
18 done
19
20 echo "Finished inserting documents."
```

Listing 1.5: Insert documents script

**Explanation**

- Line 1: Declares the script as a Bash script using the shebang.
- Lines 3–6: Define variables for Elasticsearch host URL, username, password, and the index name (`ppe_index`).
- Line 8: Prints a message indicating that documents are about to be inserted.
- Line 10: Starts a loop from 1 to 30 to generate 30 documents.
- Line 11: Uses `curl` with authentication to send a POST request to Elasticsearch for each document. The `-k` flag allows insecure SSL connections.
- Line 12: Sets the request header to `Content-Type: application/json`.
- Lines 13–15: Define the JSON body of the document: each document has a unique `doc_id`, a user object with id `siddharth`, and a message field with the document number.
- Line 16: Redirects `curl` output to `/dev/null` to avoid cluttering the console.
- Line 17: Prints a confirmation message for each inserted document.
- Line 18: Ends the loop after inserting 30 documents.
- Line 20: Prints a final message once all documents have been inserted.



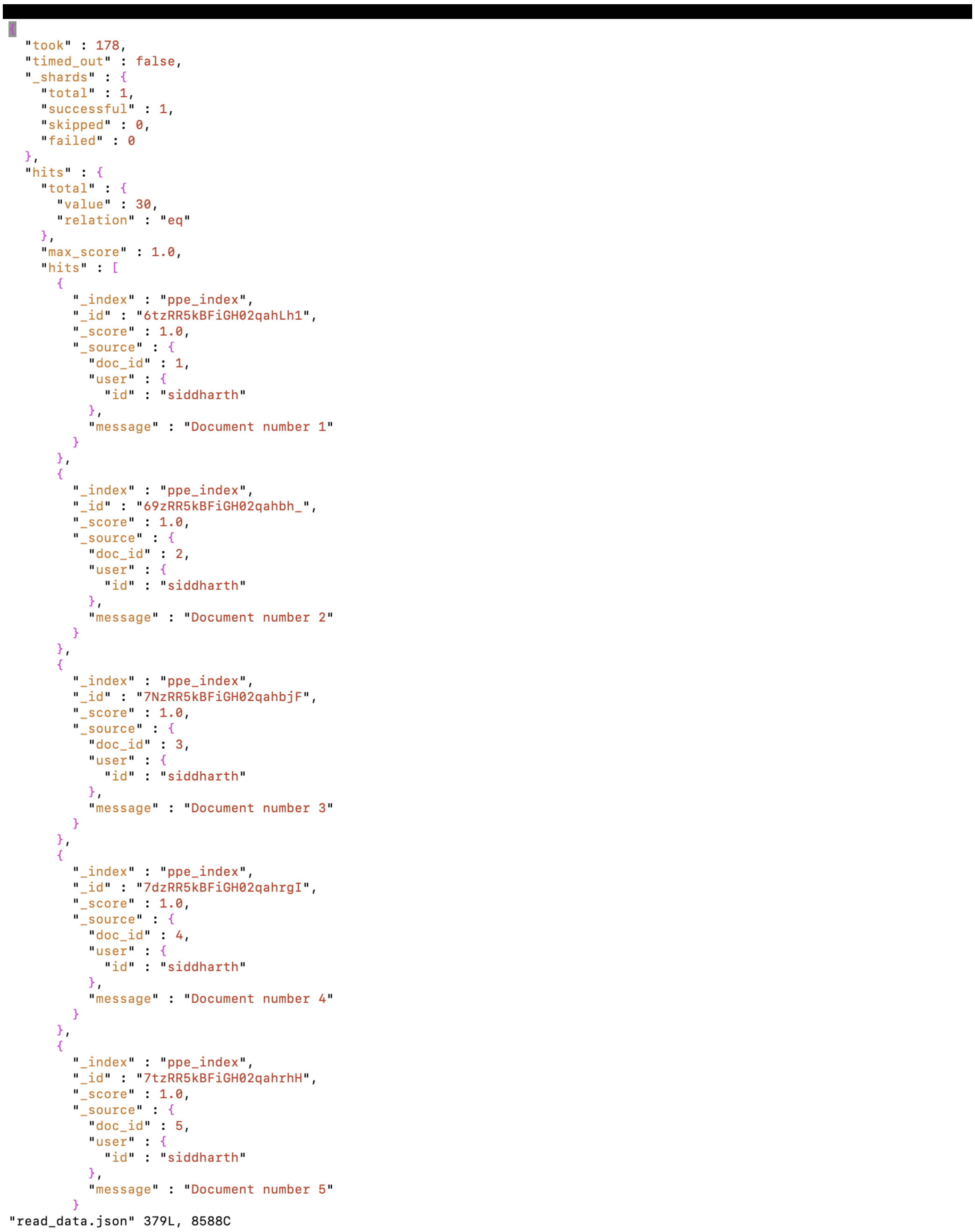
```
sre@stg-hdpsiddharth101:~/elasticsearch-9.1.3$ ./insert_docs.sh
📌 Inserting 30 documents into index: ppe_index
Inserted doc_id=1
Inserted doc_id=2
Inserted doc_id=3
Inserted doc_id=4
Inserted doc_id=5
Inserted doc_id=6
Inserted doc_id=7
Inserted doc_id=8
Inserted doc_id=9
Inserted doc_id=10
Inserted doc_id=11
Inserted doc_id=12
Inserted doc_id=13
Inserted doc_id=14
Inserted doc_id=15
Inserted doc_id=16
Inserted doc_id=17
Inserted doc_id=18
Inserted doc_id=19
Inserted doc_id=20
Inserted doc_id=21
Inserted doc_id=22
Inserted doc_id=23
Inserted doc_id=24
Inserted doc_id=25
Inserted doc_id=26
Inserted doc_id=27
Inserted doc_id=28
Inserted doc_id=29
Inserted doc_id=30
✅ Finished inserting documents.
sre@stg-hdpsiddharth101:~/elasticsearch-9.1.3$ curl -s -u elastic:$ELASTIC_PASSWORD -k https://10.57.40.168:9200/ppe_index/_count?pretty
{
  "count" : 30,
  "_shards" : {
    "total" : 1,
    "successful" : 1,
    "skipped" : 0,
    "failed" : 0
  }
}
```

Figure 1.4: Script inserting docs

### 1.2.3 Read Data and Export to JSON

