How does ES do it?

Elasticsearch uses a data structure called an inverted index that supports very fast full-text searches. An inverted index lists every unique word that appears in any document and identifies all of the documents each word occurs in.

An index can be thought of as an optimized collection of documents and each document is a collection of fields, which are the key-value pairs that contain your data. By default, Elasticsearch indexes all data in every field and each indexed field has a dedicated, optimized data structure. For example, text fields are stored in inverted indices, and numeric and geo fields are stored in BKD trees. The ability to use the per-field data structures to assemble and return search results is what makes Elasticsearch so fast.

Elasticsearch also has the ability to be schema-less, which means that documents can be indexed without explicitly specifying how to handle each of the different fields that might occur in a document. When dynamic mapping is enabled, Elasticsearch automatically detects and adds new fields to the index. This default behavior makes it easy to index and explore your data—​just start indexing documents and Elasticsearch will detect and map booleans, floating point and integer values, dates, and strings to the appropriate Elasticsearch data types.

Ultimately, however, you know more about your data and how you want to use it than Elasticsearch can. You can define rules to control dynamic mapping and explicitly define mappings to take full control of how fields are stored and indexed.

Defining your own mappings enables you to:

* Distinguish between full-text string fields and exact value string fields
* Perform language-specific text analysis
* Optimize fields for partial matching
* Use custom date formats
* Use data types such as geo\_point and geo\_shape that cannot be automatically detected

It’s often useful to index the same field in different ways for different purposes. For example, you might want to index a string field as both a text field for full-text search and as a keyword field for sorting or aggregating your data. Or, you might choose to use more than one language analyzer to process the contents of a string field that contains user input.

The analysis chain that is applied to a full-text field during indexing is also used at search time. When you query a full-text field, the query text undergoes the same analysis before the terms are looked up in the index.

Information out: search and analyze

While you can use Elasticsearch as a document store and retrieve documents and their metadata, the real power comes from being able to easily access the full suite of search capabilities built on the Apache Lucene search engine library.

Elasticsearch provides a simple, coherent REST API for managing your cluster and indexing and searching your data. For testing purposes, you can easily submit requests directly from the command line or through the Developer Console in Kibana. From your applications, you can use the [Elasticsearch client](https://www.elastic.co/guide/en/elasticsearch/client/index.html) for your language of choice: Java, JavaScript, Go, .NET, PHP, Perl, Python or Ruby.

**Searching your data**

The Elasticsearch REST APIs support structured queries, full text queries, and complex queries that combine the two. Structured queries are similar to the types of queries you can construct in SQL. For example, you could search the gender and age fields in your employee index and sort the matches by the hire\_date field. Full-text queries find all documents that match the query string and return them sorted by *relevance*—how good a match they are for your search terms.

In addition to searching for individual terms, you can perform phrase searches, similarity searches, and prefix searches, and get autocomplete suggestions.

Have geospatial or other numerical data that you want to search? Elasticsearch indexes non-textual data in optimized data structures that support high-performance geo and numerical queries.

You can access all of these search capabilities using Elasticsearch’s comprehensive JSON-style query language ([Query DSL](https://www.elastic.co/guide/en/elasticsearch/reference/7.8/query-dsl.html)). You can also construct [SQL-style queries](https://www.elastic.co/guide/en/elasticsearch/reference/7.8/sql-overview.html) to search and aggregate data natively inside Elasticsearch, and JDBC and ODBC drivers enable a broad range of third-party applications to interact with Elasticsearch via SQL.

**Analyzing your data**

Elasticsearch aggregations enable you to build complex summaries of your data and gain insight into key metrics, patterns, and trends. Instead of just finding the proverbial “needle in a haystack”, aggregations enable you to answer questions like:

* How many needles are in the haystack?
* What is the average length of the needles?
* What is the median length of the needles, broken down by manufacturer?
* How many needles were added to the haystack in each of the last six months?

You can also use aggregations to answer more subtle questions, such as:

* What are your most popular needle manufacturers?
* Are there any unusual or anomalous clumps of needles?

Because aggregations leverage the same data-structures used for search, they are also very fast. This enables you to analyze and visualize your data in real time. Your reports and dashboards update as your data changes so you can take action based on the latest information.

What’s more, aggregations operate alongside search requests. You can search documents, filter results, and perform analytics at the same time, on the same data, in a single request. And because aggregations are calculated in the context of a particular search, you’re not just displaying a count of all size 70 needles, you’re displaying a count of the size 70 needles that match your users' search criteria—​for example, all size 70 *non-stick embroidery* needles.

