sematext.com

Elasticsearch Tutorial: Getting Started Guide for Beginners - Sematext

20-25 minutes

Search and Analytics are key features of modern software applications. Scalability and the capability to handle large volumes of data in near real-time is a must for many applications such as mobile apps, web, and data analytics applications.

Today, autocomplete in search fields, search suggestions, location search, and faceted navigation are de facto standards in usability. This is where Elasticsearch comes in, as it's often the engine that powers such experiences. So, let's dive in starting with the "What is Elasticsearch?" question, and then we'll dig further to explore all its aspects.

Definition: What Is Elasticsearch?

Elasticsearch is a free, open-source search and analytics engine based on the Apache Lucene library. It's the most popular search engine and has been available since 2010. It's developed in Java, supporting clients in many different languages, such as PHP, Python, C#, and Ruby.

What Does Elasticsearch Do?

Elasticsearch can be used to search all kinds of data. It provides a scalable search solution, has near real-time search and support for multi tenancy. Elasticsearch takes in unstructured data from different locations, stores and indexes it according to user-specified mapping (which can also be derived automatically from data), and makes it searchable.

Its distributed architecture makes it possible to search and analyze huge volumes of data in near real time. It allows you to start with one machine and scale to hundreds. Elasticsearch makes it easy to run a full-featured search cluster, though running it at scale still requires a substantial level of expertise.

Besides full-text search-oriented use cases like product search, document search, email search, etc., Elasticsearch is often used for storing data that needs to be sliced and diced, grouped by various dimensions, and such. Examples of such analytical use cases include the use of Elasticsearch for metrics, logs, traces, and other timeseries data.

Elasticsearch is a formidable competitor to Apache Solr, as well as commercial search engines such as Splunk and other log analytics engines. If you want to see the two search engines side by side, read Elasticsearch vs. Solr, where we explain the differences and similarities between the two. And check out our review of Amazon's Open Distro for Elasticsearch which provides some of the previously commercial Elasticsearch extensions out-of-the-box and without any licensing.

You can run Elasticsearch both on-premise and in the Cloud. You can choose to run it yourself or use one of the hosted Elasticsearch services, like AWS Elasticsearch. If you need help deciding, see AWS Elasticsearch Service vs. Elasticsearch on EC2.

If you are using Elasticsearch for log analytics you can again take the DIY or hosted Elasticsearch route, but you can also find specialized log management service vendors that provide not just Elasticsearch, but also other functionality needed for Elasticsearch log management use case, such as Sematext's Hosted ELK.

In this talk, Radu Gheorghe from Sematext gives an Introduction to Elasticsearch (you can find his Elasticsearch intro presentation slides here):

Benefits: Why Use Elasticsearch?

Built with Java, this datastore allows you to run it on any platform. Compared to most NoSQL databases, Elasticsearch is much more focused on the search functionalities, equipped with a rich and powerful HTTP RESTful API that enables you to perform fast searches in near real time.

To fully understand how Elasticsearch can help you and why you may want to use it, we've gathered some of its main capabilities:

Full Text Search Engine

Traditional SQL database management systems are not designed for full-text searches against large volumes of data. Because it's built on top of Lucene, Elasticsearch offers one of the most powerful full-text search capabilities and lets you perform and combine many types of searches, from structured, unstructured, geo, to metric.

Analytical Engine

The analytical use case is the most popular Elasticsearch use case, even more popular than full text search. Specifically, Elasticsearch is often used for log analytics, slicing and dicing of numerical data such as application and infrastructure performance metrics. Although Apache Solr provided faceting before Elasticsearch was even born, Elasticsearch took faceting to another level, enabling its users to aggregate data on the fly using Elasticsearch's aggregation queries. These aggregation queries are what powers pretty much all data visualizations you see in tools like Kibana, Grafana, and others.

Distributed Architecture Designed for Scaling

Elasticsearch was built to scale from the beginning. Its distributed architecture allows you to scale Elasticsearch to a lot of servers and accommodate petabytes of data. As part of our Elasticsearch consulting practice, we've seen clusters with hundreds of nodes.

