Top 100 Elasticsearch Interview Questions and Answers 2024

Table of contents

BASIC ELASTICSEARCH DEVELOPER INTERVIEW QUESTIONS

1.What is Elasticsearch? Explain in brief.

Elasticsearch is an open-source, distributed search and analytics engine based on the Apache Lucene library. It is designed to operate in near real-time, providing scalable, fast, and accurate text-based search capabilities along with numerous analytical functions.

Elasticsearch can store, search, and analyze large volumes of structured and unstructured data, making it a popular choice for various use cases such as full-text search, log, and event data analysis, and application performance monitoring.

2.Define ELK stack.

The ELK Stack is a collection of three open-source products or projects – Elasticsearch, Logstash, and Kibana – developed and maintained by Elastic. The acronym "ELK" is derived from the initials of these products.

These tools work together to deliver end-to-end log and event data management, analysis, and visualization.

3.What are the primary use cases of Elasticsearch?

Elasticsearch has numerous use cases across various domains due to its powerful search and analytical capabilities. Some of its use cases include:

- Full-text search: Elasticsearch makes complex search queries easier by quickly searching across large datasets. It is particularly helpful for websites, applications, or businesses that require instant and relevant search results.
- Log and event data analysis: Elasticsearch helps in quickly analyzing log data, system events, and application events, improving system monitoring and problem diagnosis.
- Anomaly detection: It can be used to detect unusual patterns, such as fraudulent activities, cyber-attacks, or performance issues, by analyzing stored data in real time.
- Data visualization: Elasticsearch can be combined with other tools like Kibana to create interactive data visualizations. This makes it easier to explore, understand, and use data effectively.
- Metrics and performance monitoring: Elasticsearch helps collect, analyze, and visualize performance metrics such as response time and system load, which aids in system optimization and capacity planning.
- Autocompletion and spell correction: Elasticsearch can provide real-time auto-completion and spell correction while users search, enhancing user experience and making searches more efficient.
- Geo-spatial search: Elasticsearch supports searching and filtering data based on geographical location, enabling distance-based search results and location-based analytics.

4.What is an Elasticsearch index?

Elasticsearch ensures data reliability through several features, including:

- Replication: Data is replicated to multiple nodes in the cluster, which protects against data loss in the event of a node failure.
- Sharding: Data is divided into shards, which can be distributed across multiple nodes. This improves performance and scalability.
- Snapshots and Restores: Elasticsearch provides a snapshot and restore feature that allows you to create and restore backups of your data. This protects against data loss due to human error or disasters.

- Monitoring and alerting: Elasticsearch provides many monitoring and alerting features that can help you to identify and address potential data problems.
- Security: Elasticsearch can be configured with several security features to protect your data from unauthorized access.

5.How does Elasticsearch ensure data reliability?

Elasticsearch ensures data reliability through several features, including:

- Replication: Data is replicated to multiple nodes in the cluster, which protects against data loss in the event of a node failure.
- Sharding: Data is divided into shards, which can be distributed across multiple nodes. This improves performance and scalability.
- Snapshots and Restores: Elasticsearch provides a snapshot and restore feature that allows you to create and restore backups of your data. This protects against data loss due to human error or disasters.
- Monitoring and alerting: Elasticsearch provides many monitoring and alerting features that can help you to identify and address potential data problems.
- Security: Elasticsearch can be configured with several security features to protect your data from unauthorized access

6.What is a node in Elasticsearch? What are the different types of nodes in Elasticsearch?

A node in Elasticsearch refers to a single running instance of the Elasticsearch process in a cluster.
Node is used to store data and engage in the indexing as well as search capabilities of the cluster.

Nodes communicate with one another to distribute data and workload, ensuring a balanced and high-performing cluster. Nodes can be configured with different roles, which determine their responsibilities in the cluster.

By using nodes, Elasticsearch can scale to handle large amounts of data and traffic. Nodes can be added to a cluster as needed, and they can be removed without affecting the availability of data. This makes Elasticsearch a highly scalable and reliable solution for storing and searching data.

You can assign roles to the nodes by setting up node.roles in Elasticsearch.yml. However, if you don’t set nodes.roles, by default, nodes will be assigned the following roles:

- Master-eligible nodes: These nodes are responsible for cluster-wide actions such as creating or deleting indices, managing nodes, and maintaining the overall cluster health. They participate in the election process for a master node, and one of them is elected as the master node.
- Data nodes: Data nodes store the actual data, called shards. They perform data-related operations such as indexing, searching, and aggregating. They also manage data replication to ensure high availability and resilience.
- Ingest nodes: These nodes pre-process incoming data before indexing. They use Elasticsearch's ingest pipelines to transform, enrich, and filter the data as it is ingested.
- Coordinating (or client) nodes: These nodes route search requests and handle query results. They do not store data or perform ingest processing but act as smart load balancers that optimize the distribution of queries and aggregations.
- Machine learning nodes: Machine learning nodes are dedicated to running machine learning jobs in Elasticsearch, commonly used for anomaly detection and data analysis.
- Cross-cluster search (CCS) nodes: CCS nodes enable querying multiple Elasticsearch clusters at once, acting as a single point to execute federated searches across these clusters.
- Voting-only nodes: These nodes are master-eligible but cannot be elected as master. Their primary function is to vote in master node elections, helping prevent tied votes and maintain cluster stability.

When setting nodes.roles, make sure to cross-check that nodes have been assigned roles as per your cluster’s needs. For instance, master and data roles are a must for every cluster.

7.What is a shard in Elasticsearch? What are the different types of shards in Elasticsearch?

A shard in Elasticsearch is a logical division of an index. An index can have one or more shards, and each shard can be stored on a different node in the cluster. Shards are used to distribute data across multiple nodes, which improves performance and scalability.

There are two types of shards in Elasticsearch:

- Primary shards
- Replica shards.

Primary shards are responsible for storing the original data, while replica shards are used to store backups of the data. By default, each index has one primary shard, but you can add additional primary shards to improve performance. You can also add replica shards to increase the availability of your data in the event of a node failure.

8.What is a replica in Elasticsearch?

A replica in Elasticsearch is a copy of a primary shard. Replicas are used to improve the availability of data in the event of a node failure. By default, each index has one primary shard and zero or one replica shard. The number of replica shards can be configured, and it is recommended to have at least one replica shard for each primary shard.

Replicas are located on different nodes in the cluster. This ensures that if one node fails, the data will still be available on the other nodes. Replicas are also updated in real-time, so they always have the most up-to-date data. Since replicas, just like primary shards, store a part of the index data, they can serve read requests, i.e., search and aggregation queries.

Having more replicas means that the search and read workload can be distributed among the primary and replica shards, which improves query performance and reduces the overall load on individual primary shards.

9.What is a document in Elasticsearch?

A document in Elasticsearch is a basic unit of information that can be indexed, stored, and searched. Documents are represented in JSON (JavaScript Object Notation) format, which is both human-readable and machine-parseable.

Each document consists of a collection of fields with their respective values, which can be of various data types like text, numbers, dates, geolocations, or booleans.

10.How do you create, delete, list, and query indices in Elasticsearch?

You can use the following commands:

- Command to create a new index – PUT /test_index?pretty
- Command to delete index -DELETE /test_index?pretty
- Command to list all index names and their basic information – GET _cat/indices?v
- Command to query an index – GET test_index/_search
- Command to query multiple indices – GET test_index1, test_index2/ _search

11.What is the Elasticsearch query language?

The Elasticsearch query language is referred to as the Query DSL (Domain Specific Language). It is a powerful and flexible language used for expressing queries in Elasticsearch. Query DSL is built on top of JSON and is used to construct complex queries, filters, and aggregations.

Query DSL features a wide range of queries and search capabilities, which can be categorized into:

- Full-text queries
- Term level queries
- Compound queries
- Join queries
- Geo queries
- Specialized queries

Query DSL also supports various other features like pagination, source filtering, highlighting, and sorting, enabling users to build even more sophisticated search and analysis capabilities.

12.What do you understand by index alias in Elasticsearch?

An index alias in Elasticsearch is a secondary name that can be used to refer to an index. Aliases can be used to make it easier to manage and use indexes. Aliases allow you to perform operations on multiple indices simultaneously or simplify index management by hiding the complexity of the underlying index structure.

Here are some of the benefits of using aliases in Elasticsearch:

- Simplicity: Aliases make it easier to manage and use indexes by providing a secondary name that can be used to refer to an index.
- Flexibility: Aliases can be used to group together a set of indexes, which can be useful if you want to perform the same operation on a group of indexes.
- Scalability: Aliases can be used to make it easier to scale your Elasticsearch cluster by providing a way to refer to a group of indexes.

