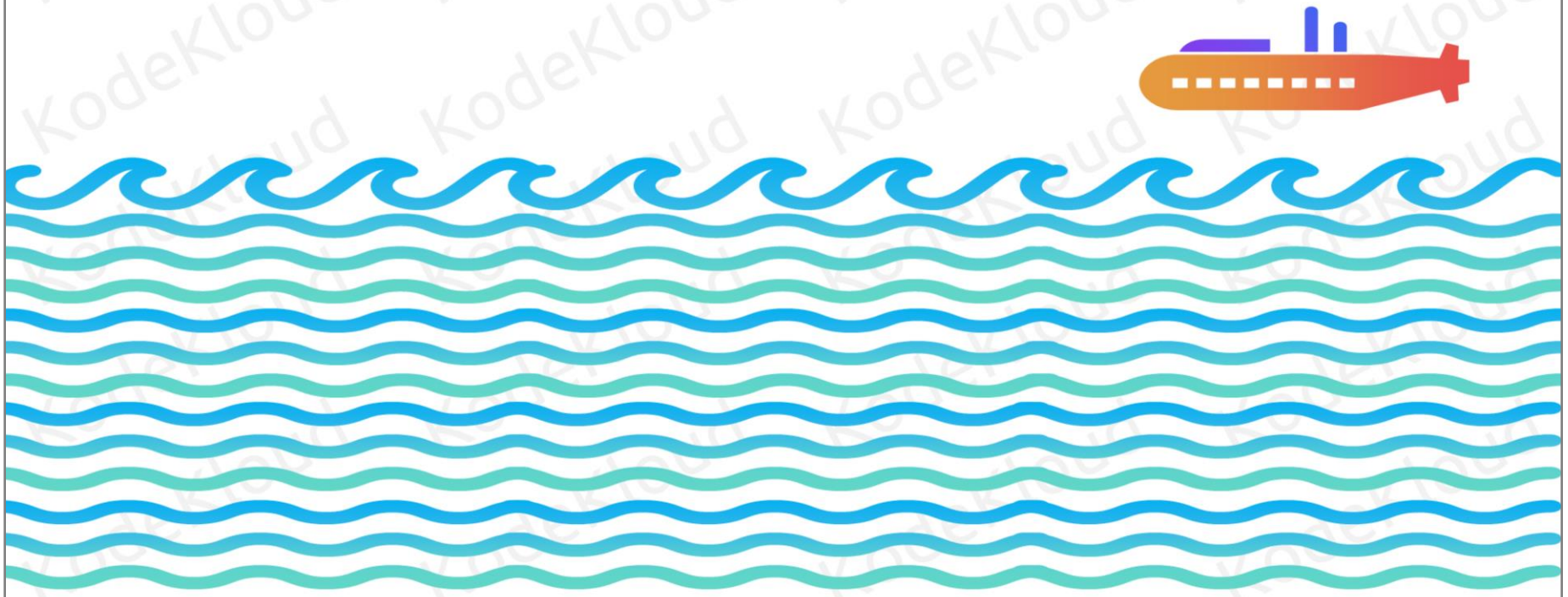
KodeKloud

Follow us on https://kodekloud.com/ to learn more about us.

Elasticsearch FluentBit and Kibana – Introduction



Mastering Elasticsearch Fundamentals

Elasticsearch

1 | Nodes

2 | Cluster

3 | Index

4 | Document

5 | CRUD Operations

6 | Mapping

Introduction to the EFK Stack

Kibana

1 | Kibana visualizations and dashboards

2 | Using Kibana's Discover interface for log exploration

3 | Kibana Query Language (KQL)