**But wait, there’s more**

Want to automate the analysis of your time-series data? You can use [machine learning](https://www.elastic.co/guide/en/machine-learning/7.8/ml-overview.html) features to create accurate baselines of normal behavior in your data and identify anomalous patterns. With machine learning, you can detect:

* Anomalies related to temporal deviations in values, counts, or frequencies
* Statistical rarity
* Unusual behaviors for a member of a population

And the best part? You can do this without having to specify algorithms, models, or other data science-related configurations.

## Scalability and resilience: clusters, nodes, and shards

Elasticsearch is built to be always available and to scale with your needs. It does this by being distributed by nature. You can add servers (nodes) to a cluster to increase capacity and Elasticsearch automatically distributes your data and query load across all of the available nodes. No need to overhaul your application, Elasticsearch knows how to balance multi-node clusters to provide scale and high availability. The more nodes, the merrier.

How does this work? Under the covers, an Elasticsearch index is really just a logical grouping of one or more physical shards, where each shard is actually a self-contained index. By distributing the documents in an index across multiple shards, and distributing those shards across multiple nodes, Elasticsearch can ensure redundancy, which both protects against hardware failures and increases query capacity as nodes are added to a cluster. As the cluster grows (or shrinks), Elasticsearch automatically migrates shards to rebalance the cluster.

There are two types of shards: primaries and replicas. Each document in an index belongs to one primary shard. A replica shard is a copy of a primary shard. Replicas provide redundant copies of your data to protect against hardware failure and increase capacity to serve read requests like searching or retrieving a document.

The number of primary shards in an index is fixed at the time that an index is created, but the number of replica shards can be changed at any time, without interrupting indexing or query operations.

#### It depends…​

There are a number of performance considerations and trade offs with respect to shard size and the number of primary shards configured for an index. The more shards, the more overhead there is simply in maintaining those indices. The larger the shard size, the longer it takes to move shards around when Elasticsearch needs to rebalance a cluster.

Querying lots of small shards makes the processing per shard faster, but more queries means more overhead, so querying a smaller number of larger shards might be faster. In short…​it depends.

As a starting point:

* Aim to keep the average shard size between a few GB and a few tens of GB. For use cases with time-based data, it is common to see shards in the 20GB to 40GB range.
* Avoid the gazillion shards problem. The number of shards a node can hold is proportional to the available heap space. As a general rule, the number of shards per GB of heap space should be less than 20.

The best way to determine the optimal configuration for your use case is through [testing with your own data and queries](https://www.elastic.co/elasticon/conf/2016/sf/quantitative-cluster-sizing).

#### In case of disaster

For performance reasons, the nodes within a cluster need to be on the same network. Balancing shards in a cluster across nodes in different data centers simply takes too long. But high-availability architectures demand that you avoid putting all of your eggs in one basket. In the event of a major outage in one location, servers in another location need to be able to take over. Seamlessly. The answer? Cross-cluster replication (CCR).

CCR provides a way to automatically synchronize indices from your primary cluster to a secondary remote cluster that can serve as a hot backup. If the primary cluster fails, the secondary cluster can take over. You can also use CCR to create secondary clusters to serve read requests in geo-proximity to your users.

Cross-cluster replication is active-passive. The index on the primary cluster is the active leader index and handles all write requests. Indices replicated to secondary clusters are read-only followers.

#### Care and feeding

As with any enterprise system, you need tools to secure, manage, and monitor your Elasticsearch clusters. Security, monitoring, and administrative features that are integrated into Elasticsearch enable you to use [Kibana](https://www.elastic.co/guide/en/kibana/7.8/introduction.html) as a control center for managing a cluster. Features like [data rollups](https://www.elastic.co/guide/en/elasticsearch/reference/7.8/rollup-overview.html) and [index lifecycle management](https://www.elastic.co/guide/en/elasticsearch/reference/7.8/index-lifecycle-management.html) help you intelligently manage your data over time.

**Kibana:**

**Explore and visualize your data and manage all things Elastic Stack.**

Whether you’re a user or admin, Kibana makes your data actionable by providing three key functions. Kibana is:

* **An open-source analytics and visualization platform.** Use Kibana to explore your Elasticsearch data, and then build beautiful visualizations and dashboards.
* **A UI for managing the Elastic Stack.** Manage your security settings, assign user roles, take snapshots, roll up your data, and more — all from the convenience of a Kibana UI.
* **A centralized hub for Elastic’s solutions.** From log analytics to document discovery to SIEM, Kibana is the portal for accessing these and other capabilities.

