Question 1 of 15

What is Elasticsearch primarily used for in distributed systems?

[✓ 1 Efficient search and retrieval of documents and data]

2 Video streaming and media processing

3 Real-time messaging and chat applications

4 Cryptocurrency mining and blockchain operations

Correct!

Elasticsearch is a distributed search and analytics engine designed for search and retrieval operations. It excels at finding relevant documents from large datasets using full-text search, filtering, sorting, and aggregations.

Question 2 of 15

What is the fundamental relationship between Elasticsearch and Apache Lucene?

1 Lucene is built on top of Elasticsearch

2 Elasticsearch is a competitor to Lucene

3 They are completely independent systems

[✓ 4 Elasticsearch is a high-level orchestration framework built on top of Lucene]

Correct!

Elasticsearch serves as a distributed orchestration layer that handles cluster coordination, APIs, and real-time capabilities, while Apache Lucene provides the core search functionality and indexing engine underneath.

Question 3 of 15

In Elasticsearch's data model, what is the relationship between documents, indices, and mappings?

1 They are all equivalent concepts

2 Documents contain indices which contain mappings

3 Mappings contain indices which contain documents

[✓ 4 Indices contain documents, and mappings define the schema for those documents]

✓ **Correct!**

An index is a collection of documents (similar to a database table), while a mapping defines the schema - specifying field types and how they should be processed and indexed for search.

Question 4 of 15

Lucene segments are mutable containers that can be directly updated when documents are modified.

1 True

[✓ 2 False] ✓

Correct!

Lucene segments are immutable containers. Updates are handled by soft-deleting the old document and inserting a new one, with cleanup happening during segment merge operations. This immutability enables better performance and caching.

Question 5 of 15

What is the primary purpose of an inverted index in Elasticsearch?

1 To map terms/words to the documents that contain them for fast keyword searches ✓

2 To handle document updates

3 To store documents in reverse chronological order

4 To compress document storage

Correct!

An inverted index maps each unique word to the documents containing it, turning an O(n) scan through all documents into an O(1) lookup for keyword searches. This is the core data structure that makes text search fast.

Question 6 of 15

What are doc values in Elasticsearch and why are they important?

[✓ 1 Columnar storage of field values for efficient sorting and aggregations]

2 Backup copies of documents

3 Metadata about documents

4 Document version numbers

Correct!

Doc values provide columnar, contiguous storage of field values across all documents in a segment. This enables efficient sorting and aggregations by allowing access to specific field data without reading entire documents.

In Elasticsearch clusters, shards allow data and indexes to be distributed across multiple nodes for improved performance and scalability.

1 [✓ True]

2 False

Correct!

Shards enable Elasticsearch to split both documents and their corresponding index structures across multiple nodes. Searches execute across relevant shards in parallel, with results merged by coordinating nodes.

Question 8 of 15

How do replica shards improve Elasticsearch cluster performance?

1 They enable both high availability and increased search throughput by load balancing queries ✓

2 They only provide data backup for high availability

3 They reduce storage costs

4 They speed up write operations

Correct!

Replica shards serve dual purposes: they provide high availability if primary shards fail, and they increase search throughput by allowing the coordinating node to distribute search requests across all available shard copies (primary and replicas).

Question 9 of 15

Why might deep pagination using from/size become inefficient in Elasticsearch?

1 It requires too many disk writes

2 [✓ The cluster must retrieve and sort all preceding documents on each request]

3 It uses too much network bandwidth

4 It conflicts with the inverted index

✓ **Correct!**

Deep pagination with from/size requires the cluster to retrieve, sort, and skip over all documents up to the desired offset on every request. For large offsets (like page 1000), this becomes prohibitively expensive.

Question 10 of 15

What is the advantage of using search_after over from/size pagination?

1 It only fetches documents after the last result from the previous page, making deep pagination efficient ✓

2 It provides better security

3 It allows random access to any page

4 It uses less memory for small result sets

✓ **Correct!**

Search_after uses sort values from the last document of the previous page to progressively restrict the search set, avoiding the need to retrieve and sort all preceding documents. This makes deep pagination much more efficient.

Question 11 of 15

Point-in-time (PIT) cursors in Elasticsearch maintain a consistent view of data throughout pagination, even if the underlying index is being updated.

[✓ 1 True]

2 False

Correct!

PIT cursors create a consistent snapshot of the index state that remains stable throughout the pagination process, preventing issues like missing or duplicate results when documents are added/removed during pagination.

How does Elasticsearch's query planner optimize search performance?

1 By always using the inverted index first

2 By caching all search results

3 By distributing queries randomly across nodes

[✓ 4 By keeping statistics on fields and keywords to choose efficient execution strategies] ✓

Correct!

The query planner uses statistics about field types, keyword popularity, and document characteristics to determine the most efficient execution strategy, such as deciding the optimal order for processing query terms or whether to use indexes or direct document searches.

Question 13 of 15

Elasticsearch update operations have better performance than insert operations because they modify documents in place.

1 True

[✓ 2 False]

Correct!

Updates actually have worse performance than inserts because Elasticsearch soft-deletes the old document and inserts a new one with updated data. The old document remains until segment merges clean it up, requiring additional bookkeeping overhead.

Question 14 of 15

Which approach is generally recommended when using Elasticsearch in system design interviews?

1. [✓ Use Elasticsearch alongside an authoritative data store via Change Data Capture (CDC)] ✓
2. Use Elasticsearch as your primary database for all data
3. Replace all SQL databases with Elasticsearch
4. Use Elasticsearch only for write-heavy workloads

✓ **Correct!**

Elasticsearch is best used as a specialized search engine alongside an authoritative data store like Postgres or DynamoDB, with CDC keeping them synchronized. This provides search capabilities while maintaining data durability and consistency guarantees.

Question 15 of 15

What is a key limitation to consider when using Elasticsearch for frequently updated data?

1 Updates corrupt the inverted index

2 It only supports batch updates

3 It cannot handle any updates

[✓ 4 Updates require soft deletion and re-insertion, creating performance overhead until segment merges occur]

✓ **Correct!**

Elasticsearch handles updates by soft-deleting the old document and inserting new data, leaving deleted documents in segments until merge operations clean them up. This creates performance penalties for workloads with frequent updates, making it less suitable for rapidly changing data.