FluentBit

1 | Logstash's role within the ELK stack

2 | FluentBit vs FluentD

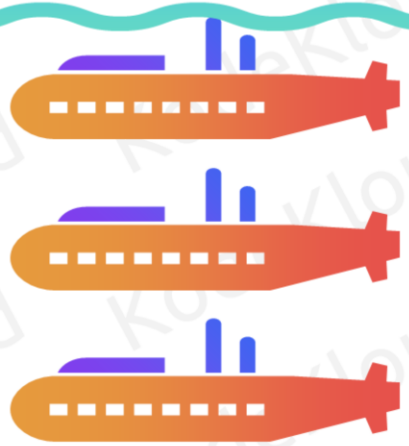
3 | Logstash vs FluentBit

4 | FluentBit Configurations

01 Setting up FluentBit to monitor K8 infra

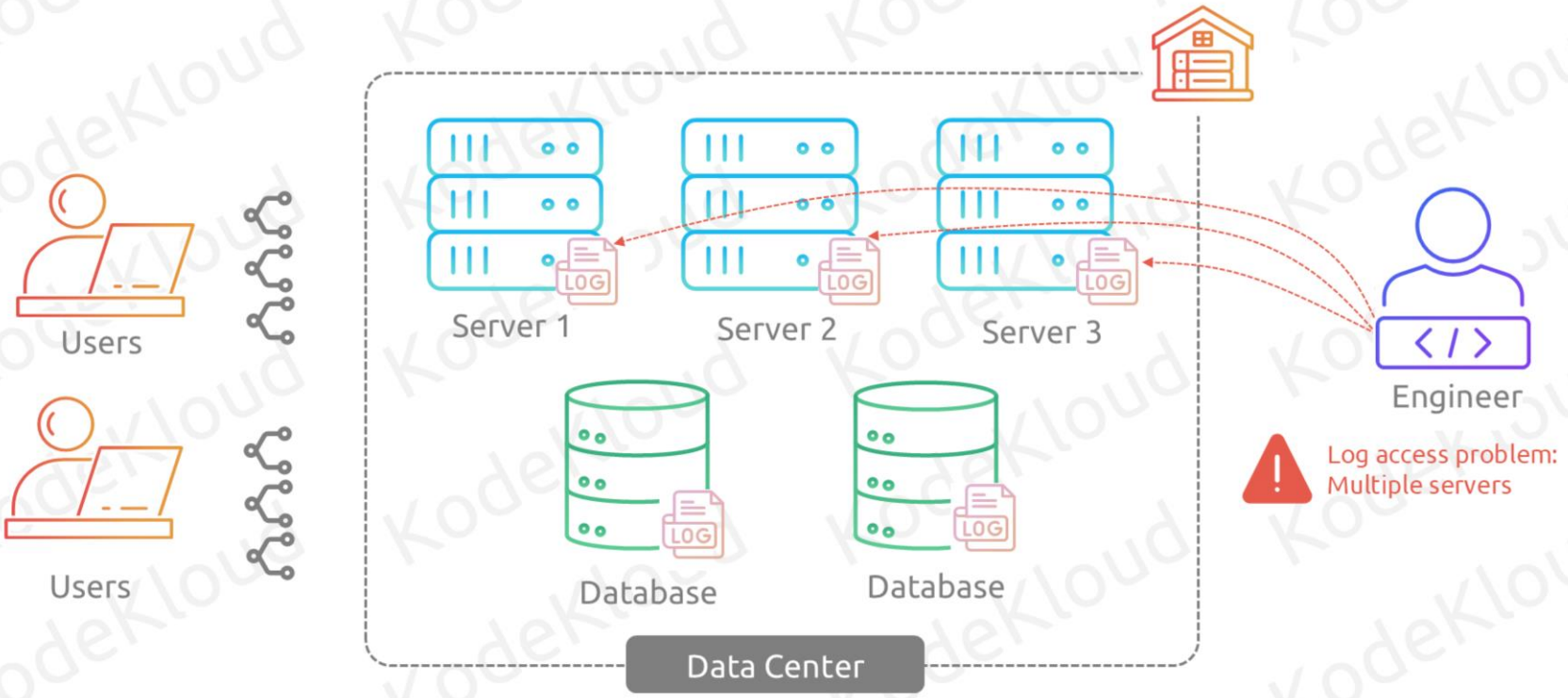
02 Monitoring a mock e-commerce app application with FluentBit and Elasticsearch

03 Monitoring a Python application with FluentBit and Elasticsearch



What Is Logging
and
Why Does It Matter?