```
curl -u elastic:<password> -X GET "https://10.57.40.168:9200/ppe_index/_search?pretty&size=1000" -k \
  -H ’Content-Type: application/json’ > read_data.json
```

Listing 1.6: Read data and write to JSON



```
{
  "took" : 178,
  "timed_out" : false,
  "_shards" : {
    "total" : 1,
    "successful" : 1,
    "skipped" : 0,
    "failed" : 0
  },
  "hits" : {
    "total" : {
      "value" : 30,
      "relation" : "eq"
    },
    "max_score" : 1.0,
    "hits" : [
      {
        "_index" : "ppe_index",
        "_id" : "6tzRR5kBFiGH02qahLh1",
        "_score" : 1.0,
        "_source" : {
          "doc_id" : 1,
          "user" : {
            "id" : "siddharth"
          },
          "message" : "Document number 1"
        }
      },
      {
        "_index" : "ppe_index",
        "_id" : "69zRR5kBFiGH02qahbh_",
        "_score" : 1.0,
        "_source" : {
          "doc_id" : 2,
          "user" : {
            "id" : "siddharth"
          },
          "message" : "Document number 2"
        }
      },
      {
        "_index" : "ppe_index",
        "_id" : "7NzRR5kBFiGH02qahbjF",
        "_score" : 1.0,
        "_source" : {
          "doc_id" : 3,
          "user" : {
            "id" : "siddharth"
          },
          "message" : "Document number 3"
        }
      },
      {
        "_index" : "ppe_index",
        "_id" : "7dzRR5kBFiGH02qahrgI",
        "_score" : 1.0,
        "_source" : {
          "doc_id" : 4,
          "user" : {
            "id" : "siddharth"
          },
          "message" : "Document number 4"
        }
      },
      {
        "_index" : "ppe_index",
        "_id" : "7tzRR5kBFiGH02qahrhH",
        "_score" : 1.0,
        "_source" : {
          "doc_id" : 5,
          "user" : {
            "id" : "siddharth"
          },
          "message" : "Document number 5"
        }
"read_data.json" 379L, 8588C
```

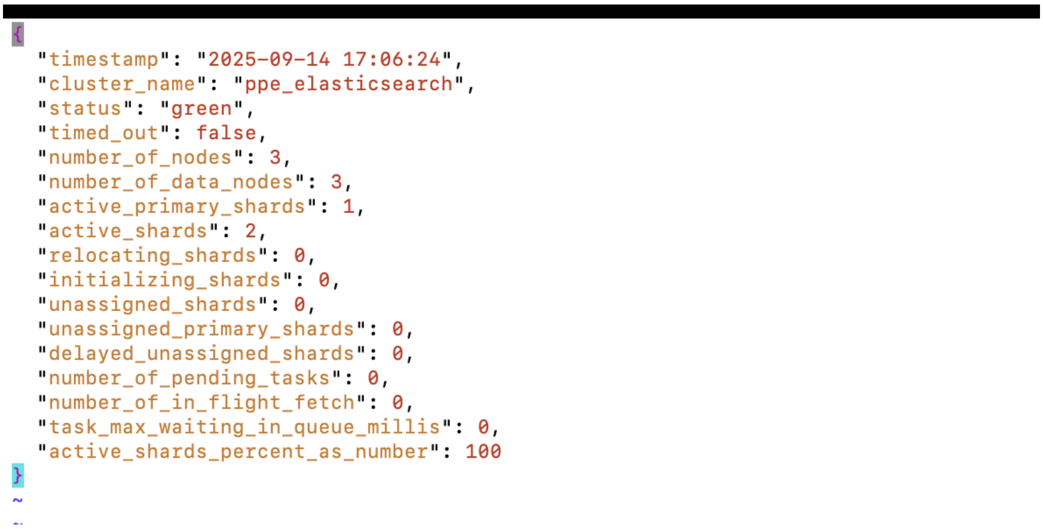
Figure 1.5: read_data.json

### 1.2.4 Capture Cluster Health (Before Node Shutdown)

```bash
1  #!/bin/bash
2
3  ES_HOST="https://10.57.40.168:9200"
4  ES_USER="elastic"
5  ES_PASS="4+fgoJIF16HbF4WxAFlR"
6  OUTPUT_FILE="cluster_health_before_step5.json"
7
8  # Get current timestamp
9  TIMESTAMP=$(date +"%Y-%m-%d %H:%M:%S")
10
11 # Capture cluster health and add timestamp
12 curl -s -u $ES_USER:$ES_PASS -k "$ES_HOST/_cluster/health/
       ppe_index?pretty" | \
13   jq --arg ts "$TIMESTAMP" '{timestamp: $ts} + .' >
         $OUTPUT_FILE
14
15 echo "Cluster health saved to $OUTPUT\_FILE with timestamp:
       $TIMESTAMP"
```

Listing 1.7: Script to capture cluster health

**Explanation**
- Line 1: Declares the script as a Bash script with the shebang.
- Lines 3–5: Define variables for Elasticsearch host URL, username, and password.
- Line 6: Sets the output filename as `cluster_health_before_step5.json`.
- Line 9: Captures the current timestamp in `YYYY-MM-DD HH:MM:SS` format using the `date` command.
- Lines 12–13: Sends a request to Elasticsearch's `_cluster/health` API for the index `ppe_index`, using curl with authentication.
• The `-k` flag allows insecure SSL connections.
• The response is piped to `jq`, which injects the captured timestamp as a new JSON field.
• The result is written to the output file defined earlier.
- Line 15: Prints a confirmation message showing the filename and timestamp of the saved cluster health data.



