15mile TECHNOLOGIES

ELASTICSEARCH MONITORING

Table of Contents

Introduction to Elasticsearch

What is Elasticsearch?

Modern software application has Search and Analytics as critical features. Many applications, such as mobile apps, web, and data analytics, demand scalability and the capability to handle large volumes of data in near real-time. Today, autocomplete in text fields, location search, search suggestions, and faceted navigation are normal in usability. Elasticsearch is an open-source text search engine based on Lucene, which Shay Bannon initially published in 2010. It is based on a " shared nothing architecture." It has essential features like easy scalability, near real-time search, and aggregations (facets) which are paired with developer-friendly APIs and client libraries for different programming languages. Its features made Elasticsearch widely used for the integration and development of search and analytics functions, and DevOps uses, like log searching.

Particular data types, such as IP Addresses and Geo-Shapes, support various possible applications. Distributed systems are usually complex, but Elasticsearch makes many automatic decisions and provides a straightforward API for the client processes. As a result, scaling Elasticsearch is much easier than with many other systems, even though large Elasticsearch clusters have their issues which often require Elasticsearch expertise. For Elasticsearch optimizations, every application has different requirement profiles:

	E-Commerce Systems	Centralized Logging	Geo-Location Apps
Indexing	Low	High	Medium
Search	High	Low	High
Capacity	Low	High	Low
Geo-Queries	Medium	Low	High

Performance Monitoring

Before going into the details of Elasticsearch, performance metrics, and cluster setups, we need to provide a methodology to put the information from this E-Book into practice. In this engineering discipline, we have development, quality assurance, and operations - done in iterative ways for all components. For all phases, you need to measure metrics to satisfy given specifications or verify assumptions made by the engineers or their bosses. In this industry, the specifications usually need to be more precise or change rapidly. But in any case, failures in backend services must not reach end users – that's why DevOps teams want to know the capabilities and the current state of the systems they work with.

Where is the need of detailed application performance monitoring?

Development and POCs

- Find out how performance is affected by the use case and implementation. The result available might provide better algorithms or new concepts in the application development or a better setup for the given use case.

QA - Benchmark and stress tests

- See how the setup does under load. The result might help in the tuning of specific settings, infrastructure changes or optimized algorithms.

Production clusters operations

- Check the reliability of the current setup. It might be helpful in detecting bottlenecks, planning for new resources, and doing root cause analysis.
- Proactive monitoring and taking actions based on the results is a good protection against system failures as they happen.

To provide a complete picture on this often underestimated issue,

We are providing a few more hints to make these procedures effective:

Monitoring and logging should be independent of the production cluster

It is seen that logs & metrics generate high data volumes continuously at a high rate. Suppose the system is in a critical state. In that case, it can lead to system overload or data loss – can you imagine what happens when so many processes log a considerable number of errors to an unhealthy cluster?

Correlate Logs, Events and Metrics

Metrics tell you that something happened, the evidence of which is usually available in logs and system events.

Monitor surrounding systems

The good thing will be to cover the entire application stack, including OS and network metrics. For example, Elasticsearch integrates well with Hadoop and is frequently used with reverse proxy servers. If any of the components causes trouble, only a side effect could be visible with part monitoring coverage.

Reports sharing in the organization

Communication is much easier when everybody on the team has quick access to the facts: before & after the testing, save the results of measurements, take screenshots or keep links to the graphs, and collate it to the documentation.

Rely on anomaly detection and intelligent alerts

You can save a lot of time that would otherwise go into watching dashboards. There are simple threshold-based alerts, but they don't work well in operational systems where the load is varying & triggering alerts each time a limit is crossed. Good monitoring solutions have ML algorithms, and they allow setting adjustments.

Integrate alerting and workflow

A typical use case involves forwarding the alerts to the incident management platforms, which route them to the suitable support desk.

ISmile Technologies provides an excellent way to monitor Elasticsearch.

The setup, tuning, and operations of Elasticsearch require deep insights about the performance metrics like index rate, query rate, merge times, query latency, and many more. Open-source tools for monitoring Elasticsearch are free, but they could be time consuming without expert help. ISmile Technologies aims to save you time & effort.