Multi-Server Log Management Challenges

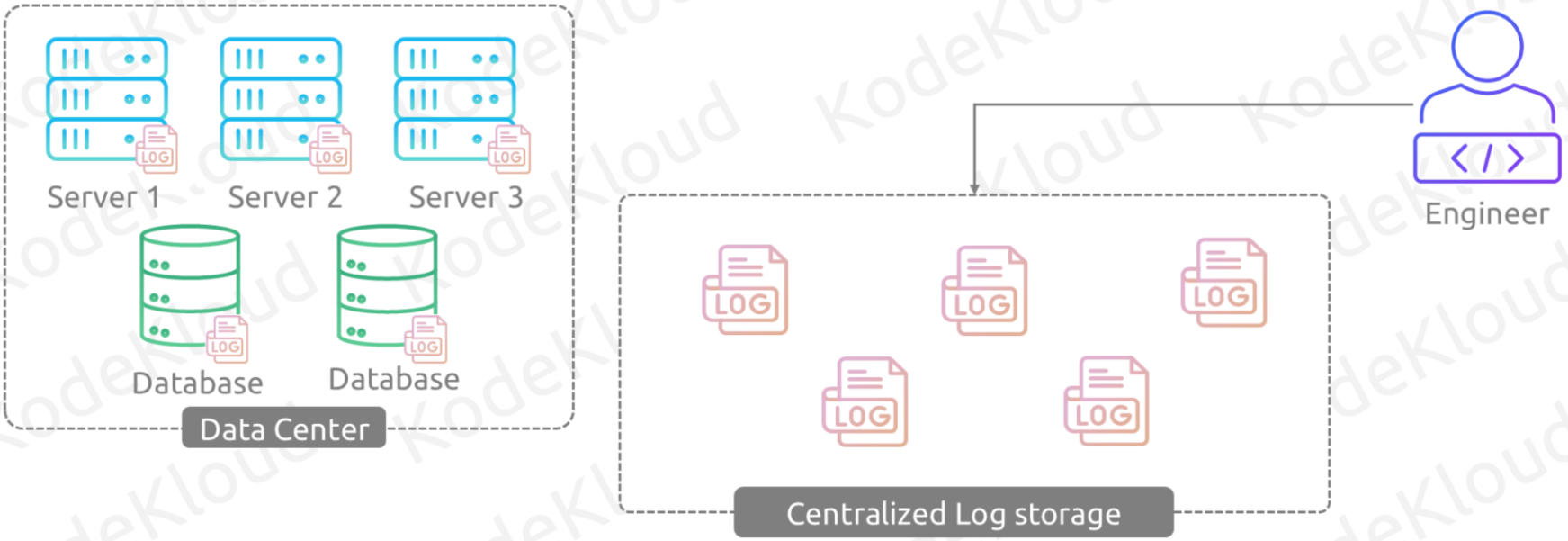


Microservices and Containerization

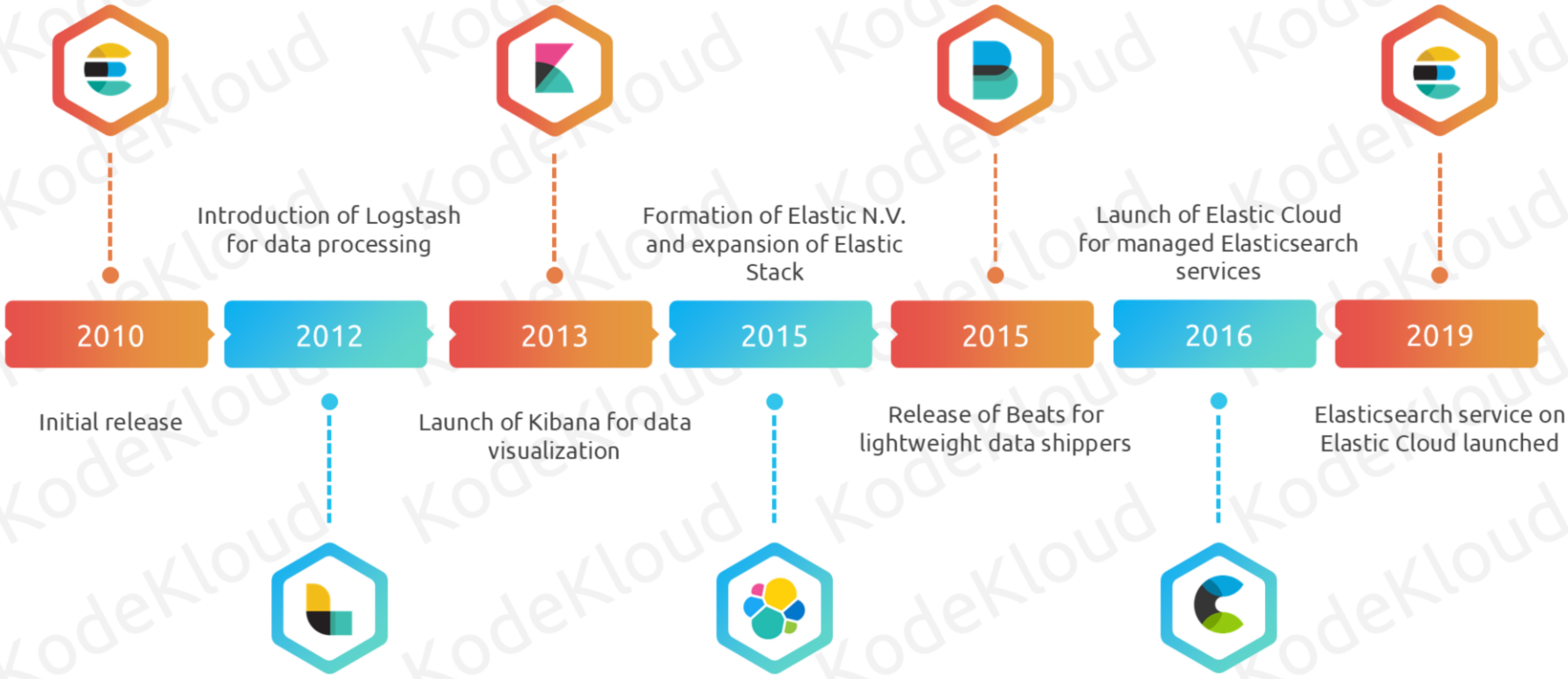
Microservices Architecture

Docker

Imagining Centralized Logging

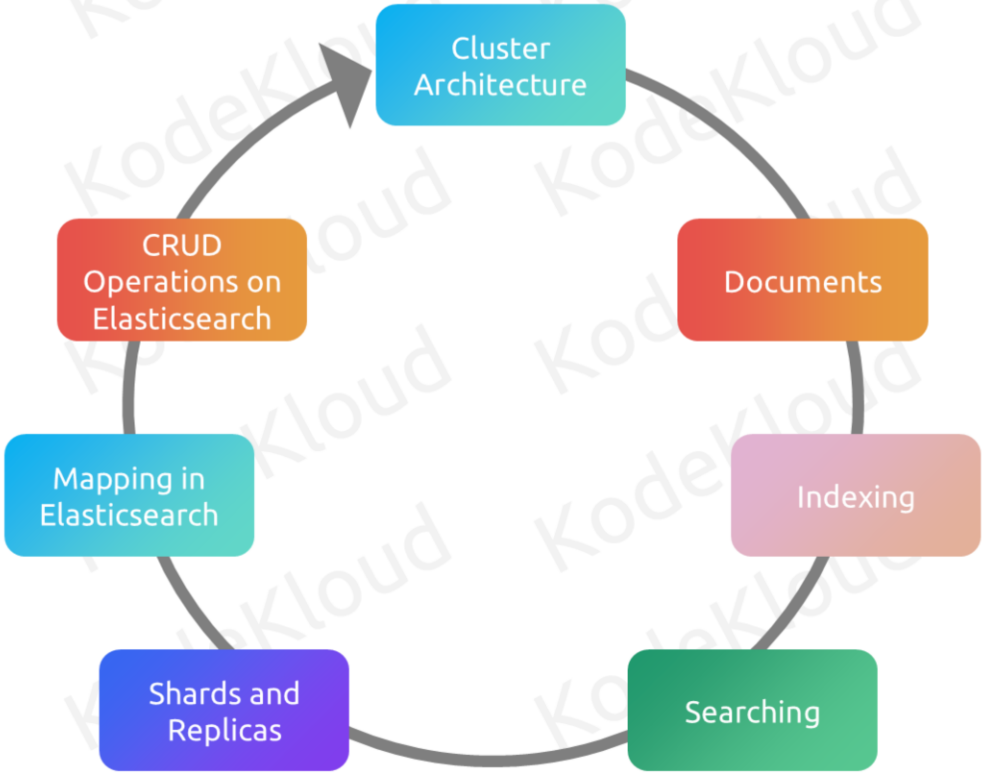


Elasticsearch and Its Evolution



Mastering Fundamentals of Elasticsearch

Elasticsearch – What We Will Learn



Nodes and Cluster in Elasticsearch

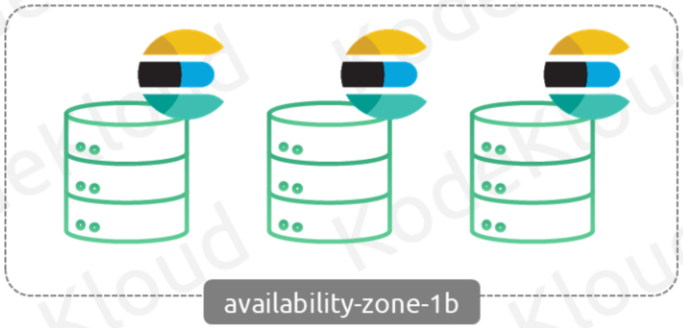
This is a **single-node** elasticsearch setup