```
{
  "timestamp": "2025-09-14 17:06:24",
  "cluster_name": "ppe_elasticsearch",
  "status": "green",
  "timed_out": false,
  "number_of_nodes": 3,
  "number_of_data_nodes": 3,
  "active_primary_shards": 1,
  "active_shards": 2,
  "relocating_shards": 0,
  "initializing_shards": 0,
  "unassigned_shards": 0,
  "unassigned_primary_shards": 0,
  "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": 100
}
~
~
```

Figure 1.6: Cluster Health Before Node Shutdown

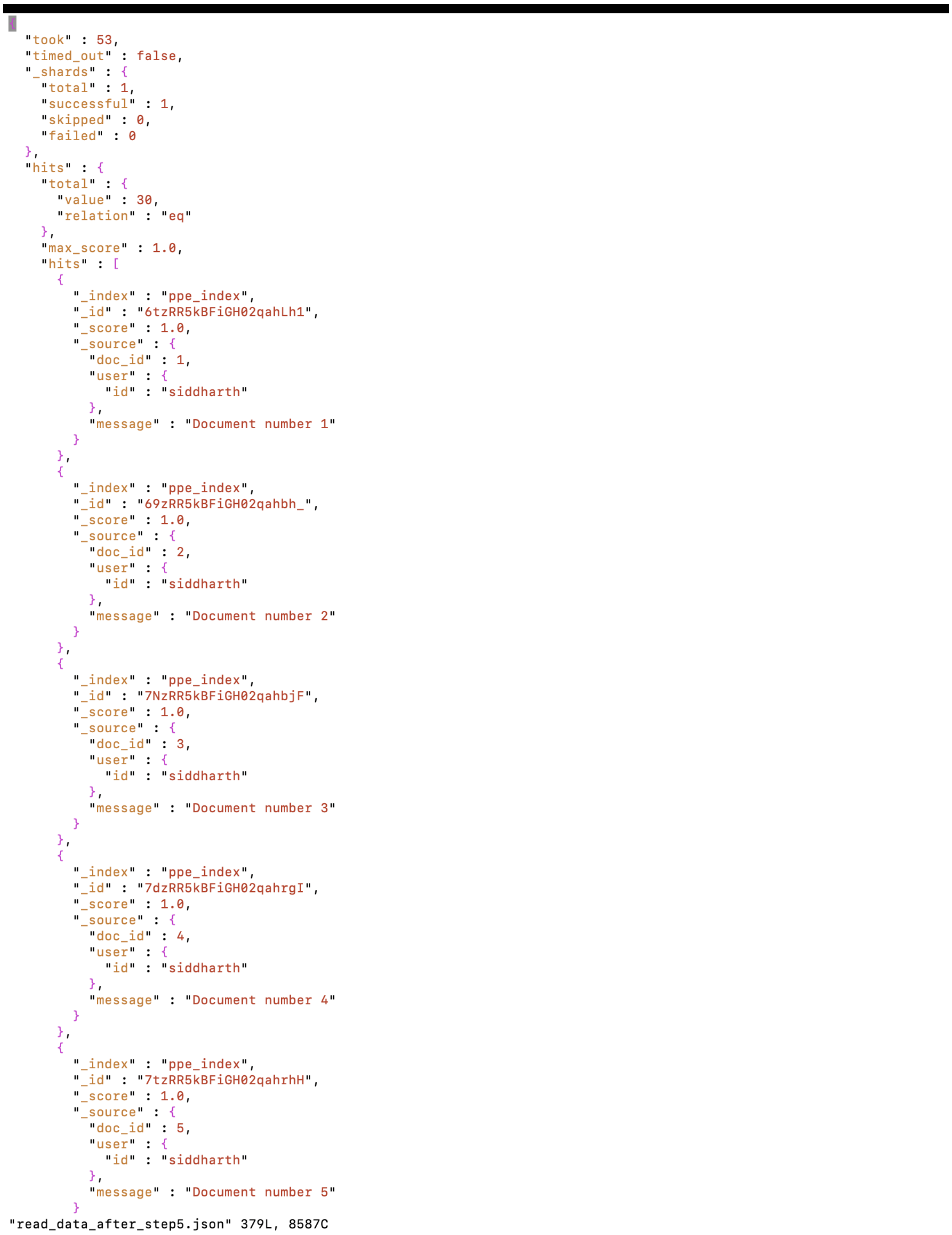
### 1.2.5 Stop Two Nodes and Verify Data Availability

