Elasticsearch Metrics Dashboard

Version No: 1.0

Date: 20-APR-2020

|  |  |
| --- | --- |
| Prepared by | Prashanth Kartan |
| Reviewed by |  |
| Approved by: |  |

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| Version No | Date | Prepared by / Modified by | Significant Changes |
| 1.0 | 20-04-2020 | Prashanth Kartan | Initial Version |

Table of Contents

[1. Objective 4](#_Toc39694008)

[2. Key benefits of dashboard 4](#_Toc39694009)

[3. Most useful Elasticsearch metrics 4](#_Toc39694010)

[4 Elasticsearch metrics APIs 5](#_Toc39694011)

[5 Dashboard design 7](#_Toc39694012)

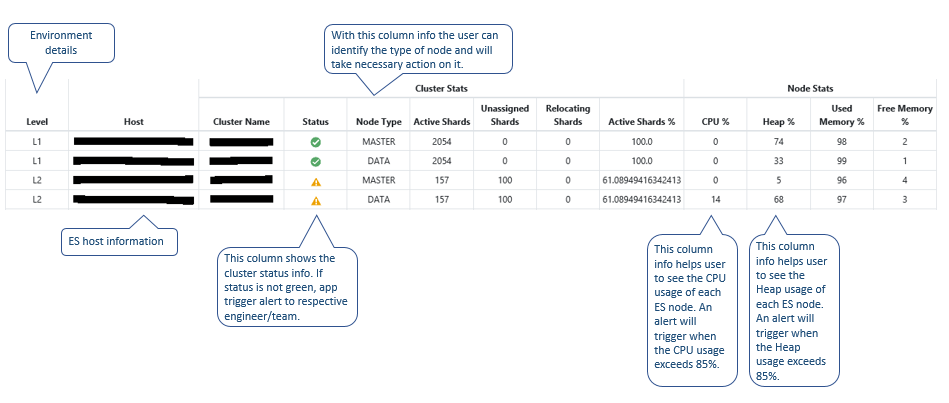
[6 Hardware and software requirements 7](#_Toc39694013)

[7 References 7](#_Toc39694014)

# 1. Objective

Elasticsearch metrics dashboard enables us to gather metrics data from different environments and display those key metrics in a single place, and then take appropriate action based on information.

The below is the sample dashboard look:



# 2. Key benefits of dashboard

This metric dashboard is a great way to glean insights from metrics data without having to code personally and we can able to access the most up-to date information of all environments at our fingertips. Some of the key benefits of the dashboard are:

**Unified view of key metrics:** This dashboard provides a single pane visibility into all key Elasticsearch metrics and serve as a central location for users to view and analyze up to date information.

**Real time updates:** Any change in the underlying data or values is reflected in dashboard immediately and trigger alerts of potential problems or conditions detected within the system. When a certain pre-defined threshold is met, an alert will be generated and notify to the team. The content of an alert is also limited, describes only the system that is affected by the failure.

**Saves time and effort:** We can save time with no longer to get metrics for individual Elasticsearch node and prepare reports of Elasticsearch performance in each environment level. This dashboard provides with all the key values without having external efforts or analyze information separately.

**Productivity improvements:** This dashboard allow us to measure performance numerically, particularly when the node is in yellow or red state, an alert is triggered to the team and no longer to put manual effort, we can visualize the problem in the dashboard itself. Similarly, when a node is down or not responding, this system will trigger a critical alert to the team continuously for every 15 minutes until it gets resolved in backend.

**Improves risk identification:** The ability to identify risk before it becomes a major issue and it is critical to business. For instance – whenever a node CPU or JVM heap usage exceeds 80% our dashboard system triggers the warning alert to the team. So that we can mitigate the risk early on.

# 3. Most useful Elasticsearch metrics

**Cluster Health – Nodes and Shards:** The cluster health status is a basic metric for Elasticsearch**.** It provides an overview of running nodes and the status of shards distributed to the nodes.

**Node Performance – CPU:**  The performance of the Elasticsearch is strongly depends on the machine CPU, memory usage and disk i/o are the basic operating system metrics for each Elasticsearch node.

**Node Performance – Memory Usage:** Elasticsearch runs inside the Java Virtual Machine, JVM memory and garbage collection are the areas to look at for Elasticsearch-specific memory utilization.

**Node Performance – Disk I/O:** As Elasticsearch engines makes heavy use of storage devices and observing the disk I/O ensures that the basic needs gets fulfilled. It is a good metric to check the effectiveness of indexing and query performance.

**Java – Heap Usage and Garbage Collection:** Elasticsearch runs inside the JVM, so the optimal settings of JVM and monitoring of the garbage collector and memory usage are critical. There are several things to consider with regards to JVM and operating system memory.