Recommended Node Configuration

Memory: 16GB
CPU: 8 Cores
Disk: SSDs
Network: At least 1 Gbps network connection

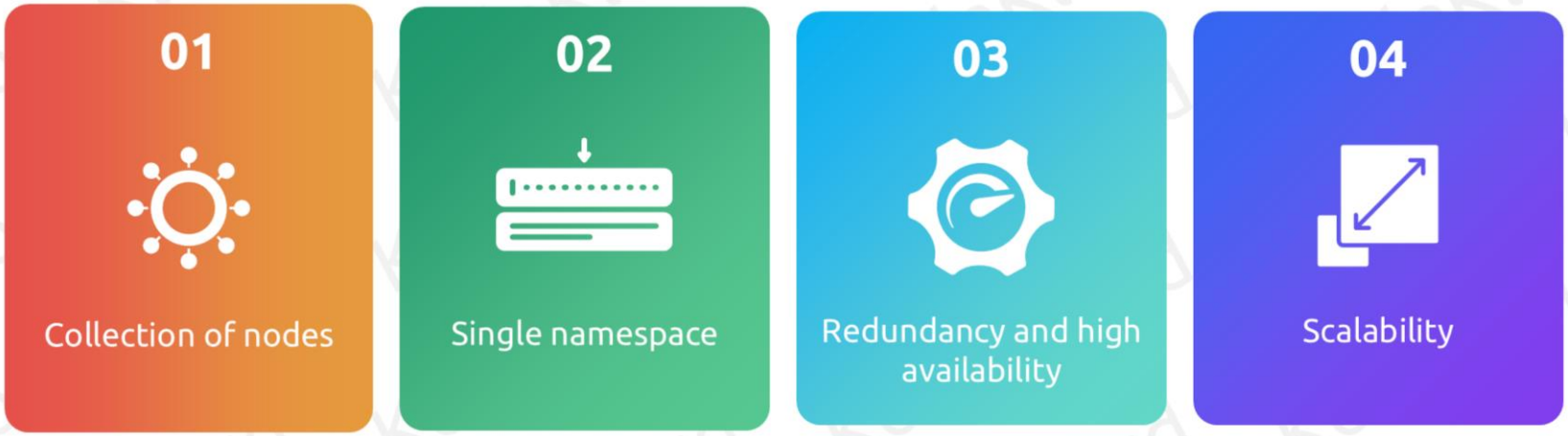
Nodes and Cluster in Elasticsearch

Nodes and Cluster in Elasticsearch



But do all nodes in the cluster play the same role ?