```
#node2-3
pkill -f elasticsearch
```

Listing 1.8: Stop nodes and verify data

**Explanation**

After stopping two nodes, quorum fails, no master available. But read is working fine from replica on node1.



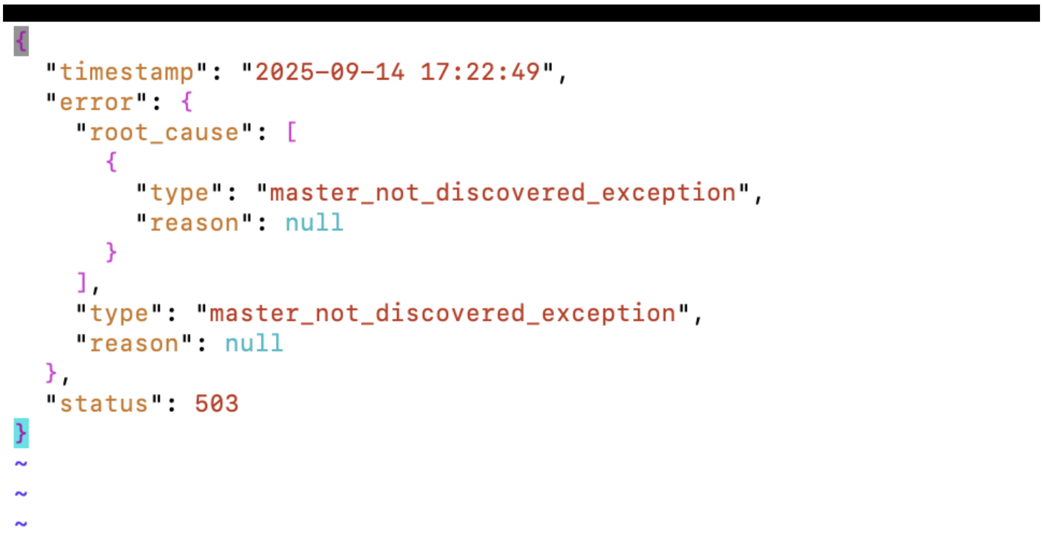
```
{
  "took" : 53,
  "timed_out" : false,
  "_shards" : {
    "total" : 1,
    "successful" : 1,
    "skipped" : 0,
    "failed" : 0
  },
  "hits" : {
    "total" : {
      "value" : 30,
      "relation" : "eq"
    },
    "max_score" : 1.0,
    "hits" : [
      {
        "_index" : "ppe_index",
        "_id" : "6tzRR5kBFiGH02qahLh1",
        "_score" : 1.0,
        "_source" : {
          "doc_id" : 1,
          "user" : {
            "id" : "siddharth"
          },
          "message" : "Document number 1"
        }
      },
      {
        "_index" : "ppe_index",
        "_id" : "69zRR5kBFiGH02qahbh_",
        "_score" : 1.0,
        "_source" : {
          "doc_id" : 2,
          "user" : {
            "id" : "siddharth"
          },
          "message" : "Document number 2"
        }
      },
      {
        "_index" : "ppe_index",
        "_id" : "7NzRR5kBFiGH02qahbjF",
        "_score" : 1.0,
        "_source" : {
          "doc_id" : 3,
          "user" : {
            "id" : "siddharth"
          },
          "message" : "Document number 3"
        }
      },
      {
        "_index" : "ppe_index",
        "_id" : "7dzRR5kBFiGH02qahrgI",
        "_score" : 1.0,
        "_source" : {
          "doc_id" : 4,
          "user" : {
            "id" : "siddharth"
          },
          "message" : "Document number 4"
        }
      },
      {
        "_index" : "ppe_index",
        "_id" : "7tzRR5kBFiGH02qahrhH",
        "_score" : 1.0,
        "_source" : {
          "doc_id" : 5,
          "user" : {
            "id" : "siddharth"
          },
          "message" : "Document number 5"
        }
"read_data_after_step5.json" 379L, 8587C
```

Figure 1.7: read_data_after_step5.json

### 1.2.6 Capture Cluster Health (After Node Shutdown)

**Explanation**

_cluster/health API not working because no master is available for that.Hence we are not able to get status of cluster.