Request a Free Consultation

Key Metrics to Monitor Elasticsearch

In this section, we will look at the capabilities & potential use cases of Elasticsearch. We will also discuss what it takes to monitor an Elasticsearch cluster. And we'll also identify key metrics that should be monitored and find out why they're essential in maintaining the health and performance of the Elasticsearch cluster.

Elasticsearch Use Cases

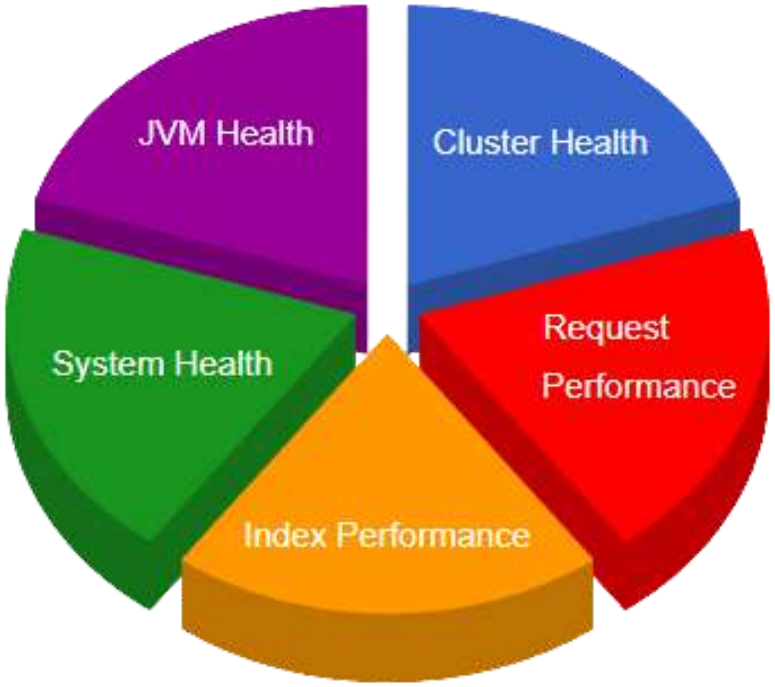
Elasticsearch was launched in 2010 and is a search engine based on the open-source Apache Lucene library. It has been developed in Java and has supported clients in many different languages, such as Python, PHP, C#, and Ruby. It is currently the most popular search engine available today. Being a distributed tool, Elasticsearch is very scalable and provides near real-time search features. All these features add up to a tool that supports many critical business functions and use cases. Enterprises have used Elasticsearch to do the following:

- Log Collection and Aggregation
- To gather & index large datasets
- Collect & manage event data & metrics
- To analyze Logs, Large Datasets, and Metric Data
- For Full Text and Document Search Features

Importance of Monitoring your Elasticsearch Cluster

Elasticsearch is versatile, scalable, and valuable. And the infrastructure that hosts your cluster must satisfy its requirements. Also, the cluster must be sized correctly to support its data store and the volume of requests. If the infrastructure is improperly sized and there are misconfigurations, it can result in many problems like sluggish performance, and the entire cluster can become unresponsive and crash.

The five areas of concern are:



By appropriately monitoring your cluster, you can ensure that it is appropriately sized and handles all data requests efficiently. Now we will look at the cluster from five different perspectives and show critical metrics to monitor from each perspective. We will also discuss what potential problems can be averted by monitoring these metrics.

Cluster Health: Shards and Nodes

An Elasticsearch cluster can have one or more nodes. A node is a cluster's member hosted on an individual server. Clusters can be scaled horizontally by adding additional nodes. Indexes are used for organizing the data within the cluster. An index is nothing but a collection of documents that have similar characteristics.

Let's take the example of an Elasticsearch cluster deployed to store an application's log entries. You might set up an index to collect log entries for a day. Each log entry is nothing but a document that has log contents & associated metadata.

If you consider a large dataset, the index's size might exceed the storage capacity on a single node. We must also ensure that we have created redundant copies of our index if something terrible happens to our node. Elasticsearch takes care of this by dividing the index into a defined number of Shards. Elasticsearch then distributes all the shards across different nodes in the cluster. As a default, an Elasticsearch index is made of five shards with one replica. So because of this default configuration, we have an index divided into five shards, each having a single replica stored on a different node.