Nodes and Cluster in Elasticsearch



NodeRoles in Elasticsearch

Master Node

Data Node

Data Content

Data Hot

Data Warm

Data Cold

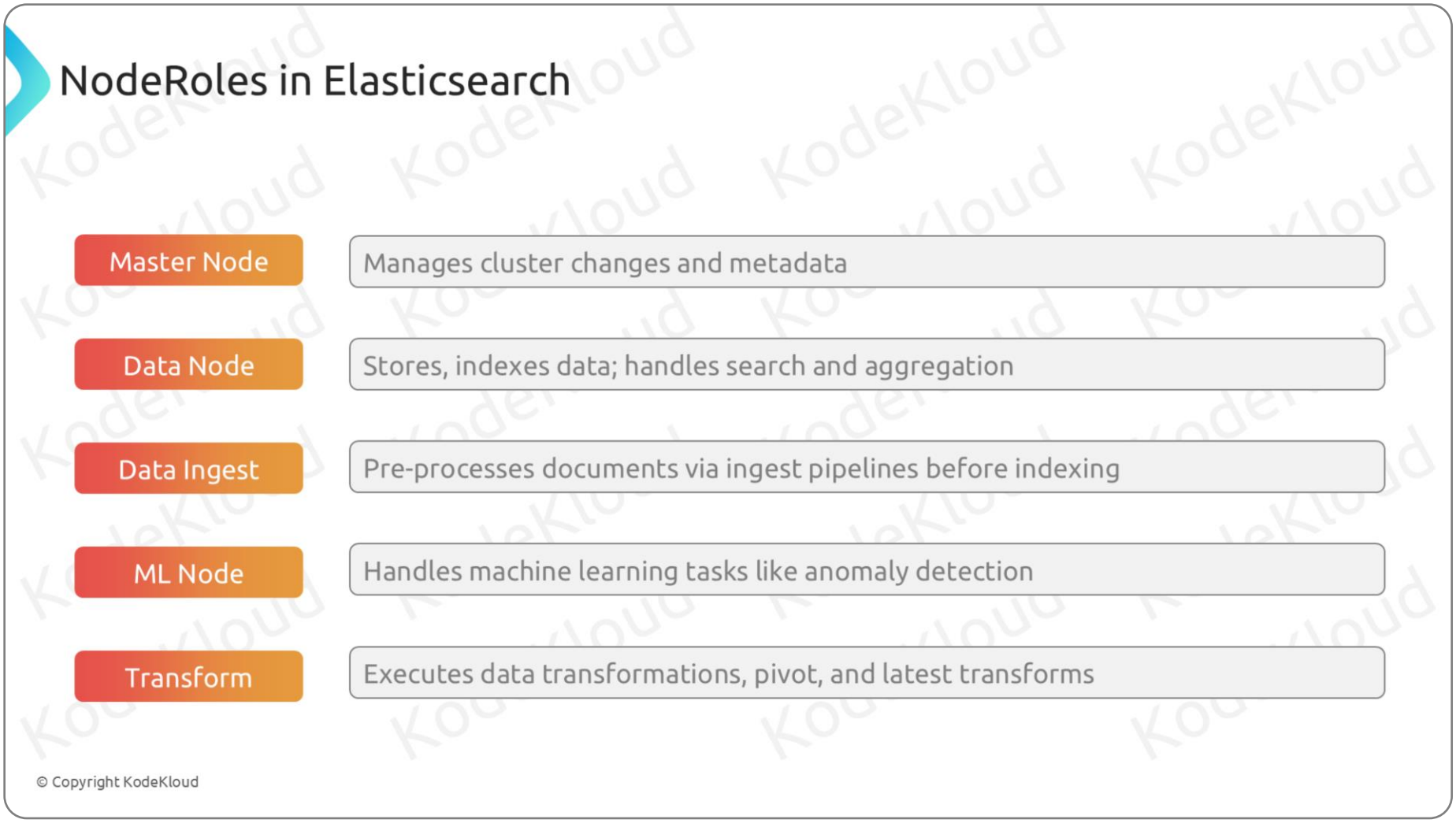
Data Frozen

Data Ingest

ML Node

Transform

Remote Cluster Client



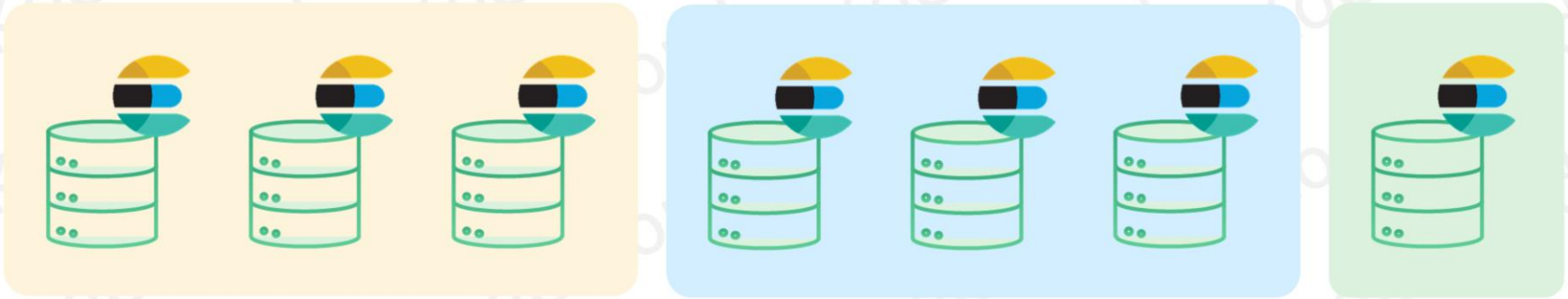
Master Node-Manages the cluster by handling changes and maintaining cluster metadata.
Data Node-Stores and indexes data, handling search and aggregation requests
Data Ingest-Pre-processes documents before indexing using ingest pipelines.
ML Node-Performs machine learning tasks like anomaly detection within the cluster.
Transform-Executes data transformations and operations such as pivot and latest transforms.

NodeRoles in Elasticsearch

Remote Cluster Client	Acts as a gateway for cross-cluster searches
Data Cold	Stores infrequently accessed data, cost-optimized over performance
Data Frozen	Holds rarely accessed data, prioritizing cost efficiency with high latency
Data Hot	Stores frequently accessed data, optimized for low latency and high performance
Data Warm	Balances storage efficiency and performance for moderately accessed

Remote Cluster Client-Facilitates cross-cluster search by acting as a gateway to remote clusters.
Data Cold-Contains infrequently accessed data, optimized for cost-effective storage over performance.
Data Frozen-Stores rarely accessed data, utilizing the most cost-efficient storage with the highest latency.
Data Hot-Stores frequently accessed and updated data, optimized for low latency and high performance.
Data Warm-Holds less frequently accessed data, balancing performance and storage efficiency.

NodeRoles in Elasticsearch



Master Node | Data Node | Data Content | Data Hot | Data Warm

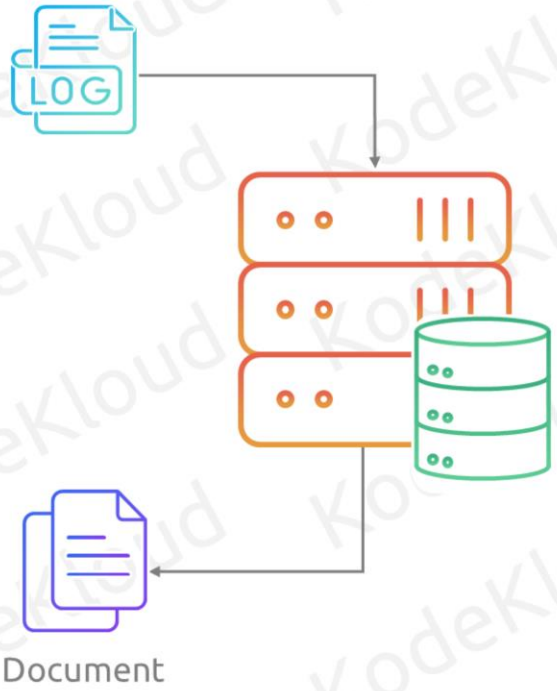
Data Cold | Data Frozen | Data Ingest

ML Node | Transform

Remote Cluster Client

So here I wanted to sort or move these roles to different servers and show how when having a big cluster of elasticsearch each node could be playing different roles.
Here I am using same server type but this can be different server icons

What Gets Stored in Elasticsearch?



In Elasticsearch, a document is a basic unit of information that can be indexed and stored within the platform. Documents are stored in a structured JSON format and are used to perform and optimize search operations in real-time

What Gets Stored in Elasticsearch?