Getting data into Kibana

Kibana is designed to use Elasticsearch as a data source. Think of Elasticsearch as the engine that stores and processes the data, with Kibana sitting on top.

From the home page, Kibana provides these options for getting data in:

* Set up a data flow to Elasticsearch using our built-in tutorials. (If a tutorial doesn’t exist for your data, go to the [Beats overview](https://www.elastic.co/guide/en/beats/libbeat/7.8/beats-reference.html) to learn about other data shippers in the Beats family.)
* [Add a sample data set](https://www.elastic.co/guide/en/kibana/7.8/add-sample-data.html) and take Kibana for a test drive without loading data yourself.
* Import static data using the [file upload feature](https://www.elastic.co/blog/importing-csv-and-log-data-into-elasticsearch-with-file-data-visualizer).
* Index your data into Elasticsearch with [REST APIs](https://www.elastic.co/guide/en/elasticsearch/reference/7.8/getting-started-index.html) or [client libraries](https://www.elastic.co/guide/en/elasticsearch/client/index.html).

Kibana uses an [index pattern](https://www.elastic.co/guide/en/kibana/7.8/index-patterns.html) to tell it which Elasticsearch indices to explore. If you add sample data or run a built-in tutorial, you get an index pattern for free, and are good to start exploring. If you load your own data, you can create an index pattern in [Management](https://www.elastic.co/guide/en/kibana/7.8/management.html).

### Explore & query

Ready to dive into your data? With [Discover](https://www.elastic.co/guide/en/kibana/7.8/discover.html), you can explore your data and search for hidden insights and relationships. Ask your questions, and then narrow the results to just the data you want.

### Visualize & analyze

A visualization is worth a thousand log lines, and Kibana provides many options for showcasing your data. Use [Lens](https://www.elastic.co/guide/en/kibana/7.8/lens.html), our drag-and-drop interface, to rapidly build charts, tables, metrics, and more. If there is a better visualization for your data, **Lens** suggests it, allowing for quick switching between visualization types.

Once your visualizations are just the way you want, use [Dashboard](https://www.elastic.co/guide/en/kibana/7.8/dashboard.html) to collect them in one place. A dashboard provides insights into your data from multiple perspectives.

Kibana also offers these visualization features:

* [Visualize](https://www.elastic.co/guide/en/kibana/7.8/visualize.html) allows you to display your data in line charts, bar graphs, pie charts, histograms, and tables (just to name a few). It’s also home to Lens, the drag-and-drop interface. Visualize supports the ability to add interactive controls to your dashboard, and filter dashboard content in real time.
* [Canvas](https://www.elastic.co/guide/en/kibana/7.8/canvas.html) gives you the ability to present your data in a visually compelling, pixel-perfect report. Give your data the “wow” factor needed to impress your CEO or to captivate people with a big-screen display.
* [Maps](https://www.elastic.co/guide/en/kibana/7.8/maps.html) enables you to ask (and answer) meaningful questions of your location-based data. Maps supports multiple layers and data sources, mapping of individual geo points and shapes, and dynamic client-side styling.
* [TSVB](https://www.elastic.co/guide/en/kibana/7.8/TSVB.html) allows you to combine an infinite number of aggregations to display complex data in a meaningful way. With TSVB, you can analyze multiple index patterns and customize every aspect of your visualization. Choose your own date format and color gradients, and easily switch your data view between time series, metric, top N, gauge, and markdown.

### Organize & secure

Want to share Kibana’s goodness with other people or teams? You can do so with [Spaces](https://www.elastic.co/guide/en/kibana/7.8/xpack-spaces.html), built for organizing your visualizations, dashboards, and indices. Think of a space as its own mini Kibana installation — it’s isolated from all other spaces, so you can tailor it to your specific needs without impacting others.

You can even choose which features to enable within each space. Don’t need Machine learning in your “Executive” space? Simply turn it off.

You can take this all one step further with Kibana’s security features, and control which users have access to each space. Kibana allows for fine-grained controls, so you can give a user read-only access to dashboards in one space, but full access to all of Kibana’s features in another.

### Manage all things Elastic Stack

[Management](https://www.elastic.co/guide/en/kibana/7.8/management.html) provides guided processes for managing all things Elastic Stack — indices, clusters, licenses, UI settings, index patterns, and more. Want to update your Elasticsearch indices? Set user roles and privileges? Turn on dark mode? Kibana has UIs for all that.