13.Explain the concept of Elasticsearch mapping.

In Elasticsearch, a mapping is a JSON object that defines the structure of a document. It specifies the fields that are allowed in a document, as well as their data types and other properties.

Mappings are used to control how documents are stored and indexed, and they also affect how documents can be searched and analyzed. Mappings are a powerful tool that can be used to store data in a structured way. They make it easier to search, filter, and analyze your data.

14.What are analyzers in Elasticsearch?

In Elasticsearch, an analyzer is a component that is used to tokenize text. Analyzers are used to break down text into smaller units called tokens. These tokens are then used to index and search the text. The primary goal of analyzers is to transform the raw text into a structured format (tokens) that can be efficiently searched and analyzed.

An analyzer consists of three main components:

- Tokenizer: The tokenizer breaks the input text into a sequence of terms (tokens), usually by splitting it on whitespace or punctuation boundaries.
- Token filters: Token filters process the stream of tokens generated by the tokenizer and can modify, add, or remove tokens.
- Character filters: Character filters are used to preprocess the input text before it reaches the tokenizer. They can modify, add, or remove individual characters from the text.

15.What is Kibana?

Kibana is an open-source data visualization and exploration tool that works on top of Elasticsearch. Kibana allows you to:

- Visualize and explore data stored in Elasticsearch indexes by creating various types of visualizations such as pie charts, line charts, histograms, and heat maps.
- Analyze Elasticsearch data in real-time by creating interactive, shareable dashboards that can display multiple visualizations.
- Manage Elasticsearch, including configuring index patterns, adding new fields, and applying mapping changes.
- Discover and explore raw data in Elasticsearch, filtering and searching based on specific fields or queries.
- Monitor Elasticsearch performance by tracking various metrics and logs.

16.How does Elasticsearch scale horizontally?

Elasticsearch scales horizontally by distributing data across multiple nodes. Each node in an Elasticsearch cluster can store and process data. By adding more nodes to the cluster, you can increase the amount of data that can be stored and processed. Elasticsearch uses the concept of sharding and replication to scale horizontally.

- Sharding: Sharding is the process of splitting data into smaller units called shards, each containing a portion of the data. Each shard is a fully functional and independent index that can be hosted on any node within the Elasticsearch cluster.
- Replication: Replication creates replica shards, which are redundant or backup copies of the primary shards, for increased fault tolerance and improved read performance.

17.Explain the role of creating, reading, updating, and deleting documents in Elasticsearch.

Elasticsearch is a document-based NoSQL database designed for fast and flexible full-text search, analytics, and data manipulation. The primary operations you can perform on the documents stored in Elasticsearch are creating, reading, updating, and deleting, collectively known as CRUD operations.

- Creating documents: Creating a document in Elasticsearch involves adding a new document with a unique ID to a specific index.
- Reading documents: Reading refers to retrieving a document or a group of documents from Elasticsearch based on specific criteria. You can fetch a document using its unique ID or perform searches using various queries that may involve matching, filtering, or aggregating the data.
- Updating documents: Updating a document involves modifying the content of an existing document or adding new fields.
- Deleting documents: Deleting documents removes data from Elasticsearch. You can delete documents by specifying their unique IDs or by using queries to delete multiple documents that match certain criteria.

18.What is an Elasticsearch cluster?

An Elasticsearch cluster is a group of one or more interconnected nodes (servers) working together to handle search, indexing, and data management tasks. The cluster enables horizontal scaling, distributes data and operations across multiple nodes, and achieves high availability and fault tolerance by replicating data across these nodes.

Key concepts associated with Elasticsearch are:

- Nodes
- Index
- Shards
- Replicas
- Cluster state

19.What is the significance of the _source field in Elasticsearch?

The _source field in Elasticsearch is an important system field that stores the original JSON object that was passed when a document was indexed. It is an essential part of Elasticsearch as it enables a variety of functionalities and provides several benefits:

- Document Retrieval: When you fetch a document or perform a search in Elasticsearch, the _source field allows you to return complete or partial original JSON objects to the user.

- Partial Updates: Elasticsearch supports partial updates, which allow you to modify specific fields within a document without reindexing the entire document. By specifying the _source field in an update request and providing the updated fields, you can update only the necessary parts of the document.
- Source Filtering: Elasticsearch provides source filtering, which allows you to control the fields returned in search results. With source filtering, you can specify a whitelist or a blacklist of fields to include or exclude from the _source field.

20.Describe the inverted index in Elasticsearch.

The inverted index is a core data structure used by Elasticsearch for efficient full-text search and retrieval. It is the backbone of Elasticsearch's search capabilities, enabling fast and accurate keyword-based search queries.

An inverted index works by breaking down text documents into smaller units called tokens. These tokens are then stored in a database, along with a list of the documents that contain the token. When a search query is performed, the inverted index is used to quickly find the documents that contain the search terms.

An inverted index is a powerful tool that can be used to search large amounts of text data and is used by various popular search engines such as Google, Bing, and Yahoo.

21.Explain the concept of eventual consistency in Elasticsearch.

Eventual consistency is a model used by some distributed systems like Elasticsearch, wherein the system guarantees that all nodes will eventually have a consistent view of the data, but not necessarily immediately after a write operation. In other words, it allows for temporary inconsistencies between nodes in order to prioritize better performance and availability.

Eventual consistency is a good choice for many applications because it provides a good balance between consistency and availability. With eventual consistency, you can be sure that your data will eventually be consistent, but you may not get the latest data immediately.

22.What are the key differences between RDBMS and Elasticsearch?

RDBMS (Relational Database Management System) and Elasticsearch are both data management systems but have different architectural principles.

- Data Model: RDBMS uses a relational data model, where data is organized into tables with rows and columns. Elasticsearch uses a document-based data model, where data is stored as JSON documents within indices.
- Query language: RDBMS uses SQL (Structured Query Language) to interact with the database, including actions like creating, retrieving, updating, or deleting data. Elasticsearch uses a query DSL (Domain Specific Language), which is JSON-based, to perform search and analytics operations on the indexed data.
- Scaling: RDBMS generally scales vertically, requiring more powerful hardware to handle larger datasets or increased operations. Elasticsearch is designed for horizontal scaling, distributing the data and operations across multiple nodes within a cluster, enabling better performance and capacity management.
- Performance: RDBMS is optimized for transactional processing, including various consistency guarantees and support for ACID properties. Elasticsearch is built for high-performance search and analytics, prioritizing fast response times and real-time indexing capabilities.

23.Describe the parent-child relationship in Elasticsearch.

A parent-child relationship in Elasticsearch is a way to model a hierarchical relationship between documents. In a parent-child relationship, one document (the parent) can have one or more child documents. The parent document is the root of the hierarchy, and the child documents are its descendants.

To create a parent-child relationship, specify the parent field in the child document. The parent field is a string that contains the ID of the parent document. When you index a child document, Elasticsearch will automatically associate it with the parent document.

24.What are aggregations in Elasticsearch? What are the different types of aggregations in Elasticsearch?

Aggregations in Elasticsearch are a powerful feature that allows you to analyze, summarize, and perform complex calculations on your dataset in real-time. Aggregations provide the capability to group and extract actionable insights from indexed data, which can be used for data visualization, reporting, and analytical purposes.

There are three main types of aggregations in Elasticsearch:

- Bucket aggregations: Bucket aggregations group documents into buckets based on a common value. For example, you can use a bucket aggregation to group documents by the value of a field, such as the product name or the user ID.
- Metric aggregations: Metric aggregations calculate metrics, such as the count, sum, average, or minimum value of a field. For example, you can use a metric aggregation to calculate the total number of documents, the sum of all prices, or the average age of all users.
- Pipeline aggregations: Pipeline aggregations work on the outputs of other aggregations rather than directly on document data. They can be used to chain multiple aggregations or to perform additional calculations, such as cumulative sums or moving averages.

25.What are field data types in Elasticsearch mapping?

In Elasticsearch, field data types define the nature of the values stored in a field and determine how the data is indexed, stored, and searched. When defining an index mapping, you can specify the field data type for each field to ensure appropriate handling and interpretation of the data.

Elasticsearch supports various field data types, each suited for different kinds of data such as text, keyword, numeric, date, boolean, binary, array, and object.

26.What are Elasticsearch refresh and flush operations?

In Elasticsearch, refresh and flush are index management operations that handle the process of making indexed documents available for search and maintaining the durability and integrity of the data.

