Elasticsearch Engineering – Digest

*The short companion to **Elasticsearch Engineering**: a glossary, then the facts worth memorising for the Elastic Certified Engineer exam (Elasticsearch 9.3), domain by domain. Use it for revision; go back to the full guide for explanations, examples and doc links.*

A. Glossary

Term	Meaning (one line)
Alias	A secondary name pointing to one or more indices or data streams; can filter, route and have one write index
Allocation (shard)	The master's decision about which node holds each shard copy, governed by *deciders*
Allocation awareness	Spreads shard copies across node attributes (zone, rack); *forced* awareness refuses to overload surviving zones
Allocation explain API	`GET _cluster/allocation/explain`: says why a shard is (un)assigned
Analyzer	Char filters → tokenizer → token filters; turns text into terms at index and search time
API key	Credential with its own (restricted) privileges; `Authorization: ApiKey <encoded>`
Auto-follow pattern	CCR rule that automatically follows *new* leader indices matching patterns
Bool query	Compound query with `must`, `should`, `filter`, `must_not`
Bootstrap checks	Startup checks enforced in production mode (heap, file descriptors, `vm.max_map_count`, memory lock…)
Bucket aggregation	Groups documents into buckets (`terms`, `date_histogram`, `range`, `filters`, `composite`…)
Bulk API	NDJSON batch of index/create/update/delete actions in one request
CCR	Cross-cluster replication: read-only follower indices pull changes from leader indices on a remote cluster

Term	Meaning (one line)
CCS	Cross-cluster search: query `remote_alias:index` from the local cluster
Classic stream	A Streams-managed view over an *existing* data stream (no migration)
Cluster state	Metadata (indices, mappings, settings, routing table) managed by the elected master
Component template	Reusable settings/mappings/aliases block referenced by index templates
Composable index template	`_index_template`: patterns + priority + `composed_of` + own template; can create data streams
Coordinating node	The node receiving a request; fans it out and merges results (a node with `roles: []` does only this)
`copy_to`	Copies a field's value into another field at index time
Cross-cluster API key	API key created on the remote (`/_security/cross_cluster/api_key`) that scopes CCS/CCR access
Data stream	Append-only name over hidden backing indices; needs `@timestamp` and a template with `data_stream: {}`
Data stream lifecycle (DSL)	Simpler alternative to ILM for data streams: retention (+ downsampling)
Data tier	Node role groups `data_hot` / `warm` / `cold` / `frozen` / `content`; placement via `_tier_preference`
Dense vector	Fixed-length float embedding; searched with `knn` (HNSW)
Dissect	Delimiter-based parsing (faster, simpler than grok)
DLS / FLS	Document-level security (role `query`) / field-level security (`field_security.grant/except`)
Doc values	On-disk columnar structure used for sorting, aggregations and scripts
Dynamic mapping	Auto-mapping of unknown fields; controlled by `dynamic: true/false/strict/runtime`
Dynamic template	Rule mapping *new* fields by detected type, name or path; first match wins
ELSER	Elastic Learned Sparse EncodeR: Elastic's sparse-embedding model for semantic search
Enrich policy / processor	Lookup table built from a source index (`_execute`) and applied at ingest
ES|QL	Piped query language: `FROM ... | WHERE ... | EVAL ... | STATS ... BY ... | SORT ... | LIMIT ...`
Failure store	Per-data-stream store that keeps documents that failed ingestion (`<ds>::failures`)

Term	Meaning (one line)
Filter context	Yes/no matching with no scoring, cacheable (`filter`, `must_not`)
Flattened	Maps an entire object as one field of keywords (avoids mapping explosion)
Flood-stage watermark	95% disk: indices on that node get `read_only_allow_delete`
Follower / leader index	CCR copy (read-only) and its source
Grok	Regex-based parsing with named patterns (`%{IP:client.ip}`)
Health report	`GET _health_report`: indicator-based health with diagnoses
Hybrid search	Combining lexical (BM25) and semantic results, e.g. with an RRF retriever
ILM	Index lifecycle management: hot → warm → cold → frozen → delete phases with actions
Index	Logical collection of documents with one mapping and settings, split into shards
Index mode	`standard`, `logsdb`, `time_series` (TSDS), `lookup`
Inference endpoint	`_inference/<task>/<id>`: a configured model/service used to create embeddings
Ingest pipeline	Ordered processors that transform documents before indexing
Keystore	`elasticsearch-keystore`: encrypted store for secure settings
kNN	k-nearest-neighbour vector search (`knn` section or query)
LOOKUP JOIN	ES|QL join against an index with `index.mode: lookup`
Mapping	Schema of an index: fields, types, analyzers, parameters
Master-eligible node	Can be elected master; production uses 3
Metric aggregation	Computes values (`avg`, `sum`, `cardinality`, `percentiles`, `stats`, `top_hits`…)
Multi-field	Same value indexed in several ways (`fields: { raw: keyword }`)
Nested	Field type keeping array-of-object members as separate hidden docs; needs `nested` query/agg
Normalizer	Analyzer-like chain for `keyword` fields (no tokenizer): lowercase, asciifolding
Pipeline aggregation	Aggregation over other aggregations' outputs (`derivative`, `bucket_selector`, `max_bucket`…)
Point in time (PIT)	Frozen view of indices for consistent deep pagination with `search_after`
Primary / replica shard	The original / copy of a shard; never on the same node