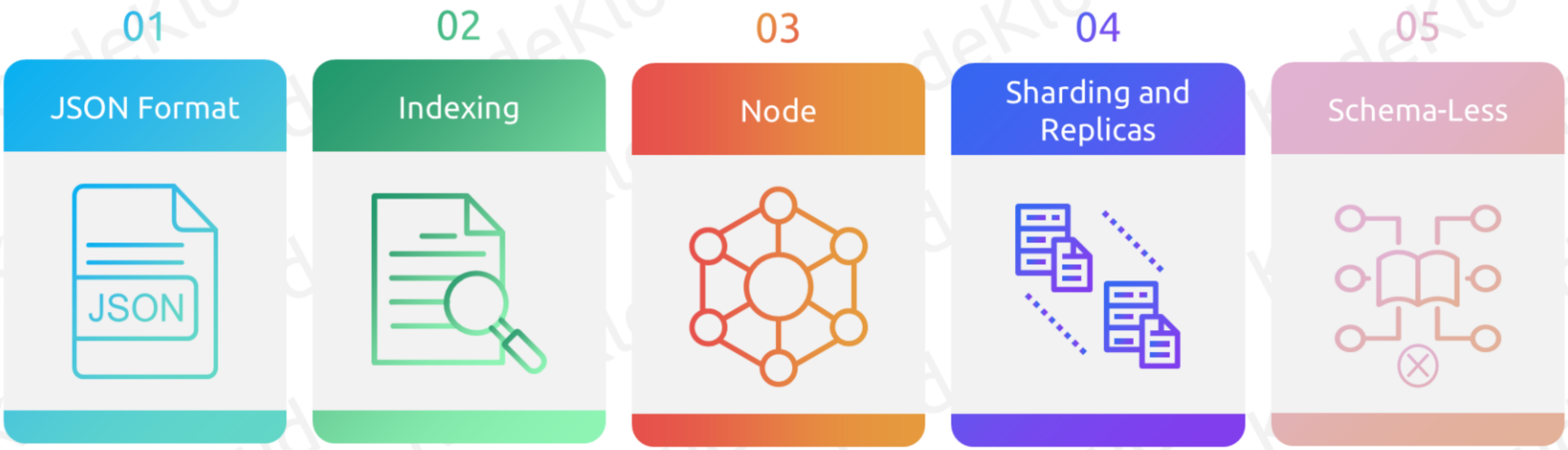
Document

1 | Fundamental unit of data

2 | Stored in structured JSON

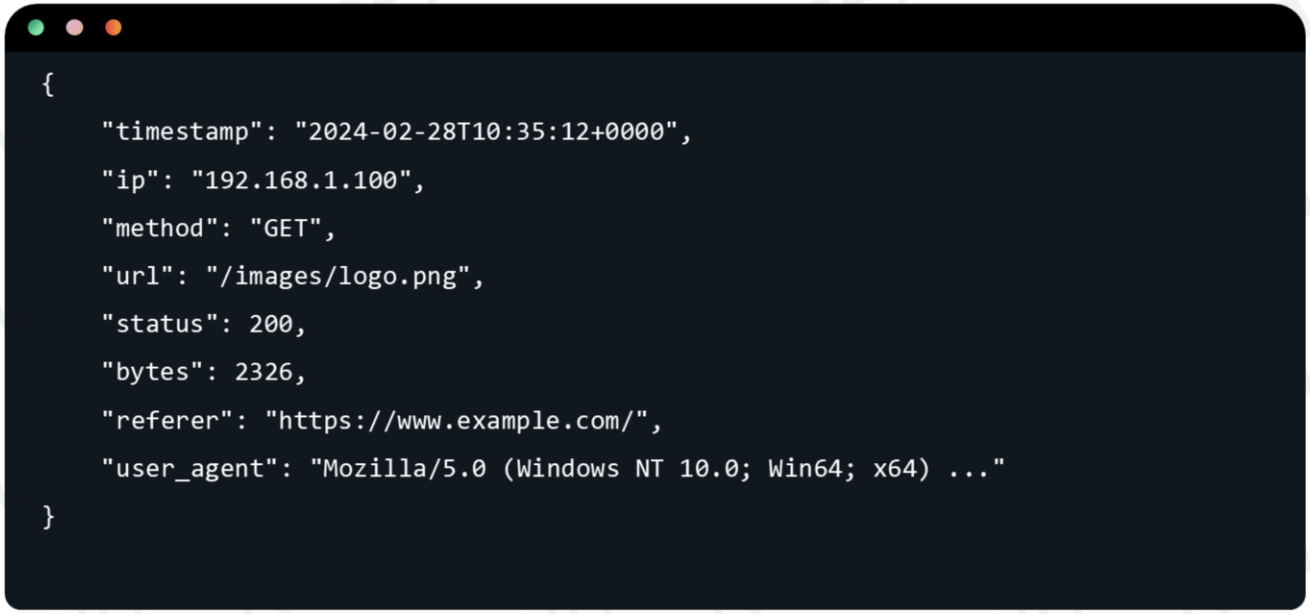
3 | Optimized for real-time search

Features of a Document in Elasticsearch



- JSON Format: Documents are stored in JSON format, which is a lightweight data interchange format.
- Indexing: Documents are indexed within an Elasticsearch index, which is a collection of similar types of documents.
- Node: Documents are stored on nodes, which are individual servers that are part of a larger Elasticsearch cluster.
- Sharding and Replicas: Data is split across multiple nodes using sharding, and replicas are copies of an index's shards to ensure data availability and redundancy.
- Schema-Less: Elasticsearch does not require a predefined schema for documents, allowing for flexible data structures and dynamic mapping

Documents in Elasticsearch



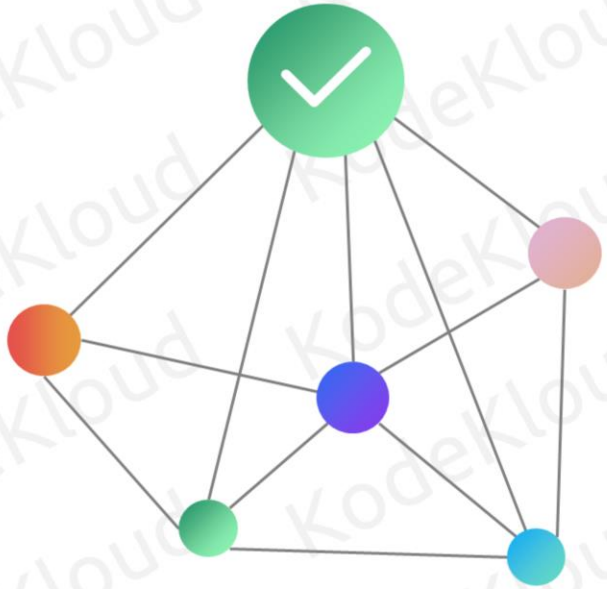
```
{
    "timestamp": "2024-02-28T10:35:12+0000",
    "ip": "192.168.1.100",
    "method": "GET",
    "url": "/images/logo.png",
    "status": 200,
    "bytes": 2326,
    "referer": "https://www.example.com/",
    "user_agent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64) ..."
}
```