# 4 Elasticsearch metrics APIs

Elasticsearch makes it easy to interact with clusters thru REST API. We can easily index documents, update our cluster settings and submit queries on the fly.

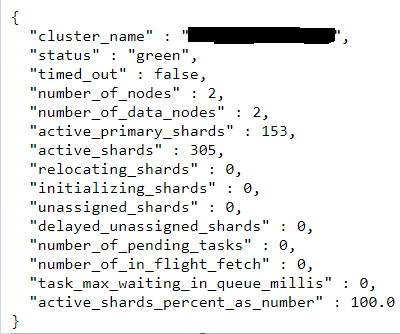
By default, Elasticsearch runs on 9200 port and expose the metrics on the same and return response about the indices, nodes and clusters in the JSON format. The below are the main http APIs that we can use to monitor the Elasticsearch.

* Cluster Health API
* Cluster Stats API
* Node Stats API
* Index Stats API
* Pending Tasks API

**Cluster Health API**

This API exposes key information about the health of the cluster in a JSON response.

<http://localhost:9200/_cluster/health?pretty>



This above endpoint results an overview of the cluster name, cluster status, shard status (number of active, initializing, and unassigned shards) and active shards in percent. Elasticsearch provides three types of cluster status i.e., green, yellow and red.

* Green indicates everything is good and no need to take any action.
* Yellow indicates the shards are initializing and the shards are not 100% active.
* Red indicates one are more primary shards have not been assigned meaning that data is missing, and search results will not be complete.

**Cluster Stats API**

This API provides cluster-wide information, basically it adds together all the stats from each node in cluster. It does not provide the level of detail that the Node Stats API offers, but it is useful for getting a general idea of how our cluster is doing. The below command exposes the stats of the cluster.

<http://localhost:9200/_cluster/stats?pretty>

The output response provides high-level information like cluster status, basic metrics about your indices (number of indices, shard and document count, field data cache usage) and basic statistics about the nodes in your cluster (number of nodes by type, file descriptors, memory usage).

**Node Stats API**

The cluster nodes stats API is used to retrieve statistics for the nodes in a cluster. By default, all the stats are returned but we can customize the returned information by using metrics.

The below command is used to get cluster nodes stats,

<http://localhost:9200/_nodes/node1,node2/stats?pretty>

We can also limit our query to one or more categories of stats by adding them at the end of the command with comma-separated form:

<http://localhost:9200/_nodes/datanode1/stats/jvm,os?pretty>

The resulting output provides information limited to datanode1’s JVM and OS metrics.

**Index Stats API**

Index Stats API helps to check on stats pertaining to one index by substituting index\_name with the actual name of the index.

<http://localhost:9200/index_name/_stats?pretty>

By using the above command, we can able to access metrics about indexing performance, search performance, merging activity, segment count, size of the field-data cache, and number of evictions from the field-data cache. These metrics are provided on two levels: aggregated across all shards in the index and limited to just the index’s primary shards.

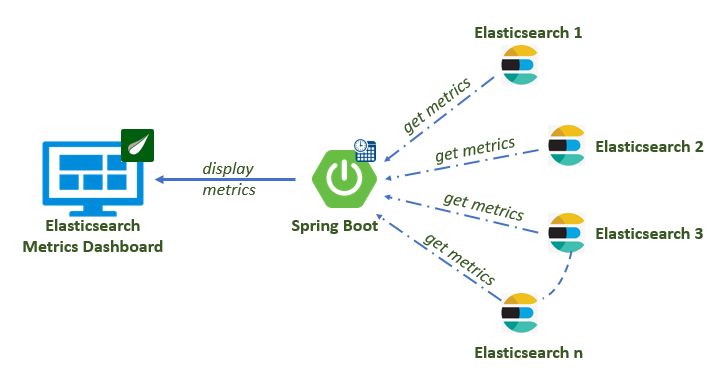
**Pending Tasks API**

This API is a quick way to look at our cluster’s pending tasks in more detail. Pending tasks are tasks that only the master node can perform, like creating new indices or redistributing shards around the cluster. If the master node is unable to keep up with the rate of these requests, pending tasks will begin to queue and you will see this number rise. To query pending tasks, find the below command:

<http://localhost:9200/_cluster/pending_tasks?pretty>

The pending cluster tasks API returns a list of any cluster-level changes (e.g. create index, update mapping, allocate or fail shard) which have not yet been executed.

# 5 Dashboard design



# 6 Hardware and software requirements

**Operating System:** Linux/Windows

**RAM:** 1 GB

**Programming language:** Java 1.8

**Framework:** Spring Boot

**Other APIs:** Elasticsearch Java API, Apache Commons, Json API, Json-simple, Jackson-databind.

# 7 References

<https://www.elastic.co/>

<https://dzone.com/>

<https://www.datadoghq.com/>