IMPORTANT METRICS FOR CLUSTER HEALTH

Status	The cluster status: • Red: No shards have been allocated. • Yellow: Only the primary shards have been allocated. • Green: All shards have been allocated.
Nodes	This metric has the total number of nodes in a cluster, and shows the count of successful and failed nodes.
Count of Active Shards	The count of active shards within the cluster.
Relocating Shards	The number of shards being moved due to the loss of a node.
Initializing Shards	The number of shards being initialized because of the addition of an index.
Unassigned Shards	The number of shards for which replicas have not been created or assigned yet.

You must decide the correct number of shards for an index, as too few shards may negatively affect the search efficiency and data distribution across the nodes. Conversely, if there are too many nodes, it can create an excessive demand on the cluster resources for their management.
While doing cluster monitoring, one can query the cluster health endpoint and get information about the cluster status, the number of nodes, and the active shards count. One can also see the number of relocating shards, initializing shards, and unassigned shards.

Search Performance: Request Rate & Latency

The goodness of a data source is depended on its usefulness, and the effectiveness of the cluster can be measured by the rate at which the system is processing requests and how long each request is taking.

When a cluster receives a request, it may need to access data from many shards across different nodes. Knowing the rate of processing & returning requests, several requests in progress, and the time of the requests can provide precious insights into the cluster's health.

The request process has two phases – the first is the query phase. In this phase, the cluster distributes the request to each shard within the index. The 2nd one is the fetch phase. The query results are collected, compiled, and returned to the user.

We must be aware of spikes in any of the mentioned metrics, also any emerging trends which might point towards growing problems within the cluster. The index calculates these metrics, which are available from the cluster's RESTful endpoints.

You can refer to the table below for metrics available from the index endpoint. It is found at /index_name/_stats. Here index_name is the name of the index.

Index Performance: Refresh & Merge Times

As you update, add, and remove documents from an index, the cluster needs to update its indexes continually and refresh them across all the nodes. All of this is done by the cluster, and you, as a user, have limited control over the entire process except configuring the refresh interval rate.

It is seen that additions, deletions, and updates are batched together, and then they are flushed to disk as a new segment. As we see that each segment consumes resources, it is essential for the sake of performance that small segments are consolidated together and merged into more significant segments. Just like indexing, this, too, is managed by the cluster itself.

IMPORTANT METRICS FOR INDEX PERFORMANCE

Total refreshes	Count of the total number of refreshes.
Total time spent refreshing	Aggregation of all time spent refreshing. Measure in milliseconds.
Current merges	Merges currently being processed.
Total merges	Count of the total number of merges.
Total time spent merging	Aggregation of all time spent merging segments

Node Health: Memory, Disk, and CPU Metrics

Each node runs with the help of physical hardware and must access system memory, disk storage, and CPU cycles for doing the data management under its control and responding to requests to the cluster.

Elasticsearch system relies heavily on memory to be performant; therefore, keeping a close watch on memory usage is very relevant for the health and performance of each node. Configuration changes for improving metrics may also lead to an adverse effect on memory allocation and usage. Therefore, it's essential to view system health holistically.

IMPORTANT METRICS FOR NODE HEALTH

Total disk capacity	Total disk capacity on the node's host machine.
Total disk usage	Total disk usage on the node's host machine.
Total available disk space	Total disk space available.
Percentage of disk used	Percentage of the disk which is already used.
Current RAM usage	Current memory usage (unit of measurement).
RAM percentage	Percentage of memory being used.
Maximum RAM	The total amount of memory on the node's host machine
CPU	Percentage of the CPU in use.

JVM Health: Heap, GC, and Pool Size

Since Elasticsearch is a Java-based application, it runs within a Java Virtual Machine (JVM). The JVM uses heap allocation to manage its memory, and it evicts objects from its head through the garbage collection process.

If the application's needs exceed the heap's capacity, the application is required to begin using swap space on the provided storage media. While this arrangement prevents the system from crashing, it can negatively affect the cluster's performance. To ensure a healthy system, it's essential to monitor the available heap space so that the system has sufficient capacity.

