**🔥 Elasticsearch Complete Guide (From Basic to Mastery)**

**1. Introduction to Elasticsearch**

* **Definition**: Elasticsearch is a **distributed search and analytics engine** built on top of **Apache Lucene**.
* It stores **JSON documents** and provides **full-text search**, **real-time indexing**, and **scalability**.
* **Part of ELK Stack**: Elasticsearch + Logstash + Kibana.

📌 **Interview Point**: "Elasticsearch is not a database replacement, it’s a search and analytics engine."

**2. Core Concepts**

* **Cluster** → A set of nodes working together.
* **Node** → A single Elasticsearch instance.
* **Index** → A collection of documents (like a database).
* **Document** → Basic unit of data, stored as JSON.
* **Shard** → An index is divided into shards for scaling.
* **Replica** → Copy of a shard for fault tolerance.

👉 Example:

{

"id": 1,

"title": "Learning Elasticsearch",

"author": "Sonali"

}

📌 **Interview Point**: "An index is divided into primary and replica shards. Replicas help in high availability."

**3. Installation & Setup**

* Download from [elastic.co](https://www.elastic.co/downloads/elasticsearch).
* Run: ./bin/elasticsearch
* Default port: 9200.

Check health:

curl -X GET "localhost:9200/\_cluster/health?pretty"

**4. CRUD Operations**

👉 Create/Index Document:

PUT my\_index/\_doc/1

{

"name": "Sonali",

"role": "Developer"

}

👉 Read Document:

GET my\_index/\_doc/1

👉 Update Document:

POST my\_index/\_update/1

{

"doc": { "role": "Senior Developer" }

}

👉 Delete Document:

DELETE my\_index/\_doc/1

📌 **Interview Point**: Elasticsearch is **schema-less** but allows **mappings** for defining structure.

**5. Mapping & Data Types**

* **Mapping** = Schema definition.
* Data types: text, keyword, date, long, boolean.

👉 Example:

PUT my\_index

{

"mappings": {

"properties": {

"name": { "type": "text" },

"age": { "type": "integer" },

"dob": { "type": "date" }

}

}

}

**6. Search (DSL - Query Language)**

👉 Match Query:

GET my\_index/\_search

{

"query": {

"match": { "name": "Sonali" }

}

}

👉 Term Query (exact match):

GET my\_index/\_search

{

"query": {

"term": { "role.keyword": "Developer" }

}

}

👉 Bool Query:

GET my\_index/\_search

{

"query": {

"bool": {

"must": [{ "match": { "role": "Developer" }}],

"filter": [{ "term": { "age": 25 }}]

}

}

}

📌 **Interview Point**:

* match → Full-text search.
* term → Exact search.
* bool → Combine multiple queries.

**7. Aggregations (Analytics)**

👉 Count average age:

GET my\_index/\_search

{

"size": 0,

"aggs": {

"avg\_age": { "avg": { "field": "age" } }

}

}

👉 Group by role:

GET my\_index/\_search

{

"size": 0,

"aggs": {

"roles": { "terms": { "field": "role.keyword" } }

}

}

📌 **Interview Point**: Aggregations are like SQL GROUP BY.

**8. Advanced Features**

* **Pagination**: from + size
* **Sorting**: sort
* **Highlighting**: highlight matching text
* **Aliases**: Virtual name for an index
* **Reindex API**: Copy data between indices

**9. Cluster Management**

* Health Check:
  + Green → All good.
  + Yellow → Replicas missing.
  + Red → Some primary shards missing.

👉 Check:

GET \_cluster/health

📌 **Interview Point**: Elasticsearch ensures **horizontal scalability** with shards.

**10. Real-World Use Cases**

* E-commerce → Product search (Amazon, Flipkart).
* Logging → With Logstash + Kibana.
* Analytics → User behavior analysis.
* Autocomplete → Typeahead suggestions.

**⏰ 2-Hour Study Plan**

✅ **0–20 min** → Basics (Concepts + Cluster + Index + Shards)  
✅ **20–40 min** → CRUD operations & Mapping  
✅ **40–70 min** → Queries (Match, Term, Bool)  
✅ **70–90 min** → Aggregations + Analytics  
✅ **90–110 min** → Cluster management & Advanced features  
✅ **110–120 min** → Real-world use cases + Mock interview prep

Sonali, this is a **complete 0 → mastery kit** 🚀.  
Would you like me to also create a **mock interview Q&A (50+ important questions with answers)** so you can **revise in parallel** while following this 2-hour plan?