Term	Meaning (one line)
Query context	Scored matching ("how well"): `must`, `should`
Realm	Authentication source: native, file, LDAP, AD, SAML, OIDC, PKI, Kerberos, JWT
Refresh	Makes recent writes searchable (default 1s)
Reindex	Copies documents from a source to a destination index (optionally with query/script/pipeline)
Remote cluster	Named connection (sniff or proxy mode; API key on port 9443, or certificate) used by CCS/CCR
Reroute processor	Ingest processor sending a doc to another data stream
Retriever	Composable search building block: `standard`, `knn`, `rrf`, `linear`, `text_similarity_reranker`
Role / role mapping	Named bundle of privileges / rule assigning roles to externally authenticated users
Rollover	Creating a new write index when conditions (age, size, docs) are met
RRF	Reciprocal Rank Fusion: merges ranked lists without score normalisation
Runtime field	Field computed at query time from a script; no reindex needed
`search_after`	Cursor-based deep pagination using the last hit's sort values
Searchable snapshot	Index mounted from a snapshot repository (fully mounted = cold, partially = frozen)
Segment	Immutable Lucene sub-index inside a shard; merged over time
`semantic_text`	Field type that auto-chunks and embeds text through an inference endpoint
Service account	Built-in identity for Elastic services (e.g. `elastic/kibana`) using service tokens
Shard	A Lucene index holding part of an Elasticsearch index
SLM	Snapshot lifecycle management: scheduled snapshots with retention
Snapshot	Incremental backup of indices or cluster state into a repository
Sparse vector	Token → weight embedding (e.g. ELSER)
Streams	Kibana feature to process, partition, map, retain and monitor log data from one UI/API
Synonyms API	`_synonyms/<set>`: manage synonym sets usable by updateable search-time analyzers

Term	Meaning (one line)
Term-level query	Exact, non-analysed matching (`term`, `terms`, `range`, `exists`, `prefix`…)
Translog	Per-shard write-ahead log for durability between flushes
Update By Query	Updates matching docs in place (script or pipeline)
Watermarks (disk)	Low 85% / high 90% / flood 95%
Wired stream	Streams-managed hierarchy rooted at `logs` (9.2–9.3) with partitioning and inheritance

B. Key things to remember

0. Exam mechanics

- **ES 9.3**, about 3 hours, about 10–12 hands-on tasks, partial credit, **only the Elastic docs** allowed.
- Grading is based on **final cluster state**: exact names matter, don't delete extras, and verify with `GET` after every task.
- The Console shortcut `Ctrl/Cmd+/` opens the docs for the API under the cursor.

1. Architecture

- The stack is **Collect** (Agent/Fleet, Beats, OTel, Logstash) → **Process** (pipelines, Logstash, Streams) → **Store/Search** (Elasticsearch) → **Explore** (Kibana).
- **Routing** is `hash(_routing) % shards`, so the primary count is fixed. Change it with `_split`, `_shrink` or reindex.
- **3 master-eligible nodes** for quorum; `cluster.initial_master_nodes` is used only on the first bootstrap.
- **Shard size 10–50 GB**, **< 200M docs**. `cluster.max_shards_per_node` = **1000**.
- **Watermarks: 85 / 90 / 95%.** Flood stage sets `index.blocks.read_only_allow_delete`.
- **Forced awareness** keeps replicas **unassigned** rather than overloading a surviving zone, so a yellow cluster can be correct.
- Allocation filters: `include` = any, `exclude` = none, `require` = all. Attributes: `_name`, `_ip`, `_host`, `_tier`, and custom `node.attr.*`.
- `index.unassigned.node_left.delayed_timeout` = **1m**. For rolling restarts, use `cluster.routing.allocation.enable: primaries`.
- Content indices go to `data_content`; data stream backing indices go to `data_hot`.