Distributed systems are complex, but Elasticsearch makes many decisions automatically and provides a good management API. Scaling Elasticsearch is, therefore, much easier than with many other systems, though large Elasticsearch clusters come with their set of issues and often require Elasticsearch expertise. Elasticsearch can also replicate data automatically to prevent data loss in case of node failures.

Effective Investment Right Out of The Box

Elasticsearch's mechanics are quite easy to grasp, at least when one is dealing with a relatively small dataset or small deployment. Its simple RESTful APIs work with ingestion tools such as Logstash to send data to Elasticsearch as JSON documents, or Kibana to build reports and visualize your data.

These capabilities along with a short learning curve, allow you to quickly start working on use cases and become more productive.

Rich Ecosystem

One of the main reasons for Elasticsearch rise in popularity is its well-documented API. The availability of this API made it possible for developers to integrate with it and over time that is exactly what they did. Virtually every log shipper or logging library have adapters for sending data to Elasticsearch. Logstash may be the most popular one, but there are many others. For example, in our blog post about Logstash alternatives, we highlight 5 other log shippers, one of which is Logagent with its own Elasticsearch plugin.

Find out if you should stick to Logstash or rather send data directly to Elasticsearch from Elasticsearch ingest node vs. Logstash performance.

Besides various tools that can ingest data into Elasticsearch via its

API, there are also tools like Kibana or Grafana aimed at Elasticsearch data exploration, analysis, and visualization.

Read more about how Elasticsearch integrates with other tools:

- How to use Kafka Connect to send data from a Kafka topic to Elasticsearch
- Elastic Stack Import-Export with Logstash & Logsene
- Shipping data to Elasticsearch with Logagent
- Structured logging with rsyslog and Elasticsearch

Compatible with Many Languages

Elasticsearch has client libraries for many programming languages such as Java, JavaScript, PHP, C#, Ruby, Python, Go, and many more. Availability of these client libraries makes it quite easy for developers to integrate with Elasticsearch.

How Elasticsearch Works

Elasticsearch works by retrieving and managing document-oriented and semi-structured data. Internally, the basic principle of how Elasticsearch works is the “shared nothing” architecture. 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’ (token) to the list of documents (locations) containing that word, which makes it possible to locate documents with given keywords very quickly. Index information is stored in one or multiple partitions also called shards. Elasticsearch is able to distribute and allocate shards dynamically to the nodes in a cluster, as well as replicate them.

This mechanism makes it flexible with regard to data distribution. Redundancy can be provided by distributing replica shards ('copies' of the primary shards) to different cluster nodes. Index operations use primary shards and search queries use both shard types. Having multiple nodes and replicas increases query performance.

Understanding the Basic Concepts of Elasticsearch

Let's take a look at the basic concepts of Elasticsearch, from index to clusters, indexes, nodes, shards, mapping, and more:

JVM

Elasticsearch is written in Java and thus uses the Java Virtual Machine (JVM). The JVM is a runtime engine that executes bytecode on many operating system platforms.

Further reading:

- Learn how Elasticsearch cache usage eats at the JVM heap memory
- Discover the best open-source Elasticsearch monitoring tools and how to monitor Elasticsearch with Sematext

Index

An index is a collection of documents that often have a similar structure and is used to store and read documents from it. It's the equivalent of a database in RDBMS (relational database management system). The index is identified by a unique index name that you will refer to whenever you perform search, update or delete actions.

Using an inverted index is a lot like searching for a book page that contains a certain keyword by scanning the index at the back of the book instead of scanning every page from beginning to end. This inverted index enables Elasticsearch to retrieve data quickly and efficiently.

In terms of data modeling, it could be compared to a collection in MongoDB or CouchDB. A single index can hold one data type, with its own data structure, while in a cluster you can have more than one index. The schema is defined by the Mapping. An index is built from 1-N primary shards, which can have 0-N replica shards.