Different memory pools get the JVM memory. It would help if you kept an eye on these pools so that they are adequately utilized and they are not facing the danger of being overrun.

IMPORTANT METRICS FOR JVM HEALTH

Memory Usage	Usage statistics for heap and non-heap processes and pools.
Threads	Current threads in use, and maximum number.
Garbage Collection	Counts and total time spent with garbage collection.

ISmile Technologies helps you to identify key metrics that you need to maintain the health & performance of your Elasticsearch Cluster.

The tight coupling between various metrics and knowing how changes in configuration might affect each metrics requires a team of experienced and well-trained engineers. ISmile Technologies has a pool of experienced talents to help you in this.

Request a Free Consultation

Elasticsearch Open Source Monitoring Tools

In this section, we will discuss some Elasticsearch open source monitoring tools that can be used to monitor your Elasticsearch cluster. We will also discuss about the pros & cons of using an Elasticsearch open source monitoring solution for monitoring your Elasticsearch cluster.

Characteristics of a Good Elasticsearch Monitoring Tool

Before we go into monitoring tools, we will establish a baseline of the essential features that a proper monitoring tool must include. First & foremost, it should be easy to install, configure and maintain. An able Elasticsearch monitoring tool must be able to gather all of the following metrics:

- System metrics like memory, CPU, and disk reads and writes.
- JVM metrics include heap usage, pool size, and garbage collector performance.
- Elasticsearch-specific metrics include request latency, indexing rate, merges and refreshes, cluster health, and node availability and usage.

Once the Elasticsearch tool has collected these metrics,
we can use them to provide:

- Visual indicators of system health on dashboards and using graphs.
- Automatic monitoring of health with anomaly detection and sending alerts on undesirable behavior.
- Integrations with communication systems such as Slack and IMS (incident management systems) like PagerDuty.

Popular Elasticsearch Monitoring Tools

In this section we will look at some of the very popular open source monitoring tools that are used today, and compare them against the tool characteristics we provided above. Specifically, we will review:

Elasticsearch HQ

Elastic Monitoring with the Stats API & Kibana

Kopf & Cerebro

Elasticsearch Head

ElasticHQ / Elasticsearch HQ

ElasticHQ is a popular open-source Elasticsearch monitoring tool. Roy Russo, the co-author of Elasticsearch in Action, started it. This tool can be availed from GitHub as a Python-based project, or it can be availed as a Docker Image from Docker Hub.

Its project website has elementary documentation, and it recommends users post their questions, problems, and bugs on the GitHub Issue Tracker. One thing is clear, this tool is very functional and gives users simplified access to the health and cat APIs available on the Elasticsearch cluster.