Refresh is an operation that makes changes to the index available for search. When you add or update a document in Elasticsearch, the change is not immediately available for search. Instead, it is first added to a buffer in memory. The refresh operation copies the changes from the buffer to the index, making them available for search.

A flush operation ensures that data that's stored in Elasticsearch's in-memory buffers (also known as the in-memory transaction log) is written to disk, providing durability and data integrity. It clears the in-memory buffers and frees up memory resources. In addition, the flush operation also commits the transaction log and starts a new one.

27.What is Elasticsearch cat()?

In Elasticsearch, the cat() API is a set of simple and concise APIs that provide information about the cluster, nodes, indices, and other components in a human-readable format. The cat() API is primarily used for troubleshooting, monitoring, and obtaining quick insights into the state and health of an Elasticsearch cluster.

Some of the common cat APIs are cat.indices, cat.nodes, cat.health, and cat.allocation among others.

28.Explain the function of cat.indices in Elasticsearch.

In Elasticsearch, the cat.indices API provides a way to retrieve information about the indices in the cluster in a human-readable format. It allows you to obtain various details and statistics about the indices, such as their names, sizes, health status, number of documents, and more.

The cat.indices API is primarily used for monitoring and troubleshooting purposes, as it provides a quick and concise overview of the indices within the Elasticsearch cluster. It is often utilized by administrators and developers to gather essential information about the state and performance of the indices.

29.What is the use of cat.nodes in Elastic search?

In Elasticsearch, the cat.nodes API is used to retrieve information about the nodes in an Elasticsearch cluster. It provides a concise and human-readable overview of the individual nodes, their roles, statuses, resource usage, and other relevant metrics.

It helps administrators, developers, and operators monitor the health, resource utilization, and roles of individual nodes, allowing for better cluster management, troubleshooting, and performance optimization.

30.What is the cat.health API in Elasticsearch?

In Elasticsearch, the cat.health API is used to retrieve information about the overall health status of the cluster. It provides a concise and human-readable overview of the health of the cluster and its indices. The cat.health API is useful for monitoring the overall health of the Elasticsearch cluster and gaining visibility into any potential issues related to shard allocation, replica shards, or unassigned shards.

It allows administrators and operators to quickly assess the state of the cluster and take appropriate actions to ensure its stability and performance.

31.Discuss Elasticsearch filter context and query context.

In Elasticsearch, queries can be executed in two different contexts: the filter context and the query context.

- Filter context: Filter context is used when you want to filter out documents that do not match certain criteria. For example, you could use filter context to find all documents that were created after a certain date. In filter context, no scoresFilter context is generally more performance-efficient since it bypasses the scoring process calculated, so the order of the documents in the results is not relevant.
- Query Context: Query context is used when you want to rank documents based on their relevance to a query, such as a full-text search or a match on multiple fields. For example, you could use query context to find all documents that contain the word

"Elasticsearch". In query context, scores are calculated, so the documents in the results are ordered by relevance.

32.Explain the differences between a query and a filter.

In Elasticsearch, queries and filters are used to retrieve specific documents from an index, but they have some key differences in terms of their functionality and usage.

- Query: The primary purpose of a query is to determine the relevance of documents to a search query. It calculates a relevance score for each document based on how well it matches the query criteria. Queries are used when you want to retrieve the most relevant documents based on their relevance scores.
- Filter: The main purpose of a filter is to narrow down the search results by applying specific conditions or filters. Filters are used when you want to include or exclude documents based on certain criteria without considering their relevance scores.

INTERMEDIATE ELASTICSEARCH DEVELOPER INTERVIEW

QUESTIONS

1.What are tokenizers in Elasticsearch?

In Elasticsearch, tokenizers are components responsible for breaking down the text into individual tokens during the indexing process. Tokenization is a crucial step in the analysis process, where text is divided into meaningful units or tokens that can be indexed and searched efficiently.

There are several tokenizers in Elasticsearch and some of these include the following:

- Standard tokenizer: The standard tokenizer is a good all-purpose tokenizer that works well for most languages. It breaks down text into individual words, taking into account grammar and punctuation.
- Whitespace tokenizer: The whitespace tokenizer breaks down text into individual words based on whitespace characters, such as spaces, tabs, and newlines.

- Keyword tokenizer: The keyword tokenizer does not perform any stemming or stopword removal. It simply breaks down text into individual words, without any further processing.

2.What are some important Elasticsearch APIs?

Elasticsearch provides a rich set of APIs (Application Programming Interfaces) that allows you to interact with and manage the Elasticsearch cluster. Here are some important Elasticsearch APIs:

- Index API: The index API is used to create, update, and delete indexes. An index is a collection of documents that share the same schema. The schema defines the structure of the documents in the index.
- Document API: The document API is used to create, update, and delete documents. A document is a piece of data that is stored in an index. Documents can be of any type, but they are typically JSON objects.
- Search API: The search API is used to search for documents. The search API allows you to specify a query, which is a set of criteria that the documents must match in order to be returned.
- Aggregation API: The aggregation API is used to aggregate search results. Aggregations allow you to group search results together and calculate summary statistics.
- Cat API: The cat API is used to get information about indexes, documents, and shards. The cat API provides a variety of information, such as the number of documents in an index, the size of an index, and the number of shards in an index.

3.What are the disadvantages of using Elasticsearch?

Elasticsearch is a powerful search engine, but it also has some disadvantages. Some of the most common disadvantages of Elasticsearch include:

- Complexity: Elasticsearch is a complex system with a steep learning curve. It can be difficult to understand how Elasticsearch works and how to use it effectively. This complexity can make it difficult to troubleshoot problems and find help when you need it.
- Resource Intensive: Elasticsearch is resource-intensive, requiring a sufficient amount of memory, CPU, and storage to run efficiently. Large-scale deployments with extensive indexing and search

operations may require substantial hardware resources and infrastructure investments.
- Query Complexity: Complex search queries, especially involving aggregations or advanced analytics, can be challenging to design and optimize in Elasticsearch. Fine-tuning and optimizing search queries for performance and relevance may require expertise and experimentation.
- Scalability: Scaling Elasticsearch clusters horizontally to handle increasing data volumes and search traffic can be challenging.

4.What are the differences between Elasticsearch and Solr?

Elasticsearch and Solr are both open-source, distributed, and scalable search engines. They are both built on top of Apache Lucene, but they have some key differences:

- Realtime search and indexing: Elasticsearch focuses heavily on near real-time indexing and search capabilities, allowing very rapid indexing and retrieval of documents. Solr supports real-time indexing and search but may not be as fast or streamlined as Elasticsearch in certain situations.
- Schema: Elasticsearch is schema-less by default, supporting dynamic mapping that automatically infers data types and fields. Solr requires a predefined schema, although recent versions have introduced support for dynamic fields and schema-less operation.
- Aggregation: Elasticsearch provides a comprehensive aggregation framework that enables complex data analysis, summarization, and visualization of indexed data. Solr has Faceting and Stats components, but its capabilities are more limited compared to Elasticsearch's aggregations.

5.Explain how Elasticsearch handles pagination.

Elasticsearch handles pagination using the ‘from’ and ‘size’ query parameters in the search request. These parameters allow you to retrieve a specific range of results from your search query, useful for dividing the search results into smaller subsets or "pages" for consumption.

- ‘from’ parameter: The from parameter specifies the starting offset to retrieve the results.

- ‘size’ parameter: The size parameter sets the maximum number of results to return in a search request.

However, it is important to note that Elasticsearch's pagination has limitations, especially when dealing with deep pagination or large result sets.

6.How does Elasticsearch handle schema-less document indexing?

Elasticsearch is a schema-less document-based search engine. This means that you don't need to define a schema for your documents before you index them. You can simply index your documents as JSON objects, and Elasticsearch will automatically create a schema for you.

When you index a document, Elasticsearch will create a mapping for the document. The mapping defines the fields in the document and their types. Elasticsearch will use the mapping to index the document and to search the document.

7.What is the Elasticsearch percolator?

The Elasticsearch percolator is a feature that allows you to store queries and then use those queries to search for documents. This can be useful for a variety of applications, such as:

- Alerting: You can store queries that represent alerts, and then use those queries to notify users when documents match the alert criteria.
- Personalization: You can store queries that represent user preferences, and then use those queries to search for documents that are likely to be of interest to the user.
- Filtering: You can store queries that represent filters and then use those queries to search for documents that match the filter criteria.

8.What is the purpose of the match query function in Elasticsearch?

