**ElasticSearch**

*Brief by Anandh Varadarajan, Matthew Matero, and Aravind Reddy Ravula*

***Why should you care? -* ElasticSearch** is a highly-scalable, open-source, Java-based, distributed, multi-tenant capable **search engine**. It allows near real-time storage, search and analysis of large volumes of data. It is built over Apache Lucene and uses a HTTP interface and JSON. As it is highly scalable and cloud oriented, ElasticSearch is used extensively to handle “Big Data”. ElasticSearch is also integrated deeply with logging and visualization software, leveraging LogStash and Kibana respectively, known as the elastic stack.

***Approach –*** The primary data structure Elasticsearch uses is an inverted index managed using Apache Lucene’s APIs. In very simple terms, an inverted index is a mapping of each unique ‘word’ to the list of documents (locations) containing that word. Index information is stored in one or multiple partitions also called shards. ElasticSearch is made up of nodes that hold shards and a single clustered, collection of nodes.

Nodes in an Elasticsearch cluster can be configured in three different ways: (1) **Master nodes** are responsible for creating/deleting indexes, tracking which nodes are part of the cluster, deciding which shards to allocate to which nodes. (2) **Data nodes** are in charge of the CRUD operations, search queries, aggregations, and store data within shards.(3) **Client nodes** handle the load balancing for incoming requests or bulk reindexing of the cluster.

Web UI

(Search Query)

Collector

(Index Query)

Search

Indexing

**ElasticSearch** Load Balancers

**ElasticSearch** Master Nodes

Search

Indexing

Data Nodes

Figure 1: General workflow of an elasticsearch cluster and its internal routing/organization when performing operations(searching and indexing). **(**Recreated, based on figure from [CUBRID](https://www.cubrid.org/blog/our-experience-creating-large-scale-log-search-system-using-elasticsearch))

***Results -*** ElasticSearch’s biggest competitor is Apache Solr. However, there are key differences in how each is used within the industry. While ElasticSearch is open source it is still organized by Elastic, which gives it more support when considering plugins for monitoring and visualizing your search cluster. Additionally, there are internal differences as to how each organizes and queries their data stores, as shown in the following table.

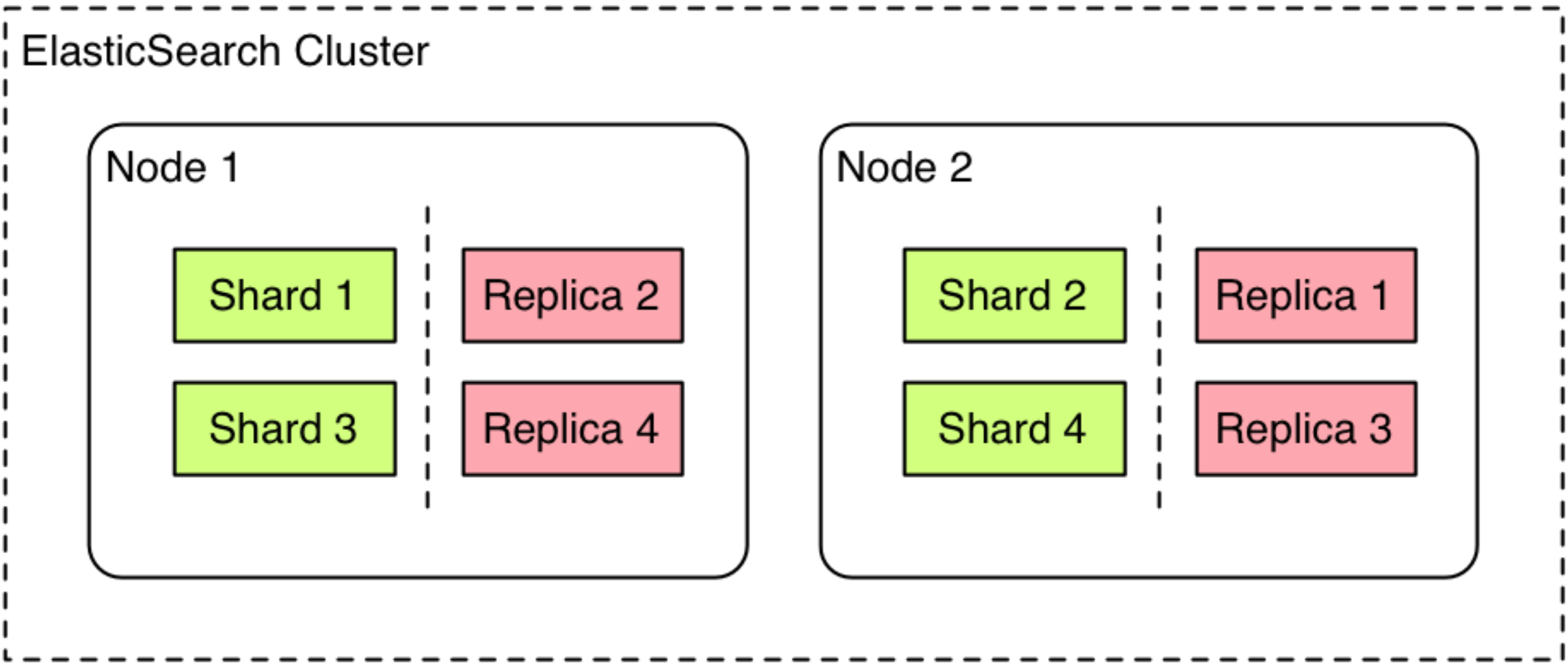


Figure 2: Each node of an ElasticSearch cluster holds internal shards and replicas of shards that are distributed across the cluster. **(**[Demosfri](https://wiki.deimos.fr/ElasticSearch:_powerful_search_and_analytics_engine.html), 2014).

|  |  |  |
| --- | --- | --- |
|  | **Apache Solr** | **ElasticSearch** |
| *Searching* | Oriented towards text search | Analytical querying, filtering and grouping |
| *Performance* | Works best on static data | Handles “dynamic” data better |
| *Scalability* | Comes with SolrCloud | Designed for scale. Can be setup locally and cloud |
| *Data Format* | XML, CSV, JSON | JSON |

Table 1: Comparison between the main differences between choosing ElasticSearch and Apache Solr.

***Pros/Cons***

* Pro: Easier to horizontally scale, distributed by design, excellent full-text search
* Pro: Asynchronous indexing for dynamically updating the search indices for real time accuracy
* Con: Possible high latency if no index for a query exists since index requests are sent to each shard this will take a while for highly distributed setup.
* Con: Doesn’t support geographic distribution of nodes. It might become an issue when an entire datacenter faces downtime.

***Conclusion -*** Elasticsearch is an easy to use and easy to setup alternative to other search services due to its HTTP REST based interface. It offers better support for other aspects of working with Big Data(viz + logs) when compared to Apache Solr.