# Elasticsearch Search and analytics engine

**START HERE**

`curl -s $ES`
Is it up? Version

`curl --json '<json>' $ES/<index>/_doc`
Add a document

`curl -s $ES/_cat/indices?v`
List indices

`curl -s $ES/<index>/_search?q=<text>`
Quick search

`curl -s $ES/_cluster/health?pretty`
Cluster status

`curl -s $ES/<index>/_doc/<id>`
Fetch by id

`curl -X PUT $ES/<index>`
Create an index

`curl -X DELETE $ES/<index>`
Delete an index

## CONCEPTS

**Reading this sheet**
Entries are METHOD path. Run them as curl -X <METHOD> $ES/<path>, with --json '<body>' when a body is given (curl 7.82+ sets the JSON headers), or paste them into Kibana Dev Tools. Set ES=localhost:9200 first.

Elasticsearch 8+ enables security by default: use ES=https://localhost:9200 and add -u elastic:<password> --cacert <http_ca.crt> to every curl.

**Cluster** nodes, one health status
**Index** mapping + settings
**Shard** primary + replicas
**Document** JSON with an _id

Health: green = all shards allocated, yellow = replicas missing, red = a primary is missing.

New documents become searchable after the next refresh (about 1 s by default), not instantly.

## CLUSTER AND CAT

`GET /`
Node name, cluster, version

`GET _cluster/health?pretty`
Green / yellow / red status

`GET _cat/indices?v`
Indices with docs and size

`GET _cat/nodes?v`
Nodes and load

`GET _cat/shards/<index>?v`
Where shards live

`GET _cat/indices?v&s=store.size:desc`
Sort cat output by a column

`GET _cat/indices?help`
List cat columns

`GET _cluster/allocation/explain`
Why a shard is unassigned

`GET _nodes/stats`
Per-node statistics

`GET _tasks?detailed`
Running tasks

## INDICES

`PUT <index>`
Create an index
*A body may hold settings and mappings.*

`DELETE <index>`
Delete an index and its data

`GET <index>/_mapping`
Show field mappings

`GET <index>/_settings`
Show index settings

`PUT <index>/_mapping`
Add fields to a mapping
*Existing field types cannot change; reindex into a new index instead.*

`PUT <index>/_settings`
Change a dynamic setting

`POST <index>/_refresh`
Make recent writes searchable

`POST <index>/_close`
Close an index

`POST _aliases`
Add or swap aliases atomically

## DOCUMENTS

`POST <index>/_doc`
Index with a generated id

`PUT <index>/_doc/<id>`
Index or replace by id

`GET <index>/_doc/<id>`
Get a document

`DELETE <index>/_doc/<id>`
Delete a document

`POST <index>/_update/<id>`
Partial update: {"doc":{...}}

`GET <index>/_count`
Count documents

`GET <index>/_source/<id>`
Only the _source

`POST <index>/_delete_by_query`
Delete all matching a query

`POST <index>/_update_by_query`
Update all matching a query

`GET _mget`
Get several by id

## SEARCH

`GET <index>/_search?q=<field>:<text>`
URI search for quick checks

`POST <index>/_search`
Search with a query body

## SEARCH (cont.)

`{"match": {"<field>": "<text>"}}`
Full-text match

`{"term": {"<field>": "<value>"}}`
Exact value (keyword, number)
*term does not analyze: on a text field use match, or the .keyword subfield.*

`{"bool": {"must": [], "filter": [], "must_not": []}}`
Combine clauses
*filter clauses do not score and are cached; use them for yes/no conditions.*

`{"range": {"<field>": {"gte": 10, "lt": 20}}}`
Range filter

`{"exists": {"field": "<field>"}}`
Field is present

`{"match_phrase": {"<field>": "<text>"}}`
Exact phrase

`"size": 20, "from": 40`
Paginate

`"sort": [{"<field>": "desc"}]`
Sort results

`"_source": ["<field>"]`
Return only some fields

`"explain": true`
Show scoring details

`"pit": {...}, "search_after": [...]`
Deep paging with PIT

`POST <index>/_validate/query?explain`
Check a query without running

## AGGREGATIONS

`"size": 0, "aggs": {"by": {"terms": {"field": "<f>"}}}`
Top values (terms)
*Use a keyword field; size 0 skips the hits.*

`{"avg": {"field": "<field>"}}`
Metric: avg, min, max, sum

`{"date_histogram": {"field": "<f>", "calendar_interval": "day"}}`
Count per day

## AGGREGATIONS (cont.)

`{"percentiles": {"field": "<field>"}}`
Percentiles

`{"cardinality": {"field": "<field>"}}`
Approximate distinct count

## BULK AND REINDEX

`POST _bulk`
Many writes, NDJSON body
*Action line, then source line; the body must end with a newline. Send with --data-binary and Content-Type application/x-ndjson.*

`POST _reindex`
Copy: {"source":{...},"dest":{...}}
*Add ?wait_for_completion=false to run it as a task, then poll GET _tasks/<task_id>.*

`GET _tasks/<task_id>`
Poll a task

## RECIPES

### Create index, bulk load, search

```
curl -X PUT $ES/books --json
'{"mappings": {"properties":
{"year": {"type": "integer"}}}}'
curl -X POST "$ES/books/_bulk?
refresh" \
  -H 'Content-Type: application/
x-ndjson' --data-binary
@books.ndjson
curl -s "$ES/books/_search?q=
title:dune&pretty"
```

### Swap to a new index behind an alias

```
# alias books → books_v1
curl -X PUT $ES/books_v2 --json
@mapping.json
curl -X POST $ES/_reindex --json
\
  '{"source": {"index":
"books_v1"}, "dest": {"index":
"books_v2"}}'
curl -X POST $ES/_aliases --json
@swap.json
```

### Filtered search with a count per genre

```
curl -s $ES/books/_search --json
'{
  "query": {"bool": {"must":
{"match": {"title": "dune"}},
    "filter": {"range": {"year":
{"gte": 1965}}}}},
  "aggs": {"genre": {"terms":
{"field": "genre"}}}}'
```

### Pull just the ids with jq

```
curl -s "$ES/books/_search?
size=100" \
  | jq -r '.hits.hits[]._id'
```