Elasticsearch

Elasticsearch is a distributed search and analytics engine optimized for speed and relevance on production-scale workloads. Elasticsearch is the foundation of Elastic's open Stack platform. Search in near real-time over massive datasets, perform vector searches, integrate with generative AI applications, and much more.

Use cases enabled by Elasticsearch include:

- Retrieval Augmented Generation (RAG)
- Vector search
- Full-text search
- Logs
- Metrics
- Application performance monitoring (APM)
- Security logs

... and more!

To learn more about Elasticsearch's features and capabilities, see our product page.

To access information on machine learning innovations and the latest Lucene contributions from Elastic, more information can be found in Search Labs.

Get started

The simplest way to set up Elasticsearch is to create a managed deployment with Elasticsearch Service on Elastic Cloud.

If you prefer to install and manage Elasticsearch yourself, you can download the latest version from elastic.co/downloads/elasticsearch.

Run Elasticsearch locally

To try out Elasticsearch on your own machine, we recommend using Docker and running both Elasticsearch and Kibana. Docker images are available from the Elastic Docker registry.

NOTE	Starting in Elasticsearch 8.0, security is enabled by default. The first time you start Elasticsearch, TLS encryption is configured automatically, a password is generated for the `elastic` user, and a Kibana enrollment token is created so you can connect Kibana to your secured cluster.

For other installation options, see the Elasticsearch installation documentation.

Start Elasticsearch

1. Install and start Docker Desktop. Go to **Preferences** > **Resources** > **Advanced** and set Memory to at least 4GB.
2. Start an Elasticsearch container:

```
docker network create elastic
docker pull docker.elastic.co/elasticsearch/elasticsearch:{version} ①
docker run --name elasticsearch --net elastic -p 9200:9200 -p 9300:9300 -e
"discovery.type=single-node" -t
docker.elastic.co/elasticsearch/elasticsearch:{version}
```

① Replace {version} with the version of Elasticsearch you want to run.

When you start Elasticsearch for the first time, the generated `elastic` user password and Kibana enrollment token are output to the terminal.

NOTE	You might need to scroll back a bit in the terminal to view the password and enrollment token.

3. Copy the generated password and enrollment token and save them in a secure location. These values are shown only when you start Elasticsearch for the first time. You'll use these to enroll Kibana with your Elasticsearch cluster and log in.

Start Kibana

Kibana enables you to easily send requests to Elasticsearch and analyze, visualize, and manage data interactively.

1. In a new terminal session, start Kibana and connect it to your Elasticsearch container:

```
docker pull docker.elastic.co/kibana/kibana:{version} ①
docker run --name kibana --net elastic -p 5601:5601
docker.elastic.co/kibana/kibana:{version}
```

① Replace {version} with the version of Kibana you want to run.

When you start Kibana, a unique URL is output to your terminal.

2. To access Kibana, open the generated URL in your browser.
   a. Paste the enrollment token that you copied when starting Elasticsearch and click the button to connect your Kibana instance with Elasticsearch.
   b. Log in to Kibana as the `elastic` user with the password that was generated when you started Elasticsearch.

Send requests to Elasticsearch

You send data and other requests to Elasticsearch through REST APIs. You can interact with

Elasticsearch using any client that sends HTTP requests, such as the Elasticsearch language clients and curl. Kibana's developer console provides an easy way to experiment and test requests. To access the console, go to **Management** > **Dev Tools**.

Add data

You index data into Elasticsearch by sending JSON objects (documents) through the REST APIs. Whether you have structured or unstructured text, numerical data, or geospatial data, Elasticsearch efficiently stores and indexes it in a way that supports fast searches.

For timestamped data such as logs and metrics, you typically add documents to a data stream made up of multiple auto-generated backing indices.

To add a single document to an index, submit an HTTP post request that targets the index.

```
POST /customer/_doc/1
{
  "firstname": "Jennifer",
  "lastname": "Walters"
}
```

This request automatically creates the `customer` index if it doesn't exist, adds a new document that has an ID of 1, and stores and indexes the `firstname` and `lastname` fields.

The new document is available immediately from any node in the cluster. You can retrieve it with a GET request that specifies its document ID:

```
GET /customer/_doc/1
```

To add multiple documents in one request, use the `_bulk` API. Bulk data must be newline-delimited JSON (NDJSON). Each line must end in a newline character (`\n`), including the last line.

```
PUT customer/_bulk
{ "create": { } }
{ "firstname": "Monica","lastname":"Rambeau"}
{ "create": { } }
{ "firstname": "Carol","lastname":"Danvers"}
{ "create": { } }
{ "firstname": "Wanda","lastname":"Maximoff"}
{ "create": { } }
{ "firstname": "Jennifer","lastname":"Takeda"}
```

Search

Indexed documents are available for search in near real-time. The following search matches all customers with a first name of *Jennifer* in the `customer` index.

```
GET customer/_search
{
  "query" : {
    "match" : { "firstname": "Jennifer" }
  }
}
```

Explore

You can use Discover in Kibana to interactively search and filter your data. From there, you can start creating visualizations and building and sharing dashboards.

To get started, create a *data view* that connects to one or more Elasticsearch indices, data streams, or index aliases.

1. Go to **Management** > **Stack Management** > **Kibana** > **Data Views**.
2. Select **Create data view**.
3. Enter a name for the data view and a pattern that matches one or more indices, such as *customer*.
4. Select **Save data view to Kibana**.

To start exploring, go to **Analytics** > **Discover**.

Upgrade

To upgrade from an earlier version of Elasticsearch, see the Elasticsearch upgrade documentation.

Build from source

Elasticsearch uses Gradle for its build system.

To build a distribution for your local OS and print its output location upon completion, run:

```
./gradlew localDistro
```

To build a distribution for another platform, run the related command:

```
./gradlew :distribution:archives:linux-tar:assemble
./gradlew :distribution:archives:darwin-tar:assemble
./gradlew :distribution:archives:windows-zip:assemble
```

To build distributions for all supported platforms, run:

```
./gradlew assemble
```

Distributions are output to `distribution/archives`.

To run the test suite, see TESTING.

Documentation

For the complete Elasticsearch documentation visit elastic.co.

For information about our documentation processes, see the docs README.

Examples and guides

The `elasticsearch-labs` repo contains executable Python notebooks, sample apps, and resources to test out Elasticsearch for vector search, hybrid search and generative AI use cases.

Contribute

For contribution guidelines, see CONTRIBUTING.

Questions? Problems? Suggestions?

- To report a bug or request a feature, create a GitHub Issue. Please ensure someone else hasn't created an issue for the same topic.
- Need help using Elasticsearch? Reach out on the Elastic Forum or Slack. A fellow community member or Elastic engineer will be happy to help you out.