**Logstash**

Logstash is an open source data collection engine with real-time pipelining capabilities. Logstash can dynamically unify data from disparate sources and normalize the data into destinations of your choice. Cleanse and democratize all your data for diverse advanced downstream analytics and visualization use cases.

While Logstash originally drove innovation in log collection, its capabilities extend well beyond that use case. Any type of event can be enriched and transformed with a broad array of input, filter, and output plugins, with many native codecs further simplifying the ingestion process. Logstash accelerates your insights by harnessing a greater volume and variety of data.

**The Power of Logstash**

**The ingestion workhorse for Elasticsearch and more**

Horizontally scalable data processing pipeline with strong Elasticsearch and Kibana synergy

**Pluggable pipeline architecture**

Mix, match, and orchestrate different inputs, filters, and outputs to play in pipeline harmony

**Community-extensible and developer-friendly plugin ecosystem**

Over 200 plugins available, plus the flexibility of creating and contributing your own

### Logs and Metrics

Where it all started.

* Handle all types of logging data
  + Easily ingest a multitude of web logs like [Apache](https://www.elastic.co/guide/en/logstash/7.8/advanced-pipeline.html), and application logs like [log4j](https://www.elastic.co/guide/en/logstash/7.8/plugins-inputs-log4j.html) for Java
  + Capture many other log formats like [syslog](https://www.elastic.co/guide/en/logstash/7.8/plugins-inputs-syslog.html), networking and firewall logs, and more
* Enjoy complementary secure log forwarding capabilities with [Filebeat](https://www.elastic.co/products/beats/filebeat" \t "_top)
* Collect metrics from [Ganglia](https://www.elastic.co/guide/en/logstash/7.8/plugins-inputs-ganglia.html), [collectd](https://www.elastic.co/guide/en/logstash/7.8/plugins-codecs-collectd.html" \t "_top), [NetFlow](https://www.elastic.co/guide/en/logstash/7.8/plugins-codecs-netflow.html), [JMX](https://www.elastic.co/guide/en/logstash/7.8/plugins-inputs-jmx.html), and many other infrastructure and application platforms over [TCP](https://www.elastic.co/guide/en/logstash/7.8/plugins-inputs-tcp.html) and [UDP](https://www.elastic.co/guide/en/logstash/7.8/plugins-inputs-udp.html)

### The Web

Unlock the World Wide Web.

* Transform [HTTP requests](https://www.elastic.co/guide/en/logstash/7.8/plugins-inputs-http.html) into events
  + Consume from web service firehoses like [Twitter](https://www.elastic.co/guide/en/logstash/7.8/plugins-inputs-twitter.html) for social sentiment analysis
  + Webhook support for GitHub, HipChat, JIRA, and countless other applications
  + Enables many [Watcher](https://www.elastic.co/products/x-pack/alerting) alerting use cases
* Create events by polling [HTTP endpoints](https://www.elastic.co/guide/en/logstash/7.8/plugins-inputs-http_poller.html) on demand
  + Universally capture health, performance, metrics, and other types of data from web application interfaces
  + Perfect for scenarios where the control of polling is preferred over receiving

### Data Stores and Streams

Discover more value from the data you already own.

* Better understand your data from any relational database or NoSQL store with a [JDBC](https://www.elastic.co/guide/en/logstash/7.8/plugins-inputs-jdbc.html) interface
* Unify diverse data streams from messaging queues like Apache [Kafka](https://www.elastic.co/guide/en/logstash/7.8/plugins-outputs-kafka.html), [RabbitMQ](https://www.elastic.co/guide/en/logstash/7.8/plugins-outputs-rabbitmq.html), and [Amazon SQS](https://www.elastic.co/guide/en/logstash/7.8/plugins-outputs-sqs.html)

### Sensors and IoT

Explore an expansive breadth of other data.

* In this age of technological advancement, the massive IoT world unleashes endless use cases through capturing and harnessing data from connected sensors.
* Logstash is the common event collection backbone for ingestion of data shipped from mobile devices to intelligent homes, connected vehicles, healthcare sensors, and many other industry specific applications.

## Easily Enrich Everything

The better the data, the better the knowledge. Clean and transform your data during ingestion to gain near real-time insights immediately at index or output time. Logstash comes out-of-box with many aggregations and mutations along with pattern matching, geo mapping, and dynamic lookup capabilities.