2. Data Management

- **Static settings**: shards, codec, `index.mode`, sort, and analysis (close → update → open). **Dynamic**: replicas, refresh, allocation, blocks, default/final pipeline.
- `dynamic`: `strict` rejects, `false` keeps in `_source` but doesn't index, `runtime` creates runtime fields.
- **Template precedence**: components (in order) < index template `template` < create-index request.
- **One template per index**: the highest `priority` wins. Equal priority with overlapping patterns is an error.
- **Built-in templates at priority 100** (`logs-*-*`, `metrics-*-*`, `logs.otel*`…). Use **> 500** to override, or avoid overlapping.
- Test with `_index_template/_simulate_index/<name>`. Templates affect **new** indices only; roll over data streams.
- **Dynamic templates**: first match wins, so put specific rules first and the catch-all last. `match` = leaf name, `path_match` = full path. There's no `integer` detection (`long`/`double`).
- **Data streams**: need a template with `data_stream: {}` and `@timestamp`. Append-only (`create` op). Deleting the stream deletes its backing indices.
- **Failure store**: `data_stream_options.failure_store.enabled: true`; read `<ds>::failures`.
- **ILM**: `min_age` counts from **rollover**. Rollover conditions are OR-ed; empty indices never roll over. Poll interval is 10m.
- Rollover needs a data stream **or** `rollover_alias` + a bootstrap `-000001` with `is_write_index: true`.
- `migrate` is implicit in warm and cold; with no tier nodes, the shards don't move.
- Frozen **requires** `searchable_snapshot`. Shrink and forcemerge in hot only with rollover.
- Debug with `_ilm/explain`, then `_ilm/retry`.
- **DSL vs ILM**: when both are set, ILM wins (`prefer_ilm`). DSL is for simple retention only.
- **Aliases**: `_aliases` actions are **atomic** (`add`, `remove`, `remove_index`). Multi-index aliases need one `is_write_index` to accept writes. Filtered aliases use `filter` + `routing`.

3. Ingestion & Processing

- **Choose the ingest method**: Agent + Fleet for managed hosts and integrations, Logstash for heavy transforms, buffering or multiple outputs, client + `_bulk` for applications, Streams for parsing logs from the UI, file upload for one-offs.
- **Bulk**: NDJSON with a final newline; check `errors`; 5–15 MB batches.
- **Immutable once mapped**: type, analyzer, index/doc_values, nested vs object. You *can* add fields, add multi-fields, and add runtime fields.
- New multi-field → `_update_by_query` to populate existing docs.
- `text` for full text, `keyword` for exact/sort/aggs, both via multi-field. Use a `normalizer` for case-insensitive keyword.
- `nested` keeps object pairs together; it needs a `nested` query/agg. Limits: 50 nested fields, 10k nested objects per doc.

- Limits: `total_fields.limit` **1000**, `depth.limit` 20. Use `flattened` for unbounded keys.
- `scaled_float` for prices, `constant_keyword` for per-index constants, `wildcard` for grep-style search, `match_only_text` for logs, `date_nanos` for ns precision.
- Analyzer order: **char_filter → tokenizer → filter**. Test with `_analyze`. Put edge n-grams at index time only, with `search_analyzer: standard`.
- Search-time synonyms: `synonym_graph` + Synonyms API + `updateable: true`.
- **Reindex**: create the destination mapping **first**; `op_type: create` for data streams; `conflicts: proceed`; `wait_for_completion=false` → `_tasks`; remote reindex needs `reindex.remote.whitelist`.
- Painless: `ctx._source.x` in reindex/UBQ scripts vs `ctx.x` in ingest; `ctx.op = 'noop'|'delete'`; null-safe `?.`.
- **Pipelines**: `if` conditions, `ignore_failure`, `ignore_missing`, and processor or pipeline `on_failure` (`_ingest.on_failure_message`). Always `_simulate?verbose`.
- `default_pipeline` can be overridden by `?pipeline=`; **`final_pipeline` always runs**.
- `_ingest.timestamp` is the ingest time. The `reroute` processor changes the target data stream. Use `@custom` pipelines to customise integrations.
- **Enrich**: create the policy → `_execute` → `enrich` processor. Policies are immutable (delete and recreate), and must be re-executed when the source changes.
- **Streams**: *classic* = existing data streams in place; *wired* = hierarchy rooted at `logs` (9.2–9.3) with **partitioning** and **inheritance**. The tabs are Processing (with preview), Schema, Data lifecycle (inherit / custom / ILM), Data quality (degraded vs failed → failure store) and Advanced. **Don't edit managed components with raw APIs.** The API lives under `kbn:/api/streams`.