The match query function in Elasticsearch is used to search for documents that contain a specific value. The match query can be used to search for documents that contain a specific string, a specific number, or a specific date.

The match query takes two arguments: the field name and the value to search for. The field name is the 'name of the field' in the document you want to search. The value is the value that you want to search for.

9.How does the range query function work in Elasticsearch?

The range query function in Elasticsearch is used to search for documents that contain a value within a specific range. The range query can be used to search for documents that contain a specific string, a specific number, or a specific date.

The range query takes three arguments: the field name, the start value, and the end value. The field name is the name of the field in the document that you want to search. The start value is the minimum value that you want to search for. The end value is the maximum value that you want to search for.

10.What are the use cases for the multi-match query function?

The multi-match query function in Elasticsearch is designed to perform searches across multiple fields with a single query. It allows you to specify multiple fields and search for matching documents across all the specified fields simultaneously. Here are some common use cases for the multi-match query function:

- Cross-field search
- Querying multiple text fields
- Search relevance
- Flexible search
- Multilingual search
- Partial matching

11.Explain the purpose of the 'exists' query function.

The 'exists' query function in Elasticsearch is used to check whether a specific field exists in a document or not. It is particularly useful when you want to search for documents based on the presence or absence of a certain field.

The ‘exists’ query takes one argument: the field name. The field name is the name of the field that you want to search for.

12.How does the ‘exists’ query function works?

Here's how the ‘exists’ query function works:

- Field Selection: Specify the field you want to check for existence in the documents.
- Document Matching: The ‘exists’ query returns documents that have the specified field. It doesn't consider the field value; it simply checks if the field exists in the document.
- Absence Filtering: If you want to find documents that do not have the specified field, you can use the must_not clause along with the exists query. This will filter out documents where the field exists, returning only documents where the field is absent.

13.Discuss the use of the Elasticsearch script_score function.

The script_score function in Elasticsearch is used to customize the scoring of search results based on a custom scoring script. It allows you to influence the relevance scoring of documents by providing a script that calculates a custom score for each document.

The script_score function offers flexibility and control over the scoring of search results in Elasticsearch.

14.Explain the purpose of the bool query and its main clauses in Elasticsearch.

A boolean query in Elasticsearch is a query that uses boolean operators to combine multiple queries into a single query. The main clauses of a boolean query are:

- must - The must clause specifies the documents that must match the query. All documents that do not match the must clause will be excluded from the results.
- should - The should clause specifies the documents that should match the query. Documents that match the should clause will be included in the results, even if they do not match the must clause.
- must_not - The must_not clause specifies the documents that must not match the query. All documents that match the must_not clause will be excluded from the results.

15.How does Elasticsearch implement the common_terms query function for text queries?

The common_terms query function in Elasticsearch is used for text queries and aims to find documents that match a query while accounting for common terms that might normally be ignored by other queries. It helps address the challenge of finding relevant results while still considering common terms that occur in a significant portion of the indexed documents.

16.What are the differences between sort() and rank_feature() functions in Elasticsearch?

- sort(): The sort() function is used to order the search results based on specific criteria. It allows you to sort the documents based on one or more fields, either in ascending or descending order. The sort() function is commonly used to order documents based on relevance, date, numeric values, or custom criteria.
- rank_feature(): The rank_feature() function, on the other hand, is used to influence the relevance scoring of documents. It allows you to assign importance or weights to specific features or attributes of the documents, which then affect the relevance score.

17.What are the primary responsibilities of master-eligible nodes in Elasticsearch?

Master-eligible nodes in Elasticsearch are responsible for managing the cluster. This includes tasks such as:

- Managing cluster state: The master node is responsible for managing the cluster state. This includes information such as which nodes are in the cluster, which indices exist, and where shards are located.

- Assigning shards: When a new index is created, the master node assigns shards to the nodes in the cluster.
- Rebalancing shards: If a node fails, the master node rebalances shards to ensure that all of the data is still available.

18.What is the function of hot-warm-cold architecture in Elasticsearch?

Hot-warm-cold architecture is a way to distribute data across different nodes in an Elasticsearch cluster. The idea is to have three tiers of nodes:

- Hot nodes: Hot nodes are the fastest and most powerful nodes in the cluster. They are used to store and serve the most frequently accessed data.
- Warm nodes: Warm nodes are not as fast as hot nodes but are still relatively fast. They are used to store and serve data that is accessed less frequently than data on hot nodes.
- Cold nodes: Cold nodes are the slowest and least powerful nodes in the cluster. They are used to store and serve data that is accessed very infrequently.

19.Explain how indexing and searching operations are performed by data nodes.

In Elasticsearch, data nodes are responsible for storing, indexing, and serving the actual data. They handle both indexing operations, which involve adding or updating documents in the index, as well as searching operations, which involve retrieving relevant documents based on search queries.

Here are the indexing and searching operations that are performed:

Indexing operations-

- Document ingestion
- Tokenizing
- Building inverted index
- Shard management

Searching operations-

- Query execution
- Query parsing and analysis
- Retrieval
- Scoring and ranking
- Result aggregation and pagination

20.What is the purpose of setting the number of replica shards in Elasticsearch?

The purpose of setting the number of replica shards in Elasticsearch is to increase the availability of your data. By setting a number of replica shards, you are creating additional copies of your data on different nodes in the cluster. This means that if one node fails, the data will still be available on other nodes.

The number of replica shards that you set will depend on the size of your data and the level of availability that you need.

21.Discuss the process of Elasticsearch node discovery. What are the benefits of using node discovery in Elasticsearch?

Elasticsearch node discovery is a process where the module for cluster formation finds other relevant nodes with which a cluster can be formed. This process starts when an Elasticsearch node is started or on the assumption that the master node has failed. This process will continue until a new master node is appointed or the master node is found.

Here are some of the benefits of using node discovery in Elasticsearch:

- Increased availability: Node discovery allows Elasticsearch to distribute data across multiple nodes. In the event of a node failure, the data remains accessible through other nodes, ensuring uninterrupted availability.
- Improved performance: Node discovery can improve the performance of your search queries. This is because Elasticsearch can distribute a load of search queries across multiple nodes.
- Reduced risk of data loss: Node discovery can reduce the risk of data loss. The system is designed to ensure data availability even if one node

experiences a failure, as the information is replicated across other nodes.

Overall, node discovery is an important part of Elasticsearch and it helps to improve the availability, performance, and resilience of your data.

22.Describe the role of coordinating nodes in Elasticsearch.

In Elasticsearch, coordinating nodes play a vital role in distributing and coordinating search and indexing operations across the cluster. Coordinating nodes act as the entry point for client requests and are responsible for orchestrating the communication between the client and the data nodes. The coordinating nodes are an important part of Elasticsearch and they help to improve the scalability, performance, and availability of the system.

23.What are ingest nodes in Elasticsearch? How and when should ingest nodes be used in Elasticsearch?

Ingest nodes in Elasticsearch are used for processing documents before they are indexed. Ingest nodes can be used to improve the quality and usability of your data. Ingest nodes are also a good way to improve the performance of your Elasticsearch cluster.

By performing some of the processing of documents on ingest nodes, you can free up the data nodes to focus on indexing and searching.

Ingest nodes should be used when you need to:

Improve the quality of your data: Ingest nodes can be used to remove unwanted fields or documents, transform the format of fields or documents, or enrich fields or documents with additional information. This can help to improve the accuracy and usability of your data.

Improve the performance of your Elasticsearch cluster: By performing some of the processing of documents on ingest nodes, you can free up the data nodes to focus on indexing and searching. This can improve the performance of your cluster, especially if you are dealing with a large volume of data.

Reduce the costs of running your Elasticsearch cluster: Ingest nodes can help to reduce the costs of running an Elasticsearch cluster by offloading some of the processing of documents to less powerful nodes. This can be a cost-effective way to improve the performance and scalability of your cluster.

24.Explain how the automatic removal of old indices works in the Elasticsearch rollover node.

In Elasticsearch, the rollover process is used to automatically manage and remove old indices to ensure efficient data organization and resource utilization. A rollover node is a designated node responsible for performing the rollover operation. These are some of the aspects associated with the removal of old indices:

- Index Lifecycle Management (ILM): The automatic removal of old indices is typically achieved through the use of Index Lifecycle Management (ILM) in Elasticsearch.
- Rollover Conditions: When configuring an ILM policy, you define conditions that trigger the rollover process.
- Rollover Operation: When the rollover conditions are met, the rollover process is triggered.
- Retention and Deletion: As part of the ILM policy, you can specify the retention period for the indices.
- Deletion Methods: Elasticsearch provides different mechanisms for deleting old indices, depending on your requirements.