```
{
  "timestamp": "2025-09-14 17:22:49",
  "error": {
    "root_cause": [
      {
        "type": "master_not_discovered_exception",
        "reason": null
      }
    ],
    "type": "master_not_discovered_exception",
    "reason": null
  },
  "status": 503
}
~
~
~
```

Figure 1.8: Cluster Health after Node Shutdown

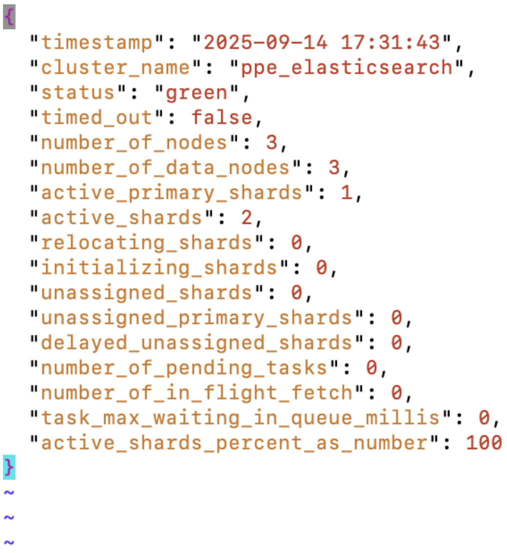
### 1.2.7 Restart Nodes and Capture Cluster Health

```
#on node2 and node 3
$ES_HOME/bin/elasticsearch
```

Listing 1.9: Restart nodes

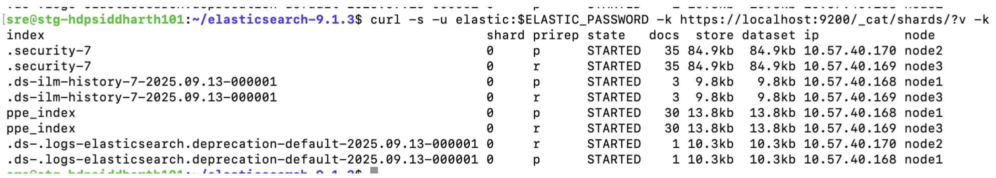
**Explanation**

After restarting, qorum is formed and master is elected. replica on node 1 is upgraded to primary. and it's repilca is made on node3.



```
{
  "timestamp": "2025-09-14 17:31:43",
  "cluster_name": "ppe_elasticsearch",
  "status": "green",
  "timed_out": false,
  "number_of_nodes": 3,
  "number_of_data_nodes": 3,
  "active_primary_shards": 1,
  "active_shards": 2,
  "relocating_shards": 0,
  "initializing_shards": 0,
  "unassigned_shards": 0,
  "unassigned_primary_shards": 0,
  "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": 100
}
~
~
~
```

Figure 1.9: Cluster Health after restarting Nodes