Further reading:

- Reindexing data with Elasticsearch
- Scalable and flexible Elasticsearch reindexing via rsyslog

Shard

A shard is a subset of documents of an index. Elasticsearch uses shards when the volume of data stored in your cluster exceeds the limits of your server. Therefore, it allows you to split your index into smaller pieces called shards. A shard is a single Lucene index instance. Elasticsearch has two types of shards:

- primary shards, or active shards that hold the data
- replica shards, or copies of the primary shard

Mapping

A mapping is the schema definition for the index. Extending the mapping with new fields or adding sub-fields is possible at any time, but changing the type of fields is a more complex operation including re-indexing of the data.

When no mapping is defined, Elasticsearch tries to detect the type of field (String, Number, IP, Geo-Point) automatically. It creates an automatic mapping for the data type and sets default analyzers for strings and adds the “keyword” sub-field (not analyzed). By default you get a string mapped as both text and a keyword sub-field. So you can do full-text search on one hand, and exact matches, sorting and aggregations on the other.

It’s important to define the correct mapping to avoid problems at query time. For example, you want to avoid having Elasticsearch identify some field as Number and then later try indexing data that, in that same field now contains a string. Trying to index such data will fail.

Segments

A segment is a Lucene-level concept. They represent chunks of a shard (Lucene Index). Each Lucene index contains one or more segments. While this is a Lucene-level thing, Elasticsearch does offer knobs to manage segment sizes and how you configure that will have an impact on Elasticsearch indexing performance, as pointed out in our article about the key Elasticsearch metrics you should to monitor.

Document

A document is the main and basic unit of information entity in Elasticsearch and is represented in JSON (JavaScript Object Notation) format. Documents can be stored and indexed. An index has one or more documents and a document has one or more fields. The original is represented as “_source” in the API besides the actual indexed fields of a document.

Search is only possible against indexed fields and retrieving the original field content is only possible in fields defined as “stored” in the Mapping (aside from the mentioned “_source” object that holds the complete document values).
For efficient field-based display, the stored flag should be set when the “_source” objects are large – this can reduce network traffic and speed up the display of results. In RDBMS terms, a document is a row.

Further reading:

- How to update documents by query with Elasticsearch

Node

A node is a single instance of Elasticsearch process. It’s a server that stores data and is a part of the cluster’s indexing and searching functions. Nodes discover each other in the cluster by their shared cluster name.

A cluster can have multiple nodes depending on the node configuration, multicast or unicast discovery is used. Multiple nodes can run on a single physical server, VM, or container. The two main node types are data nodes and master nodes. Nodes can be configured to hold data or act as cluster master nodes, or both.

Cluster

A cluster consists of one or more nodes (servers) that store all the data and provides indexing and searching capabilities across all nodes. Each cluster has a single active master node, which is automatically elected (e.g., when the current master node fails).

Replica

A replica is a mechanism that Elasticsearch uses to handle failures such as a node going offline, without losing data. It's a copy of the primary shard and can be used for searches just as the original shard.

Now that you know the main Elasticsearch concepts, download our Elasticsearch DevOps cheat sheet to learn the basic commands to manage your clusters!

Elasticsearch Use Cases & Applications Examples

As a distributed engine, Elasticsearch is highly scalable and offers near real-time search capabilities. This adds up to a solution that can do more than a search engine and supports a multitude of growing critical business needs and operational use cases.

Generally, thanks to its powerful search capabilities, Elasticsearch is used as the underlying technology that powers applications with complex search features and requirements. From numbers, text, geo, structured, unstructured, Elasticsearch supports all data types.

Elasticsearch is popular due to its versatile nature in handling data and being paired with other tools. Companies like Wikipedia, Github, NY Times or Facebook all use Elasticsearch for various use cases: from easy search for all 164 years of published articles to instantaneous live chat or seamless e-commerce experience, any business that needs to serve information in a fast way can put Elasticsearch to good use.

With pretty much endless and versatile capabilities that continue to grow and change depending on business goals, here’s how businesses have used Elasticsearch for different use cases:

Instantaneous E-commerce Search Across Retail Product Catalogues

Retailers are using Elasticsearch to index their product catalogs and inventory, alongside all the product attributes, so when the clients search for a specific product attribute, their store can display the right products instantly.

A near instant search bar can boost revenue by delivering a better product catalog search experience and make search the primary form of navigation.