Top of Form



Bottom of Form

# 🔥 Elasticsearch Complete Guide (From Zero → Mastery)

## 1. ****Introduction****

* **What is Elasticsearch?**
  + A **distributed search and analytics engine**.
  + Built on **Lucene**.
  + Stores data in **JSON documents**.
  + Used for **full-text search, log analytics, metrics, monitoring**.
* **Why use it?**
  + Very fast (real-time search).
  + Horizontal scaling.
  + Works with **huge data**.
  + Supports **aggregations, filters, queries**.
* **Use Cases:**
  + E-commerce search (Amazon).
  + Logging (ELK stack → Elasticsearch + Logstash + Kibana).
  + Analytics dashboards.
  + Autocomplete & suggestions.

## 2. ****Core Concepts****

* **Cluster** → A collection of nodes (servers).
* **Node** → A single Elasticsearch server (Master / Data / Coordinating).
* **Index** → A database in Elasticsearch (collection of documents).
* **Document** → A JSON record stored in an index.
* **Shard** → Index split into parts for distribution.
* **Replica** → Copy of shard for fault tolerance.

📌 Example:

{

"id": 1,

"name": "Sonali",

"role": "Software Developer"

}

This JSON is a **document** in an index employees.

## 3. ****Installation & Setup****

👉 Already familiar since you’ve installed on Linux & Docker.  
Quick command:

docker run -d --name es-container -p 9200:9200 -e "discovery.type=single-node" elasticsearch:8.13.2

Check:

curl -X GET "localhost:9200/"

## 4. ****CRUD Operations****

### (a) Create Index

PUT /employees

### (b) Insert Document

POST /employees/\_doc/1

{

"name": "Sonali",

"role": "Developer",

"skills": ["Java", "Angular", "Elasticsearch"]

}

### (c) Get Document

GET /employees/\_doc/1

### (d) Update Document

POST /employees/\_update/1

{

"doc": {

"role": "Senior Developer"

}

}

### (e) Delete Document

DELETE /employees/\_doc/1

## 5. ****Searching Data****

### (a) Match Query

GET /employees/\_search

{

"query": {

"match": { "role": "Developer" }

}

}

### (b) Term Query (exact)

GET /employees/\_search

{

"query": {

"term": { "name.keyword": "Sonali" }

}

}

### (c) Boolean Query

GET /employees/\_search

{

"query": {

"bool": {

"must": [{ "match": { "role": "Developer" }}],

"filter": [{ "term": { "skills.keyword": "Java" }}]

}

}

}

## 6. ****Aggregations (Analytics)****

### Example: Count by Role

GET /employees/\_search

{

"size": 0,

"aggs": {

"roles\_count": {

"terms": { "field": "role.keyword" }

}

}

}

### Example: Average Salary

GET /employees/\_search

{

"size": 0,

"aggs": {

"avg\_salary": { "avg": { "field": "salary" } }

}

}

## 7. ****Index Management****

* **View all indexes**

GET /\_cat/indices?v

* **Delete index**

DELETE /employees

* **Reindex**

POST /\_reindex

{

"source": { "index": "employees" },

"dest": { "index": "employees\_new" }

}

## 8. ****Mappings & Data Types****

* **String** → text, keyword
* **Number** → integer, float
* **Date** → date
* **Boolean** → true/false
* **Object** → nested JSON

### Example Mapping

PUT /employees

{

"mappings": {

"properties": {

"name": { "type": "text" },

"role": { "type": "keyword" },

"salary": { "type": "float" },

"skills": { "type": "keyword" }

}

}

}

## 9. ****Advanced Features****

* **Update by Query**

POST /employees/\_update\_by\_query

{

"query": {

"match": { "role": "Developer" }

},

"script": {

"source": "ctx.\_source.role = 'Senior Developer'"

}

}

* **Bulk Insert**

POST /employees/\_bulk

{ "index": { "\_id": "1" } }

{ "name": "Sonali", "role": "Developer" }

{ "index": { "\_id": "2" } }

{ "name": "Rahul", "role": "Tester" }

## 10. ****Monitoring & Tools****

* **Kibana** → UI for Elasticsearch.
* **/\_cat/nodes** → List cluster nodes.
* **/\_cluster/health** → Check cluster health.

## 11. ****Elasticsearch in Projects****

* Use with **Spring Boot (Java)** via Elasticsearch RestHighLevelClient.
* Used for:
  + Logging & Monitoring.
  + Fast Searching APIs.