25.What is shard allocation filtering? How does shard allocation filtering play a role in Elasticsearch attributes?

Shard allocation filtering is a feature in Elasticsearch that allows you to control the placement of shards within a cluster based on certain criteria. It enables you to define rules and conditions to filter out specific nodes from participating in shard allocation. This feature gives you fine-grained control over how shards are distributed across your cluster.

Shard allocation filtering can be used to filter shards on a variety of attributes, including:

- Node attributes: You can filter shards on node attributes such as the node's name, IP address, or role.

- Index attributes: You can filter shards on index attributes such as the index's name, age, or size.
- Shard attributes: You can filter shards on shard attributes such as the shard's primary/replica status or its routing id.

Shard allocation filtering can be a powerful tool for managing the performance, availability, and security of your Elasticsearch cluster.

26.How does Elasticsearch use "gateway.recover_after_nodes" attribute during cluster recovery?

The gateway.recover_after_nodes attribute in Elasticsearch is used to control the recovery process of a cluster after a failure or restart. It specifies the minimum number of nodes that need to be present and recoverable in the cluster before the recovery process begins.

- Cluster Startup: When an Elasticsearch cluster starts up, it goes through a recovery process to restore the cluster state and data from the available shards
- Minimum Nodes Requirement: The gateway.recover_after_nodes attribute comes into play during cluster recovery by specifying the minimum number of nodes required for the recovery to begin.
- Node Availability: Elasticsearch checks the availability and recoverability of nodes in the cluster based on their status and health.
- Recovery Process: Once the minimum required number of nodes (gateway.recover_after_nodes) becomes available, Elasticsearch initiates the recovery process.

27.Describe the function of the "node.attr.box_type" attribute in Elasticsearch architecture.

The node.attr.box_type attribute in Elasticsearch architecture is used to designate a node's role in the cluster. This attribute can be used to create a hot/warm/cold architecture, where different nodes are responsible for different types of data.

By using the node.attr.box_type attribute, you can improve the performance and scalability of your Elasticsearch cluster. Hot nodes can be optimized for performance, while warm and cold nodes can be optimized for cost-effectiveness. To use the node.attr.box_type attribute, you need to set the attribute in the node's configuration file.

28.How can you use Elasticsearch custom attributes to control node behavior?

Elasticsearch custom attributes allow you to control node behavior in a variety of ways. You can use custom attributes to:

- Designate a node's role in the cluster: You can use custom attributes to designate a node's role in the cluster, such as hot, warm, or cold.
- Control where shards are allocated: You can use custom attributes to control where shards are allocated in your cluster.
- Configure node-specific settings: You can use custom attributes to configure node-specific settings, such as the amount of memory or the number of threads. This can be used to optimize your nodes for specific workloads.
- Implement custom logic: You can use custom attributes to implement custom logic, such as routing documents to specific nodes based on their content.

29.Describe the role of the“cluster.routing.allocation.node_concurrent_recoveries” attribute.

The “cluster.routing.allocation.node_concurrent_recoveries” attribute in Elasticsearch controls the number of concurrent shard recoveries that are allowed on a node. This attribute can be used to improve the performance and availability of your cluster.

If you increase the “cluster.routing.allocation.node_concurrent_recoveries” attribute, you can allow more concurrent shard recoveries on a node. This can improve the performance of your cluster by allowing shards to recover in parallel. It can also improve the availability of your cluster by allowing shards to recover even if some nodes are unavailable.

However, increasing the “cluster.routing.allocation.node_concurrent_recoveries” attribute can also increase the load on your nodes. If you increase this attribute too much, you may begin to see performance degradation or even node failures.

30.How can you update index-level settings using Elasticsearch attributes?

In Elasticsearch, you can update index-level settings using attributes by leveraging the Index Templates feature. Index Templates allow you to define default settings and mappings for new indices that match certain patterns. By using attributes in your index templates, you can dynamically configure index-level settings based on the attributes assigned to the indices.

31.What is node responsiveness and how can you monitor it using Elasticsearch attributes?

Node responsiveness is a measure of how quickly a node can respond to requests. It is important to monitor node responsiveness because it can impact the performance of your Elasticsearch cluster.

There are a few ways to monitor node responsiveness using Elasticsearch attributes. One way is to use the node.status attribute. This attribute provides information about the health of a node, including its responsiveness.

Another way to monitor node responsiveness is to use the node.info attribute. This attribute provides more detailed information about a node, including its CPU usage, memory usage, and disk usage. Additionally, you can also use node.stats API to get more detailed information about node responsiveness.

32.What is the purpose of the "indices.recovery.max_bytes_per_sec" attribute?

The indices.recovery.max_bytes_per_sec attribute in Elasticsearch is a cluster-level setting that allows you to control the rate at which data is recovered during the shard recovery process. It specifies the maximum number of bytes per second that can be transferred for recovery operations.

The purpose of this attribute is to regulate the bandwidth allocated for recovery, preventing excessive resource consumption and potential performance degradation.

33.How does Elasticsearch use the thread_pool.bulk.queue_size" attribute?

In Elasticsearch, the thread_pool.bulk.queue_size attribute is used to control the size of the queue that holds bulk requests when they cannot be immediately processed by the bulk thread pool.

The bulk thread pool is responsible for executing bulk indexing operations, which involve processing multiple index or update requests together in a batch. The queue_size attribute helps manage the backlog of bulk requests and ensures efficient utilization of system resources.

34.Describe the purpose of the "transport.tcp.compress" attribute in Elasticsearch.

The transport.tcp.compress attribute in Elasticsearch is used to enable or disable compression of network communication between nodes in a cluster. When enabled, it compresses the data being transmitted over the network, reducing the size of network packets and potentially improving network performance and efficiency.

The purpose of the transport.tcp.compress attribute is to optimize network utilization and reduce bandwidth consumption in an Elasticsearch cluster.

35.Explain how Elasticsearch scales its performance.

Elasticsearch scales its performance by using a number of techniques, including:

- Sharding: Sharding is the process of dividing data into smaller pieces, called shards. Shards are then distributed across the nodes in the cluster. This allows Elasticsearch to scale horizontally, by adding more nodes to the cluster.

- Replication: Replication is the process of copying shards to multiple nodes in the cluster. This ensures that the 14-data is always available, even if a node fails.
- Caching: Caching is the process of storing frequently accessed data in memory. This can improve performance by reducing the number of times Elasticsearch has to access the disk.
- Indexing optimizations: Elasticsearch supports a number of indexing optimizations that can improve performance, such as prefix and suffix matching, and field collapsing.

36.How does Elasticsearch handle configuration anagement? What are the different configuration management tools supported by Elasticsearch?

Elasticsearch handles configuration management through a combination of static and dynamic settings. Static settings are defined in the elasticsearch.yml file, which is located in the Elasticsearch installation directory. Dynamic settings can be changed on a running cluster using the cluster.update_settings API.

Elasticsearch provides support for various configuration management tools to help with the deployment and management of Elasticsearch clusters such as:

- Ansible: Ansible is an open-source automation platform that allows you to define and manage infrastructure as code. It provides modules and playbooks for installing, configuring, and managing Elasticsearch clusters.
- Puppet: Puppet is a widely used configuration management tool that provides a declarative language for defining infrastructure configurations. Puppet has modules available for managing Elasticsearch, allowing you to define the desired state of Elasticsearch nodes and automate their configuration, deployment, and maintenance.
- Chef: Chef is another popular configuration management tool that provides a framework for automating infrastructure configurations. Chef provides resources and recipes to manage Elasticsearch nodes and automate tasks like cluster setup, node provisioning, and configuration changes.

ADVANCED ELASTICSEARCH DEVELOPER INTERVIEW QUESTIONS

1.What is the role of routing in Elasticsearch?

Routing plays a crucial role in Elasticsearch for efficient data distribution, search execution, and indexing operations. It is a mechanism that determines how documents are distributed across shards within an index and how search and indexing requests are routed to the appropriate shards.

Elasticsearch uses a number of factors to determine the routing of a document, including:

- The index name
- The document type
- The document ID
- The routing value

2.Explain how Elasticsearch handles node failure.

Elasticsearch handles node failure by using a process called rebalancing. When a node fails, Elasticsearch will rebalance the cluster by moving shards from the failed node to the remaining nodes in the cluster. This ensures that all data is always available, even if a node fails.

Besides that, Elasticsearch deploys replication, shard recovery, automatic node discovery, and monitoring mechanisms to handle node failures and ensure the availability and reliability of data in the cluster.