4. Searching Data

- **`match`** is analysed (OR by default; `operator: and`, `minimum_should_match`, `fuzziness: AUTO`). **`term`** is exact. Never use `term` on a `text` field; use `.keyword`.
- `match_phrase` + `slop`; `multi_match` with `fields: ["title^3"]`, types `best_fields` (default), `most_fields`, `cross_fields`, `phrase`, `bool_prefix`.
- **Bool**: `must` (scored), `filter` (unscored, cached), `should` (scored, optional **if** a `must`/`filter` exists, otherwise at least 1 required), `must_not` (unscored).
- `post_filter` filters hits but not aggregations (facets).
- `from + size` ≤ **10,000** (`max_result_window`); for deeper pages, use **`search_after` + PIT**. You can't sort on `text`.
- Aggregations: set `size: 0`; use `terms` on keyword only; `terms` counts are approximate (`shard_size`); use `composite` to page through all buckets.
- `date_histogram`: `calendar_interval` (1d, 1M, week…) vs `fixed_interval` (30m, 7d); `min_doc_count: 0` + `extended_bounds` for empty buckets; `time_zone`.
- `range` agg: `from` is inclusive, `to` exclusive.
- `buckets_path`: `>` for nesting, `.` for a metric's value, `_count` for doc count. `bucket_selector` works like HAVING; `bucket_sort` sorts and truncates buckets.
- **ES|QL**: `POST _query` (`?format=txt`). The **default LIMIT is 1000** (max 10k). `STATS … BY`, `EVAL`, `KEEP`/`DROP`/`RENAME`, `DISSECT`/`GROK`, `ENRICH`, `LOOKUP JOIN` (the lookup index needs `index.mode: lookup`), `BUCKET()`, `MATCH()` or `:` for full text, `METADATA _index, _score`, `?`

params, and `remote:index` for CCS.
- **Semantic search**: an inference endpoint (a default ELSER one exists) + a `semantic_text` field. Query with `match` or `semantic` on it; highlight with `type: semantic`. Hybrid uses the `rrf` **retriever** (`rank_window_size`, `rank_constant`). Manual path: `dense_vector` + `knn` (`k`, `num_candidates`, `filter`) or `sparse_vector` query. It needs ML capacity and a licence (trial in the lab).
- **CCS**: `alias:index`, `*:index`, exclusions `-alias:*`. `skip_unavailable` defaults to **true**; `ccs_minimize_roundtrips` defaults to true. `_resolve/cluster` checks connectivity. `_clusters` in the response shows per-cluster status.

5. Cluster Management

- **Green** / **yellow** / **red** = all assigned / replica(s) unassigned / primary unassigned.
- The workflow: `_cluster/health` → `_cat/shards` → `_cluster/allocation/explain` → fix → `_cluster/reroute?retry_failed=true` if max retries was hit.
- Common causes: too few nodes for the replicas, allocation filters, awareness, disk watermarks, a missing tier, `allocation.enable` set to none/primaries, max retries.
- Red with all copies lost: restore from a snapshot. `allocate_empty_primary` / `allocate_stale_primary` (with `accept_data_loss`) is the last resort.
- `persistent` cluster settings survive restarts; `transient` is deprecated. `null` resets a setting.
- Monitor heap (> 85% is bad), breakers, thread-pool rejections, disk, slow logs and hot threads. Use Stack Monitoring (Agent/Metricbeat → monitoring cluster) or AutoOps.
- **Snapshots**: `fs` repos need `path.repo` on **every** node. Snapshots are incremental. Restore needs the target closed or absent; use `rename_pattern`/`rename_replacement` otherwise. `include_global_state` and `feature_states` control what's restored.
- **SLM**: Quartz cron (seconds first), date-math names `<nightly-{now/d}>`, retention (`expire_after`, `min_count`, `max_count`), `_execute` to run now.
- **Remote clusters**: the **API-key model is recommended** (remote port **9443**, `remote_cluster_server.enabled`, cross-cluster API key with `access.search`/`access.replication`, local keystore `cluster.remote.<alias>.credentials`). The **certificate model is deprecated** (port 9300, same-named roles on the remote). Modes are **sniff** (default) and **proxy**. Check with `GET _remote/info`. Local nodes need `remote_cluster_client`.
- The local role grants remote access through **`remote_indices`**.
- **CCR**: licence on both clusters; run on the **follower** cluster: `PUT <follower>/_ccr/follow {remote_cluster, leader_index}`. Auto-follow uses `leader_index_patterns` + `follow_index_pattern: "{{leader_index}}-x"` and covers **new** indices only. Followers are read-only. Failover (promote a follower): `pause_follow` → `_close` → `unfollow` → `_open`. Soft deletes must be on (the default).