* [Grok](https://www.elastic.co/guide/en/logstash/7.8/plugins-filters-grok.html) is the bread and butter of Logstash filters and is used ubiquitously to derive structure out of unstructured data. Enjoy a wealth of integrated patterns aimed to help quickly resolve web, systems, networking, and other types of event formats.
* Expand your horizons by deciphering [geo coordinates](https://www.elastic.co/guide/en/logstash/7.8/plugins-filters-geoip.html) from IP addresses, normalizing [date](https://www.elastic.co/guide/en/logstash/7.8/plugins-filters-date.html) complexity, simplifying [key-value pairs](https://www.elastic.co/guide/en/logstash/7.8/plugins-filters-kv.html) and [CSV](https://www.elastic.co/guide/en/logstash/7.8/plugins-filters-csv.html) data, [fingerprinting](https://www.elastic.co/guide/en/logstash/7.8/plugins-filters-fingerprint.html)(anonymizing) sensitive information, and further enriching your data with [local lookups](https://www.elastic.co/guide/en/logstash/7.8/plugins-filters-translate.html) or Elasticsearch [queries](https://www.elastic.co/guide/en/logstash/7.8/plugins-filters-elasticsearch.html).
* Codecs are often used to ease the processing of common event structures like [JSON](https://www.elastic.co/guide/en/logstash/7.8/plugins-codecs-json.html) and [multiline](https://www.elastic.co/guide/en/logstash/7.8/plugins-codecs-multiline.html) events.

See [Transforming Data](https://www.elastic.co/guide/en/logstash/7.8/transformation.html) for an overview of some of the popular data processing plugins.

## Choose Your Stash

Route your data where it matters most. Unlock various downstream analytical and operational use cases by storing, analyzing, and taking action on your data.

**Analysis**

* [Elasticsearch](https://www.elastic.co/guide/en/logstash/7.8/plugins-outputs-elasticsearch.html)
* Data stores such as [MongoDB](https://www.elastic.co/guide/en/logstash/7.8/plugins-outputs-mongodb.html) and [Riak](https://www.elastic.co/guide/en/logstash/7.8/plugins-outputs-riak.html" \t "_top)

**Archiving**

* [HDFS](https://www.elastic.co/guide/en/logstash/7.8/plugins-outputs-webhdfs.html)
* [S3](https://www.elastic.co/guide/en/logstash/7.8/plugins-outputs-s3.html)

**Monitoring**

* [Nagios](https://www.elastic.co/guide/en/logstash/7.8/plugins-outputs-nagios.html)
* [Ganglia](https://www.elastic.co/guide/en/logstash/7.8/plugins-outputs-ganglia.html)
* [Zabbix](https://www.elastic.co/guide/en/logstash/7.8/plugins-outputs-zabbix.html)
* [Graphite](https://www.elastic.co/guide/en/logstash/7.8/plugins-outputs-graphite.html)
* [Datadog](https://www.elastic.co/guide/en/logstash/7.8/plugins-outputs-datadog.html)
* [CloudWatch](https://www.elastic.co/guide/en/logstash/7.8/plugins-outputs-cloudwatch.html)

**Alerting**

* [Watcher](https://www.elastic.co/products/watcher) with Elasticsearch
* [Email](https://www.elastic.co/guide/en/logstash/7.8/plugins-outputs-email.html)
* [Pagerduty](https://www.elastic.co/guide/en/logstash/7.8/plugins-outputs-pagerduty.html)
* [IRC](https://www.elastic.co/guide/en/logstash/7.8/plugins-outputs-irc.html)
* [SNS](https://www.elastic.co/guide/en/logstash/7.8/plugins-outputs-sns.html)

**Beats overview**

Beats are open source data shippers that you install as agents on your servers to send operational data to [Elasticsearch](https://www.elastic.co/products/elasticsearch). Elastic provides Beats for capturing:

|  |  |
| --- | --- |
| Audit data | [Auditbeat](https://www.elastic.co/products/beats/auditbeat) |
| Log files | [Filebeat](https://www.elastic.co/products/beats/filebeat) |
| Cloud data | [Functionbeat](https://www.elastic.co/products/beats/functionbeat) |
| Availability | [Heartbeat](https://www.elastic.co/products/beats/heartbeat) |
| Systemd journals | [Journalbeat](https://www.elastic.co/downloads/beats/journalbeat) |
| Metrics | [Metricbeat](https://www.elastic.co/products/beats/metricbeat) |
| Network traffic | [Packetbeat](https://www.elastic.co/products/beats/packetbeat) |
| Windows event logs | [Winlogbeat](https://www.elastic.co/products/beats/winlogbeat) |

Beats can send data directly to Elasticsearch or via [Logstash](https://www.elastic.co/products/logstash), where you can further process and enhance the data, before visualizing it in [Kibana](https://www.elastic.co/products/logstash).