3.What is the significance of the minimum master node configuration?

The minimum master node configuration in Elasticsearch specifies the minimum number of nodes that must be available in the cluster in order for the cluster to be considered healthy and operational.

If the number of nodes in the cluster falls below the minimum master nodes configuration, the cluster will enter a "split-brain" state, which can lead to data loss and other problems.

The minimum master node configuration is important because it ensures that there is always a quorum of nodes available to manage the cluster. A quorum is a majority of nodes in the cluster, and it is required for tasks such as electing a new master node, creating or deleting indices, and rebalancing shards.

4.Describe Elasticsearch's _all field.

In Elasticsearch, the _all field is a special field that allows you to search across all other fields in a document. It is designed to simplify and improve search functionality by providing a unified field that combines the content of all other fields in a document.

This can be particularly useful when you want to perform a quick and simple search without specifying specific fields to search in.

5.Explain the Phrase Match option in Elasticsearch.

In Elasticsearch, the "Phrase Match" option is a search feature that allows you to find documents that contain an exact sequence of terms in a specific order. It is particularly useful when you want to search for a specific phrase or a set of terms that occur together in the same order within a document. The Phrase Match option allows you to specify a set of terms that should appear together in a document in the same order as the query.

6.What is Optimization in Elasticsearch?

Optimization in Elasticsearch refers to the process of improving the performance and efficiency of an Elasticsearch cluster and its indices. Here are some key aspects of optimization in Elasticsearch:

- Indexing Optimization: Elasticsearch provides several strategies to optimize the indexing process. These include bulk indexing, which allows efficient indexing of multiple documents in a single request, and the use of the refresh_interval setting to control the frequency of index refreshing.

- Shard and Replica Configuration: Sharding is the process of splitting an index into smaller parts called shards, which can be distributed across the cluster. By properly configuring the number of shards and replicas, you can balance the indexing and search load, distribute data evenly, and improve parallelism for better performance.
- Query Optimization: Optimizing queries involves understanding and utilizing query types that are most appropriate for specific search scenarios, using filters for non-scoring conditions, and leveraging features like caching, filter caches, and query profiling to improve query performance.

7.How does Elasticsearch ensure data replication between nodes?

Elasticsearch ensures data replication between nodes by using a process called "sharding." Sharding is the process of dividing a large index into smaller pieces, called shards. Each shard is then replicated to multiple nodes in the cluster.

This ensures that if one node fails, the data is still available on other nodes. Data replication is a valuable feature that can help you to improve the availability, reliability, performance, and scalability of your Elasticsearch cluster.

8.What is fuzzy searching in Elasticsearch? Explain how Elasticsearch handles fuzzy searching.

In Elasticsearch, fuzzy searching is a feature that allows you to find documents that match a search term even if they have slight variations or spelling errors. It is particularly useful when you want to retrieve relevant results that closely resemble the search term, accounting for potential typos, misspellings, or variations in word endings.

Elasticsearch handles fuzzy searching by using the Levenshtein edit distance algorithm. The Levenshtein edit distance is a measure of how similar two strings are. It is calculated by counting the number of single-character edits that are required to transform one string into the other.

When performing a fuzzy search in Elasticsearch, you can specify the maximum number of allowed edits (insertions, deletions, or substitutions) that can be made to the search term. This is known as the fuzziness parameter.

The fuzziness value can be an absolute value, such as "1" to allow a maximum of one edit, or a relative value like "auto" to calculate the fuzziness based on the length of the search term.

9.What is the Search-as-You-Type functionality in Elasticsearch?

Search-as-You-Type functionality in Elasticsearch enables real-time suggestions and completion of search queries as users type in their input. It provides a responsive and interactive search experience by offering instant search suggestions based on the partial input entered by the user.

Elasticsearch implements search-as-you-type functionality through the search_as_you_type field. This field creates various sub-fields, which are then analyzed for indexing terms that can be effectively matched with a query for partial matching of the complete indexed text value.

By implementing the Search-as-You-Type functionality, you can significantly enhance the search experience for users, providing them with immediate feedback and relevant suggestions as they type.

10.How does Elasticsearch handle mappings?

In Elasticsearch, mappings define how documents and their fields are indexed and stored. They provide instructions to Elasticsearch on how to interpret and analyze the data within the documents. Elasticsearch handles mappings to ensure efficient indexing, search, and retrieval of data.

Elasticsearch handles mappings by using a process called "dynamic mapping." Dynamic mapping allows you to index documents without having to specify the mapping for each field in advance. When you index a document, Elasticsearch will automatically create a mapping for each field in the document.

11.What are the differences between Elasticsearch and Apache Kafka?

Elasticsearch and Apache Kafka are both open-source technologies that are used for data processing and analytics. However, they have different purposes and use cases.

- Data type: Elasticsearch is designed to index and search text data. Apache Kafka can be used to process and distribute any type of data, including text, binary data, and structured data.
- Use cases: Elasticsearch is often used for log analysis, real-time analytics, and full-text search. Apache Kafka is often used for event streaming, data integration, and streaming analytics.
- Performance: Elasticsearch is designed for high performance for search queries. Apache Kafka is designed for high performance for streaming data.
- Data retention: Elasticsearch provides configurable data retention but is not designed for long-term data storage. Kafka provides long-term data retention and durability.

12.How does Elasticsearch handle geolocation and geometry data fields?

Elasticsearch provides robust support for geolocation and geometry data fields, allowing you to index, search, and perform spatial operations on spatial data.
Elasticsearch handles geolocation and geometry data fields by using the geo_point and geo_shape data types.

The geo_point data type is used to store latitude and longitude coordinates, while the geo_shape data type can be used to store a variety of geometric shapes, such as points, lines, polygons, and circles.

13.Explain the Bulk API in Elasticsearch.

The Bulk API in Elasticsearch is a powerful feature that allows you to perform multiple indexing, updating, or deleting operations in a single API call. Instead of sending individual requests for each document, the Bulk API enables efficient batch processing, reducing network overhead and improving indexing or updating performance.

The Bulk API is a fundamental tool for efficiently managing large volumes of data in Elasticsearch. It helps optimize indexing and updating workflows, reduces network round-trips, and improves overall performance.

14.What is the difference between Lucene and Elasticsearch?

Lucene and Elasticsearch are closely related technologies, but they serve different purposes and have distinct characteristics.

Lucene is a powerful, high-performance, full-text search library written in Java. It provides indexing and search capabilities at the core level and serves as the foundation for many search-related applications. Lucene is a library, not a standalone server, and requires integration into an application or framework to be used effectively.

Elasticsearch is a distributed search and analytics engine built on top of Lucene. It provides a RESTful API and a distributed architecture, making it easy to scale horizontally and handle large volumes of data. Elasticsearch extends Lucene's functionality and offers additional features such as distributed search, real-time analytics, aggregations, and support for structured and unstructured data.

15.How can Elasticsearch security be improved?

Here are some tips on how to improve Elasticsearch security:

- Enable authentication and authorization: Elasticsearch supports a variety of authentication and authorization methods, such as basic authentication, LDAP authentication, and role-based access control (RBAC).
- Use a firewall to restrict access: You should use a firewall to restrict access to your Elasticsearch cluster. The firewall should only allow access from authorized IP addresses.
- Encrypt data in transit and at rest: You should encrypt data in transit and at rest to protect your data from unauthorized access. Elasticsearch supports a variety of encryption methods, such as TLS and S3 encryption.

- Implement monitoring and alerting: Set up monitoring and alerting systems to detect and respond to any anomalies or security-related events in real time.

16.Describe the role of machine learning in Elasticsearch.

Machine learning can be used to improve the performance and accuracy of Elasticsearch in a number of ways.

- Anomaly detection: Machine learning can be used to identify anomalies in data that is stored in Elasticsearch. This can be useful for identifying potential problems, such as fraud or security breaches.
- Predictive Analytics: Machine learning can be used to predict future events based on data that is stored in Elasticsearch. This can be useful for making business decisions, such as forecasting demand or predicting customer behavior.
- Natural language processing (NLP): Machine learning can be used to improve NLP in Elasticsearch. This can be useful for tasks such as search, translation, and text analysis.

17.What is Elasticsearch caching?

Elasticsearch caching is a mechanism that improves query performance by storing frequently accessed data in memory. It helps reduce the need for repetitive computations and disk I/O, resulting in faster response times for subsequent queries.

Elasticsearch employs various caching techniques to optimize performance, including filter cache, field data cache, and query cache.