Entry in a log file

Documents in Elasticsearch


```
{
        "_index": "webserver_logs",
        "_type": "_doc",
        "_id": "DD46948BFqy1faFsINf8",
        "_score": 0.2876821,
        "_source": {
          "timestamp": "2024-02-28T10:35:12+0000",
          "ip": "192.168.1.100",
          "method": "GET",
          "url": "/images/logo.png",
          "status": 200,
          "bytes": 2326,
          "referer": "https://www.example.com/",
          "user_agent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64) ..."
        }
}
```

Inverted Index in Elasticsearch



Elasticsearch uses an **inverted index** for rapid **full-text searches**.

- An inverted index is the core data structure used by Elasticsearch to enable fast full-text searches.

Inverted Index in Elasticsearch

Book's Index

Term	Page Numbers
Python	24, 56, 78
Java	12, 45, 67, 89
Database	3, 34, 56, 78, 90
Algorithm	1, 23, 45, 67, 89

Inverted Index

Term	Document IDs
Python	101, 202, 303
Java	303, 404, 505
Database	404, 505, 606
Algorithm	707, 808, 909

Similar to a book's index, Inverted Index maps terms to the documents where they appear.

- It is similar to the index at the end of a book that maps terms to the documents they appear in.

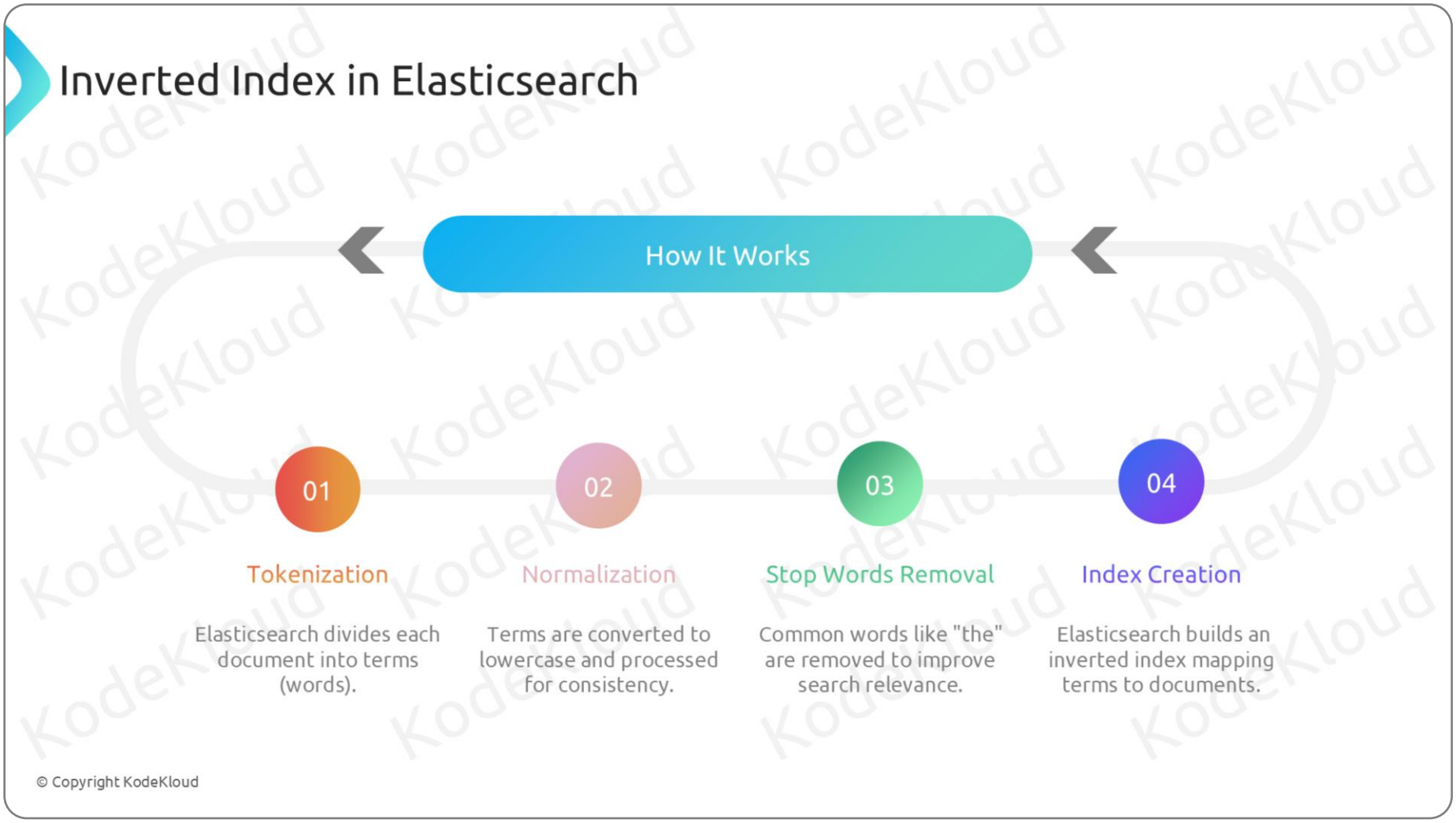
Inverted Index in Elasticsearch

Let's say you have three documents in Elasticsearch:

Document 1: "The quick brown fox jumps over the lazy dog."
Document 2: "The dog chased the cat."
Document 3: "The fox is cunning."

Inverted Index structure

Term	Document IDs
the	1, 2, 3
quick	1
brown	1
fox	1, 3
jumps	1
over	1
lazy	1
dog	1, 2
chased	2
cat	2
is	3
cunning	3



How It Works
Tokenization: Elasticsearch breaks down each document into individual terms (words).
Normalization: It converts terms to lowercase and performs other operations to ensure consistency.
Stop Words Removal: Common words like "the" are often removed as they don't add much value to searches.
Index Creation: The inverted index is built, associating each term with the documents it appears in.