Walgreens and Kreeger are some of the biggest retail companies streamlining their online grocery shopping experience with Elasticsearch.

Operational Logging Analytics

Using Elasticsearch to process billions of events every day to analyze logs and ensure consistent system performance or detect anomalies helped companies like GoDaddy to improve customer experience and enhance the user experience. Check out our blog post about how to use Elasticsearch for log analysis or how to use it together with Grafana to learn more about this type of use case.

Site Content and Media Search

Engadget and The New York Times are using Elasticsearch for site content search to better understand what their users are searching for and why – all with the goal to improve their user engagement KPIs.

Using Elasticsearch for site content search is not limited to publishers – Shopify and Asana also use it to make their documentation and support content easily findable to clients. Search is also not limited to articles. One of the biggest video hosting companies, Vimeo, powers the search of millions of videos every day through Elasticsearch.

Instantaneous Live Chat

Live Chat is one company that improved the customer experience for 6,000 customers conducting millions of queries daily – all by using Elasticsearch to maintain an archive of 460 million documents and deliver instantaneous query response times.

Fraud Monitoring and Early Detection

SoftBank and Xoom are preventing and protecting against fraud and security threats by monitoring their system with Elasticsearch.

Application Search

One of the biggest companies using Elasticsearch for application search is eBay, searching across 800 million listings in subseconds and maintaining a world-class end-user experience for millions of people every day.

Business Analytics

Walmart is using Elasticsearch to gain insights into customer purchasing patterns and store performance metrics, in order to enhance the in-store and online retail customer shopping experience and boost their commercial success.

Enterprise Search

Facebook uses Elasticsearch and has gone from a simple enterprise search to over 40 tools across multiple clusters with 60+ million queries a day and growing.

Metrics Analytics

Sprint is using Elasticsearch to analyze over 200 dashboards, representing 3 billion events per day from logs, databases, emails, syslogs, test messages, and internal and vendor application APIs, in order to search for better retail operations insights.

Security Analytics

Slack is building a defensive security program to monitor malicious activity by using Elasticsearch. Cisco is also using Elasticsearch to leverage data to detect and defeat hackers and fight cyber threats.

Scraping and Analyzing Public Data

Public data like social media conversations can be mined by using Elasticsearch to do real-time analysis, resulting in a social sentiment analysis to understand customers.

However, these applications only scratch the surface of how companies can use Elasticsearch to solve a variety of growing challenges.

You can also check out this fun Elasticsearch use case where we put together our own Internet of Things setup for measuring air pollution using a couple of IoT devices, Node.js, Elasticsearch, and MQTT.

Running Elasticsearch in the Cloud

Elasticsearch can run both on-premise and in the Cloud. You can choose to self-manage it or reach out to a log management service provider to manage it for you, such as Sematext Cloud.

With the former, you will need to have a team of specialists that can set up and configure Elasticsearch, provide the hardware, and manage the cluster. If you have an in-house team, that’s great! However, there are solutions such as Sematext Enterprise that enable you to easily set up the full-stack observability platform on your own infrastructure and that your team can use for all their monitoring and log management needs.

If you want to run ELK stack on your own you’ll need to master

Elasticsearch. Elasticsearch training classes will help you get there!

On the other hand, with Sematext Cloud you get a fully managed service that eliminates the need to manage any infrastructure while offering you the Elasticsearch API, Kibana, compatibility with Logstash, Filebeat and other Beats, and other tools from the ecosystem.

Furthermore, Sematext is your one-stop-shop for all things Elasticsearch.

Besides hosted ELK Sematext can help you with: expert Elasticsearch consulting, production support, and Elasticsearch monitoring.

If you want to continue learning about Elasticsearch, here are a few blog posts that might interest you further:

- Running Elasticsearch clusters on Docker
- How to make Elasticsearch in Docker Swarm Elastic
- Running and deploying Elasticsearch on Kubernetes
- Running and deploying Elasticsearch Operator on Kubernetes
- Elasticsearch refresh interval vs. indexing performance
- Elasticsearch security: Authentication, encryption, and backup