18.Explain the need for an index template in Elasticsearch.

An index template is used to suggest to the Elasticsearch engine the ways in which the index can be configured when it is created. An index template is a predefined configuration that defines how new indices should be created and what settings, mappings, and aliases they should have.

Here are some of the reasons why you need index templates:

- Reduced configuration time: Index templates can reduce the configuration time for new indices by allowing you to reuse existing settings and mappings.
- Increased consistency: Index templates can help to ensure consistency across your indices by applying the same settings and mappings to all indices that match the template pattern.
- Improved performance: Index templates can improve the performance of your indices by allowing you to pre-configure settings and mappings.
- Simplified management: Index templates can simplify the management of your indices by providing a central location to store and manage settings and mappings.

19.What are the differences between Elasticsearch and Amazon DynamoDB?

Elasticsearch and Amazon DynamoDB are both popular database technologies, but they serve different purposes and have distinct characteristics.

Data Model: Elasticsearch is a full-text search engine and analytics platform that is built on top of Apache Lucene. It is designed for fast and efficient searching, indexing, and analysis of unstructured or semi-structured data. DynamoDB is a NoSQL database service offered by Amazon Web Services (AWS). It is a key-value store that allows you to store and retrieve structured data with flexible schemas.

Consistency and Durability: DynamoDB is known for its strong consistency model, meaning that once a write operation is successful, subsequent read operations will always return the latest data. Elasticsearch, by default, provides eventual consistency, which means that after a write operation, it may take a short period for all nodes to reflect the updated data.

20.How does Elasticsearch handle rebalancing and shard allocation?

Elasticsearch handles rebalancing and shard allocation in the following way:

- Shard allocation: Elasticsearch uses a number of factors to determine where to allocate shards, including the number of shards on each node, the amount of disk space available on each node, and the node's CPU and memory usage. Elasticsearch also allows you to configure custom allocation rules.
- Shard rebalancing: Elasticsearch automatically rebalances shards when necessary. This can happen when a new node joins the cluster, when a node leaves the cluster, or when a node's disk space becomes full. Elasticsearch also allows you to manually rebalance shards.

21.Explain Elasticsearch's role in data analysis.

Elasticsearch, a distributed search and analytics engine, plays a crucial role in data analysis and visualization by providing fast, scalable, and near real-time storage, indexing and querying capabilities.

Log analysis: Elasticsearch can be used to collect and analyze system logs from applications, servers, and networks. This information can be used to identify problems, troubleshoot issues, and improve performance.

Security analytics: Elasticsearch can be used to collect and analyze security logs from firewalls, intrusion detection systems, and other security devices. This information can be used to identify threats, investigate incidents, and improve security posture.

Business intelligence: Elasticsearch can be used to collect and analyze business data from a variety of sources, including customer data, financial data, and operational data. This information can be used to gain insights into customer behavior, identify trends, and make better business decisions.

22.What is an Elasticsearch analyzer whitelist?

In Elasticsearch, an analyzer whitelist refers to a configuration option that allows you to restrict the use of specific analyzers for indexing or searching your data. It allows you to define a list of analyzers that are permitted to be used, while excluding others.

By default, Elasticsearch provides a wide range of built-in analyzers that cater to various language-specific requirements, tokenization rules, and stemming algorithms. These analyzers are designed to handle different types of text data effectively.

However, in some cases, you may want to limit the analyzers that can be applied to your data to ensure consistency or enforce specific language or domain-specific rules. This is where the Elasticsearch analyzer whitelist can be particularly useful.

23.Describe Elasticsearch snapshot and restore operations. How is Elasticsearch snapshot and restore features beneficial for users?

Elasticsearch snapshot and restore operations are used to backup and restore data from an Elasticsearch cluster. Snapshots are created by copying the data from the cluster to a remote location, such as a cloud storage service or a local disk.

Restores are performed by copying the data from the snapshot back to the cluster. Snapshots and restores can be used for a variety of purposes, such as backup, restore, data migration, and data sharing.

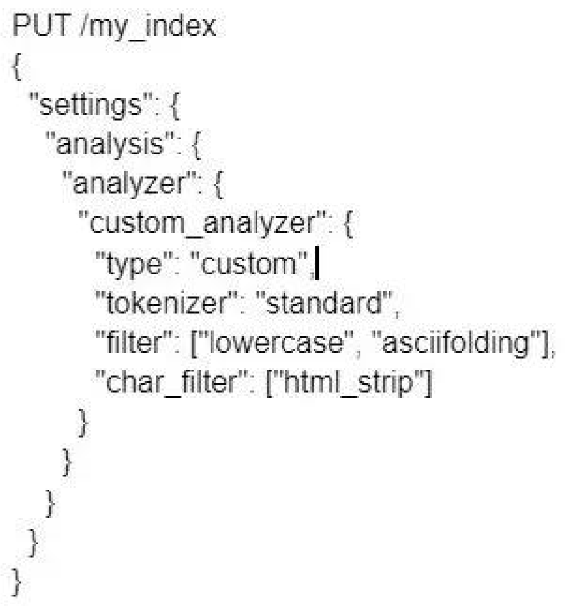
The snapshot and restore features in Elasticsearch provide important benefits to users in terms of data backup, disaster recovery, and data migration. Here are some key advantages:

- Data Backup: By taking regular snapshots, users can protect their data from accidental deletions, hardware failures, or other data loss scenarios.
- Disaster Recovery: In the event of a cluster failure or data corruption, the restore capability enables users to recover their data from previously created snapshots.
- Data Migration: Users can create a snapshot in one cluster and restore it in another, making it convenient to transfer data between development, staging, and production environments.
- Incremental Backups: Elasticsearch supports incremental snapshots which reduces the storage and network requirements for backups, allowing users to efficiently manage their backup infrastructure.
- Granular Recovery: Users can restore only the necessary data instead of restoring the entire snapshot, which saves time and

storage space. This granular recovery capability is valuable in situations where only a subset of data needs to be recovered.

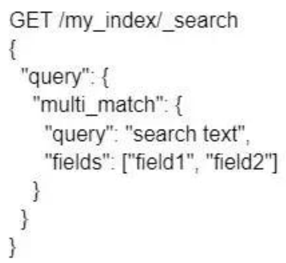
24.How can you create a custom analyzer in Elasticsearch?

You can create a custom analyzer using the PUT API method, analysis, and specifying the tokenizer, filter, and char_filter configuration.



```
PUT /my_index
{
  "settings": {
    "analysis": {
      "analyzer": {
        "custom_analyzer": {
          "type": "custom",
          "tokenizer": "standard",
          "filter": ["lowercase", "asciifolding"],
          "char_filter": ["html_strip"]
        }
      }
    }
  }
}
```

25.How do you search across multiple fields in Elasticsearch?

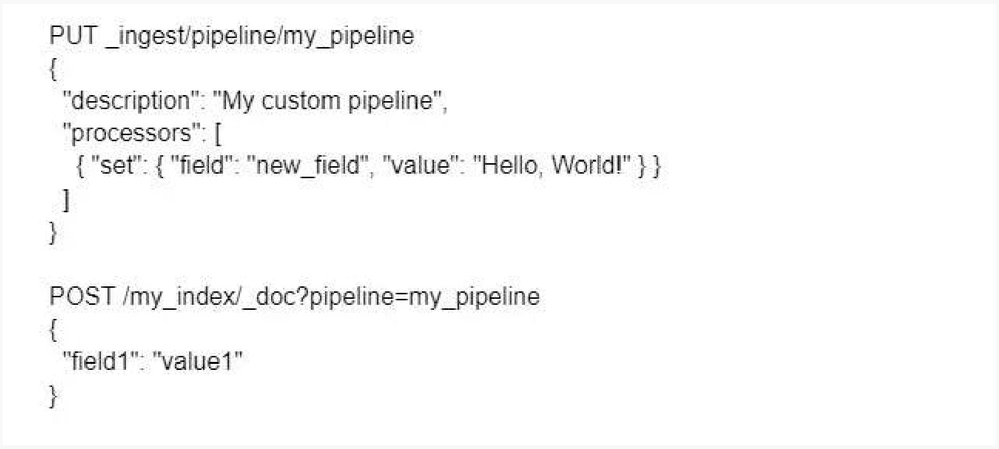


```
GET /my_index/_search
{
  "query": {
    "multi_match": {
      "query": "search text",
      "fields": ["field1", "field2"]
    }
  }
}
```

You can use the multi_match query type to search across multiple fields

26.How can you index a document by defining a pipeline during ingestion?