6. Security & Access Control

- A role has `cluster` + `indices` (`names`, `privileges`, `field_security`, `query`) + `applications` (Kibana: `kibana-.kibana`, `feature_x.read/all`, `space:<id>`) + `remote_indices`.

- **A user's roles are a union**: one role with unrestricted `read` cancels another role's DLS.
- Key privileges: `read` + `view_index_metadata` for Kibana data views; `create_doc` + `auto_configure` for shippers writing to data streams; `manage_security` is effectively admin.
- DLS templating: `{{_user.username}}`. FLS: `grant` / `except` with wildcards.
- **API keys** get the intersection of their `role_descriptors` and the creator's privileges; use `expiration`; invalidate with `DELETE _security/api_key`.
- **Role mappings** use `rules` (`field` / `any` / `all` / `except` on `groups`, `dn`, `username`, `realm.name`) for LDAP, AD, SAML and OIDC users.
- Test with `_security/_authenticate`, `_security/user/_has_privileges` and the `es-security-runas-user` header.
- **Production security**: security on; **transport TLS is mandatory** for multi-node clusters with security; HTTPS for clients and Kibana; secrets in the **keystore**; Kibana via a **service account token**; no daily use of `elastic`.
- Tools: `elasticsearch-certutil ca|cert|http`, `elasticsearch-reset-password -u <user>`, `elasticsearch-create-enrollment-token -s kibana|node`, `elasticsearch-service-tokens create`, `elasticsearch-keystore add`, `POST _nodes/reload_secure_settings`.
- **Bootstrap checks** (production mode starts when you bind a non-loopback `network.host`): heap `Xms = Xmx`, ≤ 50% RAM, < ~31 GB; file descriptors ≥ **65,535**; threads ≥ **4,096**; `vm.max_map_count` ≥ **262,144**; memory lock works if enabled; discovery configured; swap disabled.
- **Audit logging** (`xpack.security.audit.enabled`, `events.include`), **IP filtering** (`xpack.security.transport|http.filter.allow/deny`), and `action.destructive_requires_name: true`.
- Kibana needs a 32+ character `xpack.encryptedSavedObjects.encryptionKey` for alerting and encrypted saved objects.

C. Ten-second API recall

Need	API
Health & diagnosis	`_cluster/health`, `_health_report`, `_cat/shards`, `_cluster/allocation/explain`, `_cluster/reroute?retry_failed=true`
Templates	`_component_template`, `_index_template`, `_index_template/_simulate_index/<n>`
Data streams	`_data_stream/<n>`, `<ds>/_rollover`, `<ds>::failures`, `_data_stream/<n>/_lifecycle`
ILM	`_ilm/policy/<p>`, `<idx>/_ilm/explain`, `<idx>/_ilm/retry`
Aliases	`POST _aliases {actions:[add/remove/remove_index]}`
Mapping & analysis	`<idx>/_mapping`, `<idx>/_analyze`, `_synonyms/<set>`
Move data	`_reindex`, `<idx>/_update_by_query`, `<idx>/_delete_by_query`, `_tasks/<id>`
Pipelines	`_ingest/pipeline/<p>`, `.../_simulate?verbose`, `_ingest/_simulate`, `_enrich/policy/<p>/_execute`

Need	API
Search	`<idx>/_search`, `_async_search`, `<idx>/_pit`, `_query` (ES|QL), `_resolve/cluster/<pattern>`
Semantic	`_inference/<task>/<id>`, `semantic_text`, `semantic` query, `rrf` retriever
Snapshots	`_snapshot/<repo>`, `_snapshot/<repo>/<snap>/_restore`, `_slm/policy/<p>/_execute`
Remote / CCR	`_remote/info`, `_security/cross_cluster/api_key`, `<f>/_ccr/follow`, `_ccr/auto_follow/<n>`, `<f>/_ccr/unfollow`
Security	`_security/role/<r>`, `_security/user/<u>`, `_security/api_key`, `_security/role_mapping/<m>`, `_security/_authenticate`, `_security/user/_has_privileges`

Revision #1
Created 28 September 2026 14:29:01 by N8N
Updated 28 September 2026 14:29:01 by N8N