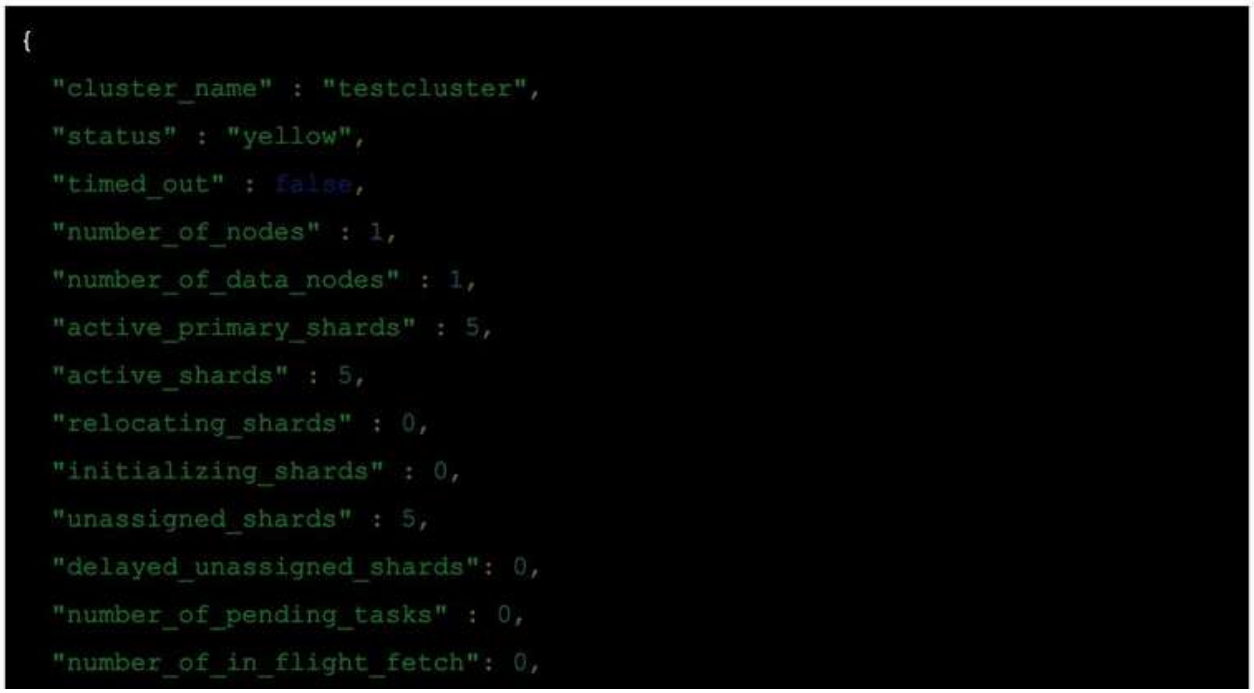
The project is aggressively maintained and is also updated on a regular cadence. One potential security concern mentioned in the documentation is that the username and password used to access the Elasticsearch cluster are kept in plain text within the ElasticHQ database. This leads to quicker connections to the cluster but will lead to heartburn among those responsible for providing security to your environments.

Elasticsearch Monitoring with Kibana

The primary data source component in the collection of products known as the Elastic Stack is Elasticsearch. Each component within this stack monitors itself and reports the results to the Elasticsearch cluster – which also monitors the environment. One should collect metrics in a different cluster, mainly if you're operating an Elasticsearch cluster within the production.

Elasticsearch consists of a comprehensive API component to query and get information about the cluster health, nodes, and tasks. The endpoints that can be called & example queries and responses are documented well in the Elasticsearch Reference.

Example of Response to a Request Sent to GET_cluster-health



Elasticsearch Kopf and Cerebro

Elasticsearch Kopf is also a prominent open-source monitoring option. However, the project only supports Elasticsearch up to version 2.0, as it hasn't been updated since the end of 2015.

For the latest versions, Cerebro supports them on Elasticsearch. The project can be availed from its GitHub repository. This project required the Java 1.8 runtime environment and was completed using a combination of Scala and Angular.

Project documentation is limited to the README page, which mentions how to get the tool running and also specifies how to build and run a Docker image with the same. The last stable release of the project was in June 2018.

Elasticsearch Head

This is the final open-source utility – ElasticSearch Head. This product is available on GitHub and is being regularly maintained and updated. The tool has been written with Node.js, and it can be run in the following forms:

- Built-in server
- Docker
- Chrome extension

Compared to the previous two tools, this project has more updates and enjoys greater community participation. The project also has an essential website, installation instructions, and several vital screenshots.

Conclusion

Choosing a proper monitoring tool that can efficiently monitor your Elasticsearch helps you maintain your Elasticsearch clusters better. It also ensures that they remain healthy and offer good performance for the end users. Furthermore, open-source tools come with a budget-friendly price tag, but if you are not well-versed in handling these tools, you may face problems and maintenance overhead in your organization. Rely on ISmile Technologies for your complete Elasticsearch monitoring requirements. Being an expert in this field, we guarantee peace of mind with us.

About ISmile Technologies

ISmile Technologies is a proud automation-enabled intelligent cloud solution and managed IT services provider, and it is your multi-cloud technology advisor & key implementation partner.

We operate globally and leverage disruptive technologies alongside deep expertise to deliver business-specific cloud solutions. We maximize impact at an unparalleled value and securely accelerate business agility while infusing competitive excellence.

https://www.ismiletechnologies.com

sales@ISmileTechnologies.com

501 S Weber Rd Unit 108, Bolingbrook, IL 60490