You can use the “PUT” API method to create a pipeline and index API to define pipeline during ingestion.

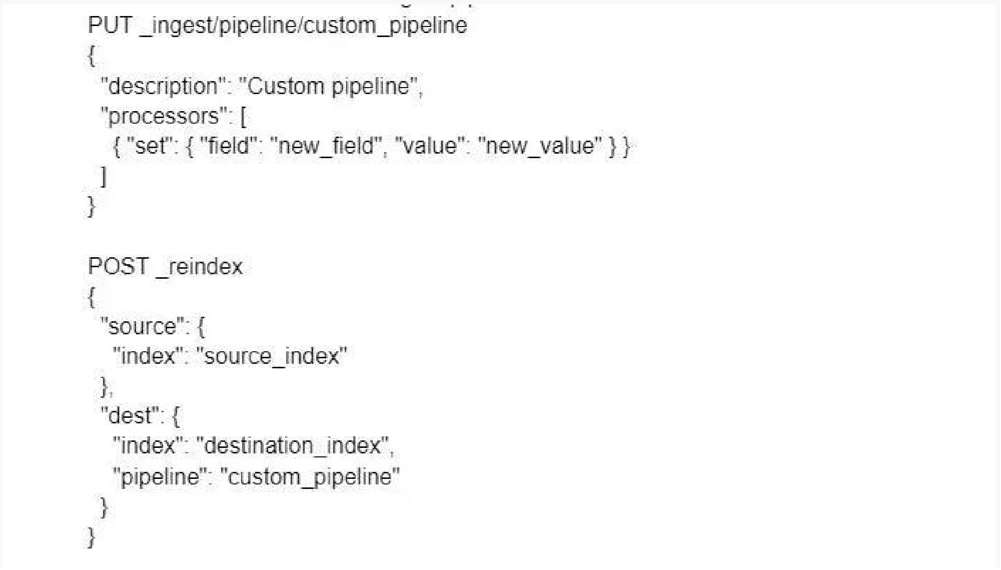


```
PUT _ingest/pipeline/my_pipeline
{
  "description": "My custom pipeline",
  "processors": [
    { "set": { "field": "new_field", "value": "Hello, World!" } }
  ]
}

POST /my_index/_doc?pipeline=my_pipeline
{
  "field1": "value1"
}
```

27.How do you reindex data in Elasticsearch with custom transformation?

Use the Reindex API with an ingest pipeline for transformations.

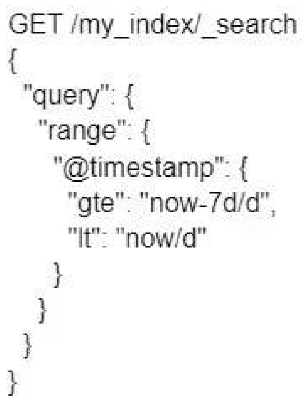


```
PUT _ingest/pipeline/custom_pipeline
{
  "description": "Custom pipeline",
  "processors": [
    { "set": { "field": "new_field", "value": "new_value" } }
  ]
}

POST _reindex
{
  "source": {
    "index": "source_index"
  },
  "dest": {
    "index": "destination_index",
    "pipeline": "custom_pipeline"
  }
}
```

28.How do you perform a date range search in Elasticsearch using the Query DSL?

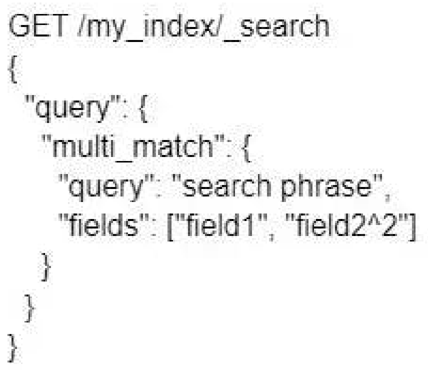
To perform a date range search in Elasticsearch using the Query DSL, you need to use the range query type.



```
GET /my_index/_search
{
  "query": {
    "range": {
      "@timestamp": {
        "gte": "now-7d/d",
        "lt": "now/d"
      }
    }
  }
}
```

29.How can you apply a boost factor to a specific field during search query execution?

To apply a boost factor to a specific field during search query execution, you need to use the boost attribute on the match query.

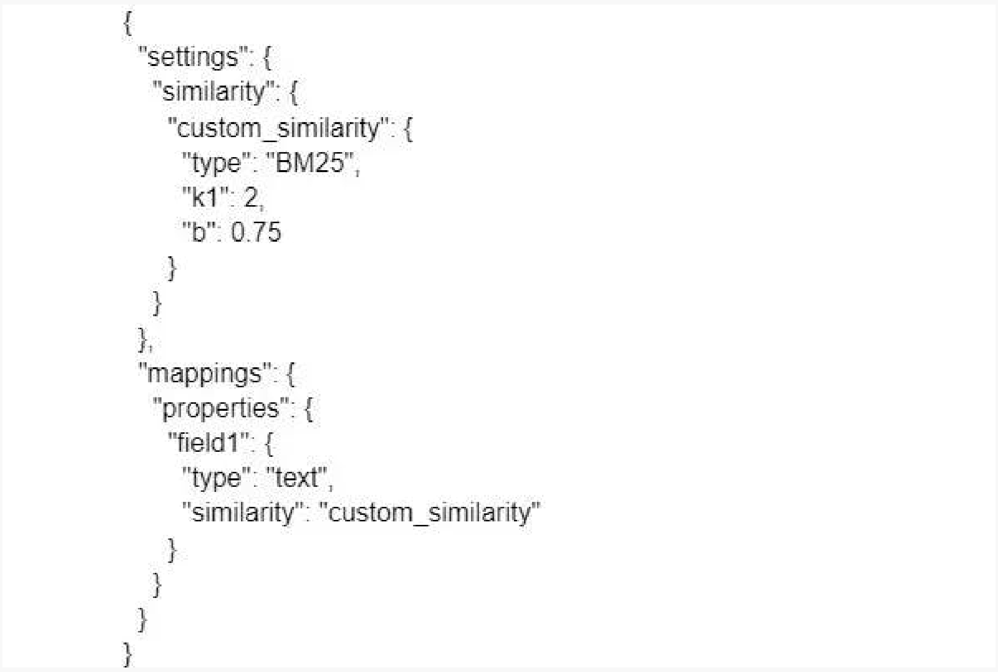


```
GET /my_index/_search
{
  "query": {
    "multi_match": {
      "query": "search phrase",
      "fields": ["field1", "field2^2"]
    }
  }
}
```

30.How do you configure Elasticsearch to use a custom similarity algorithm for ranking documents in search results?

To configure Elasticsearch to use a custom similarity algorithm for ranking documents in search results, you will have to define a custom similarity in the index settings and apply it to the field mapping.

PUT /my_index



```
{
  "settings": {
    "similarity": {
      "custom_similarity": {
        "type": "BM25",
        "k1": 2,
        "b": 0.75
      }
    }
  },
  "mappings": {
    "properties": {
      "field1": {
        "type": "text",
        "similarity": "custom_similarity"
      }
    }
  }
}
```

31.How can you implement a search-as-you-type feature using the edge_ngram tokenizer?

Create a custom analyzer and apply it to field mapping.

```
PUT /my_index
{
  "settings": {
    "analysis": {
      "tokenizer": {
        "edge_ngram_tokenizer": {
          "type": "edge_ngram",
          "min_gram": 1,
          "max_gram": 10,
          "token_chars": ["letter"]
        }
      },
      "analyzer": {
        "search_completion_analyzer": {
          "type": "custom",
          "tokenizer": "edge_ngram_tokenizer",
          "filter": ["lowercase"]
        }
      }
    }
  },
  "mappings": {
    "properties": {
      "name": {
        "type": "text",
        "analyzer": "search_completion_analyzer",
        "search_analyzer": "standard"
      }
    }
  }
}
```

32.How can you add synonyms to a text search in Elasticsearch?

Create a custom analyzer with a synonym filter.

```
PUT /my_index
{
  "settings": {
    "analysis": {
      "filter": {
        "synonym_filter": {
          "type": "synonym",
          "synonyms": [
            "big, large",
            "small, tiny"
          ]
        }
      },
      "analyzer": {
        "synonym_analyzer": {
          "tokenizer": "standard",
          "filter": ["lowercase", "synonym_filter"]
        }
      }
    }
  },
  "mappings": {
    "properties": {
      "name": {
        "type": "text",
        "analyzer": "synonym_analyzer"
      }
    }
  }
}
```