To get started, see [*Get started with Beats*](https://www.elastic.co/guide/en/beats/libbeat/7.8/getting-started.html).

Want to get up and running quickly with infrastructure metrics monitoring and centralized log analytics? Try out the Metrics app and the Logs app in Kibana. For more details, see the [Metrics Monitoring Guide](https://www.elastic.co/guide/en/metrics/guide/7.8) and the [Logs Monitoring Guide](https://www.elastic.co/guide/en/logs/guide/7.8).

If you have a specific use case to solve, we encourage you to create a [community Beat](https://www.elastic.co/guide/en/beats/libbeat/7.8/community-beats.html). We’ve created an infrastructure to simplify the process. The libbeat library, written entirely in Go, offers the API that all Beats use to ship data to Elasticsearch, configure the input options, implement logging, and more. To learn how to create a new Beat, see the [Beats Developer Guide](http://www.elastic.co/guide/en/beats/devguide/7.8/index.html).

**APM Servers and Agents**

The APM agent installed in your application collects and streams application performance metrics to your APM server, where they are processed and stored in Elasticsearch. In a matter of minutes you can start viewing your performance data either in the dedicated APM app or prebuilt dashboards.

Elastic APM is an application performance monitoring system built on the Elastic Stack. It allows you to monitor software services and applications in real-time, by collecting detailed performance information on response time for incoming requests, database queries, calls to caches, external HTTP requests, and more. This makes it easy to pinpoint and fix performance problems quickly.

Elastic APM also automatically collects unhandled errors and exceptions. Errors are grouped based primarily on the stacktrace, so you can identify new errors as they appear and keep an eye on how many times specific errors happen.

Metrics are another vital source of information when debugging production systems. Elastic APM agents automatically pick up basic host-level metrics and agent-specific metrics, like JVM metrics in the Java Agent, and Go runtime metrics in the Go Agent.

#### APM Agents[edit](https://github.com/elastic/apm-server/edit/7.9/docs/guide/apm-doc-directory.asciidoc)

APM agents are open source libraries written in the same language as your service. You may only need one, or you might use all of them. You install them into your service as you would install any other library. They instrument your code and collect performance data and errors at runtime. This data is buffered for a short period and sent on to APM Server.

Each agent has its own documentation:

* [Go agent](https://www.elastic.co/guide/en/apm/agent/go/1.x/introduction.html)
* [Java agent](https://www.elastic.co/guide/en/apm/agent/java/1.x/intro.html)
* [.NET agent](https://www.elastic.co/guide/en/apm/agent/dotnet/1.x/intro.html)
* [Node.js agent](https://www.elastic.co/guide/en/apm/agent/nodejs/3.x/intro.html)
* [Python agent](https://www.elastic.co/guide/en/apm/agent/python/5.x/getting-started.html)
* [Ruby agent](https://www.elastic.co/guide/en/apm/agent/ruby/3.x/introduction.html)
* [JavaScript Real User Monitoring (RUM) agent](https://www.elastic.co/guide/en/apm/agent/rum-js/5.x/intro.html)

#### APM Server[edit](https://github.com/elastic/apm-server/edit/7.9/docs/guide/apm-doc-directory.asciidoc)

APM Server is an open source application that receives performance data from your APM agents. It’s a [separate component by design](https://www.elastic.co/guide/en/apm/server/7.9/overview.html#why-separate-component), which helps keep the agents light, prevents certain security risks, and improves compatibility across the Elastic Stack.

After the APM Server has validated and processed events from the APM agents, the server transforms the data into Elasticsearch documents and stores them in corresponding [Elasticsearch indices](https://www.elastic.co/guide/en/apm/server/7.9/exploring-es-data.html). In a matter of seconds, you can start viewing your application performance data in the Kibana APM app.

The [APM Server reference](https://www.elastic.co/guide/en/apm/server/7.9/index.html) provides everything you need when it comes to working with the server. Here you can learn more about [installation](https://www.elastic.co/guide/en/apm/server/7.9/getting-started-apm-server.html), [configuration](https://www.elastic.co/guide/en/apm/server/7.9/configuring-howto-apm-server.html), [security](https://www.elastic.co/guide/en/apm/server/7.9/securing-apm-server.html), [monitoring](https://www.elastic.co/guide/en/apm/server/7.9/monitoring.html), and more.