```
sre@stg-hdpsiddharth101:~/elasticsearch-9.1.3$ curl -s -u elastic:$ELASTIC_PASSWORD -k https://localhost:9200/_cat/shards/?v -k
index                                                         shard prirep state   docs  store dataset ip           node
.security-7                                                   0     p      STARTED   35 84.9kb  84.9kb 10.57.40.170 node2
.security-7                                                   0     r      STARTED   35 84.9kb  84.9kb 10.57.40.169 node3
.ds-ilm-history-7-2025.09.13-000001                           0     p      STARTED    3  9.8kb   9.8kb 10.57.40.168 node1
.ds-ilm-history-7-2025.09.13-000001                           0     r      STARTED    3  9.8kb   9.8kb 10.57.40.169 node3
ppe_index                                                     0     p      STARTED   30 13.8kb  13.8kb 10.57.40.168 node1
ppe_index                                                     0     r      STARTED   30 13.8kb  13.8kb 10.57.40.169 node3
.ds-.logs-elasticsearch.deprecation-default-2025.09.13-000001 0     r      STARTED    1 10.3kb  10.3kb 10.57.40.170 node2
.ds-.logs-elasticsearch.deprecation-default-2025.09.13-000001 0     p      STARTED    1 10.3kb  10.3kb 10.57.40.168 node1
```

Figure 1.10: Shards allocation after restarting Nodes

# Chapter 2

# Task 2: Monitoring and Alerting

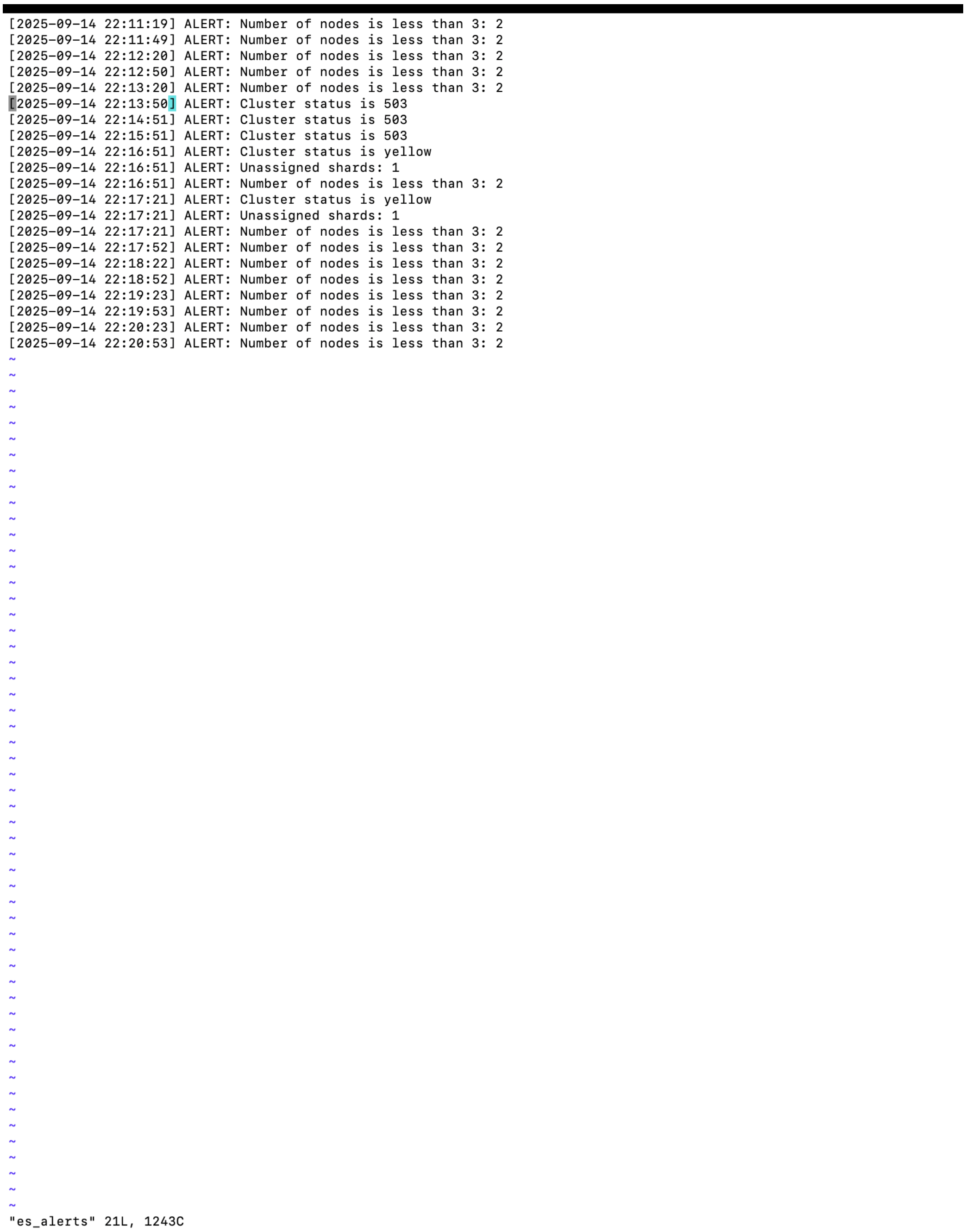
## 2.1 Alerting Example

```bash
#!/bin/bash

ALERT_FILE="/home/sre/elasticsearch-9.1.3/es_alerts"
ES_URL="https://localhost:9200"
USER="elastic"
PASS="4+fgoJIF16HbF4WxAFlR"

while true; do
    TIMESTAMP=$(date '+%Y-%m-%d %H:%M:%S')

    RESPONSE=$(curl -s -k -u "$USER:$PASS" "$ES_URL/_cluster/health" 2>/dev/null)
    CURL_EXIT=$?

    if [ $CURL_EXIT -ne 0 ] || [ -z "$RESPONSE" ]; then
        echo "[$TIMESTAMP] ALERT: Cannot reach cluster or no master elected!" >> "$ALERT_FILE"
    else
        STATUS=$(echo "$RESPONSE" | jq -r '.status // "null"')
        UNASSIGNED=$(echo "$RESPONSE" | jq -r '.unassigned_shards // 0')
        NODES=$(echo "$RESPONSE" | jq -r '.number_of_nodes // 0')

        # Convert "null" to safe integers
        [ "$UNASSIGNED" = "null" ] && UNASSIGNED=0
        [ "$NODES" = "null" ] && NODES=0
```

```
24
25         if [ "$STATUS" != "green" ]; then
26             echo "[$TIMESTAMP] ALERT: Cluster status is $STATUS" >> "$ALERT_FILE"
27         fi
28
29         if [ "$UNASSIGNED" -gt 0 ]; then
30             echo "[$TIMESTAMP] ALERT: Unassigned shards: $UNASSIGNED" >> "$ALERT_FILE"
31         fi
32
33         if [ "$NODES" -lt 3 ]; then
34             echo "[$TIMESTAMP] ALERT: Number of nodes is less than 3: $NODES" >> "$ALERT_FILE"
35         fi
36     fi
37
38     sleep 30
39 done
```

Listing 2.1: Script for alert

**Explanation**
- Lines 3–6: Define variables for alert log file path, Elasticsearch URL, and authentication credentials.
- Line 8: Begins an infinite loop that continuously checks the cluster.
- Line 9: Captures the current timestamp in `YYYY-MM-DD HH:MM:SS` format.
- Line 11: Uses `curl` to query the cluster health endpoint, saving the response into `RESPONSE`. Errors are redirected to `/dev/null`.
- Line 12: Stores curl's exit code in `CURL_EXIT` for connectivity checks.
- Lines 14–15: If curl failed or the response is empty, logs an alert indicating the cluster is unreachable or no master is elected.
- Lines 17–19: Parse JSON response with `jq` to extract `status`, number of `unassigned_shards`, and `number_of_nodes`.
- Lines 22–23: Replace any `null` values with safe integers (0).
- Lines 25–27: If the cluster status is not `green`, log an alert with the current status.
- Lines 29–31: If there are unassigned shards, log an alert with their count.
- Lines 33–35: If the number of nodes is less than 3, log an alert showing the current node count.
- Line 38: Waits 30 seconds before the next check.
- Line 39: Ends the infinite loop, ensuring continuous monitoring.



```
[2025-09-14 22:11:19] ALERT: Number of nodes is less than 3: 2
[2025-09-14 22:11:49] ALERT: Number of nodes is less than 3: 2
[2025-09-14 22:12:20] ALERT: Number of nodes is less than 3: 2
[2025-09-14 22:12:50] ALERT: Number of nodes is less than 3: 2
[2025-09-14 22:13:20] ALERT: Number of nodes is less than 3: 2
[2025-09-14 22:13:50] ALERT: Cluster status is 503
[2025-09-14 22:14:51] ALERT: Cluster status is 503
[2025-09-14 22:15:51] ALERT: Cluster status is 503
[2025-09-14 22:16:51] ALERT: Cluster status is yellow
[2025-09-14 22:16:51] ALERT: Unassigned shards: 1
[2025-09-14 22:16:51] ALERT: Number of nodes is less than 3: 2
[2025-09-14 22:17:21] ALERT: Cluster status is yellow
[2025-09-14 22:17:21] ALERT: Unassigned shards: 1
[2025-09-14 22:17:21] ALERT: Number of nodes is less than 3: 2
[2025-09-14 22:17:52] ALERT: Number of nodes is less than 3: 2
[2025-09-14 22:18:22] ALERT: Number of nodes is less than 3: 2
[2025-09-14 22:18:52] ALERT: Number of nodes is less than 3: 2
[2025-09-14 22:19:23] ALERT: Number of nodes is less than 3: 2
[2025-09-14 22:19:53] ALERT: Number of nodes is less than 3: 2
[2025-09-14 22:20:23] ALERT: Number of nodes is less than 3: 2
[2025-09-14 22:20:53] ALERT: Number of nodes is less than 3: 2
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
"es_alerts" 21L, 1243C
```

Figure 2.1: Alert logs stored in es_alerts file