Inverted Index in Elasticsearch

Inverted Index structure

Term	Document IDs
the	1, 2, 3
quick	1
brown	1
fox	1, 3
jumps	1
over	1
lazy	1
dog	1, 2
chased	2
cat	2
is	3
cunning	3

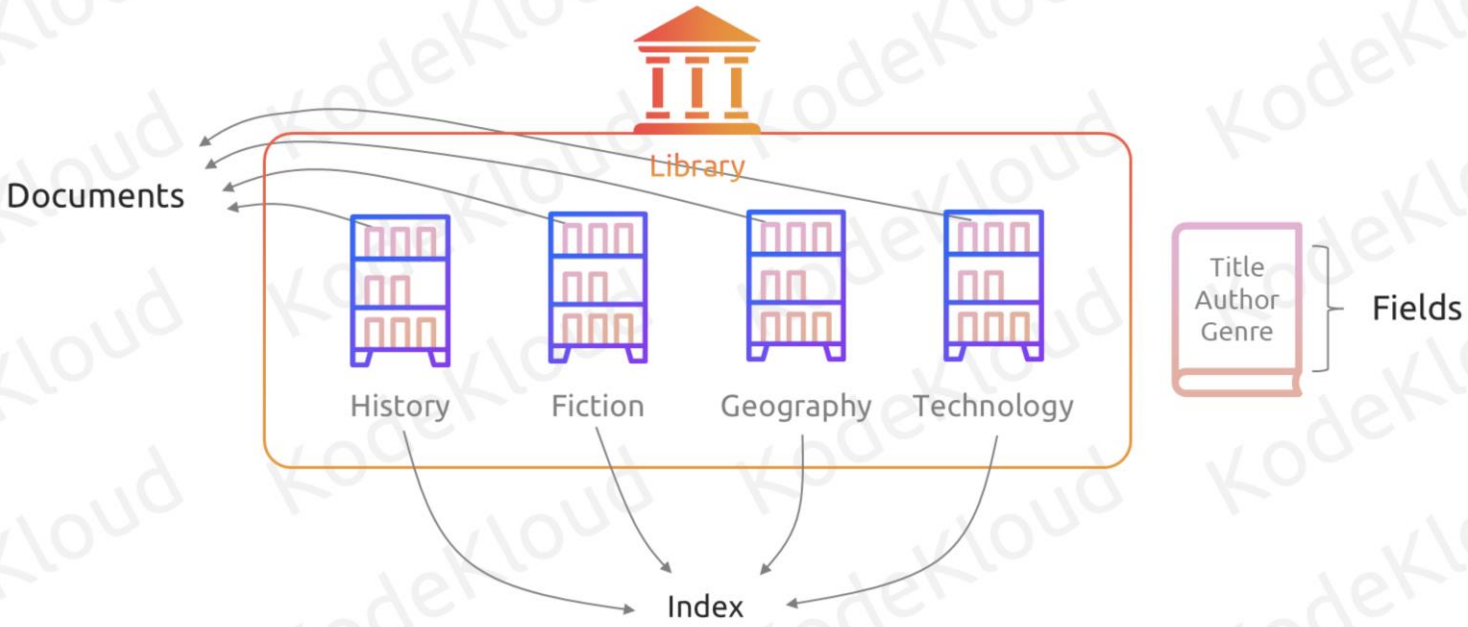
Conduct a search for "fox dog"

It returns document 1 as the result.

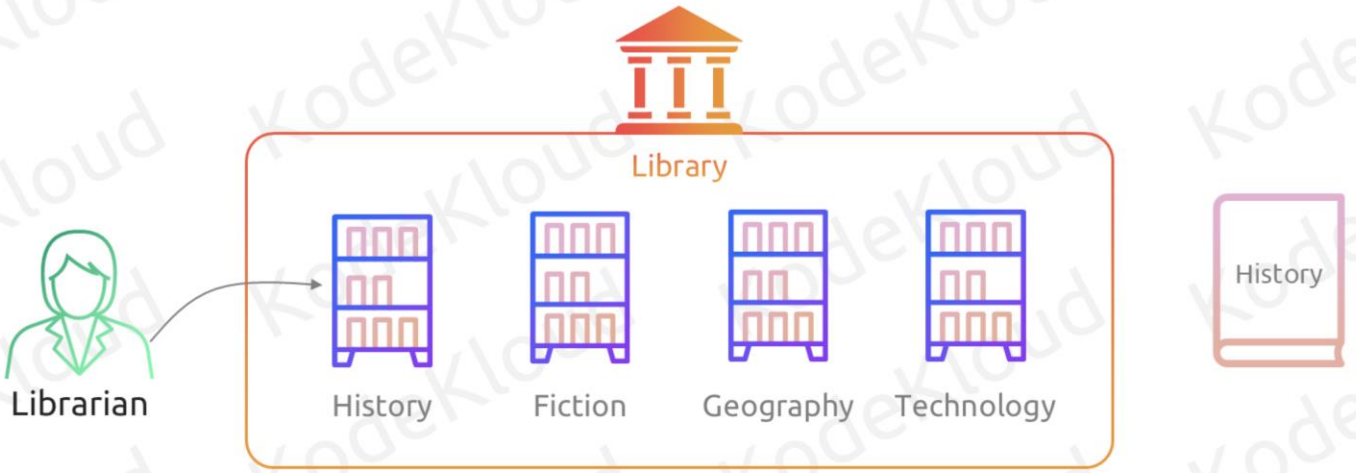
Searching with an Inverted Index

When you search for "fox dog", Elasticsearch consults the inverted index:
It looks up the document IDs associated with "fox" (1, 3).
It looks up the document IDs associated with "dog" (1, 2).
It finds the **intersection** of these two sets of document IDs (1).
It returns document 1 as the result.

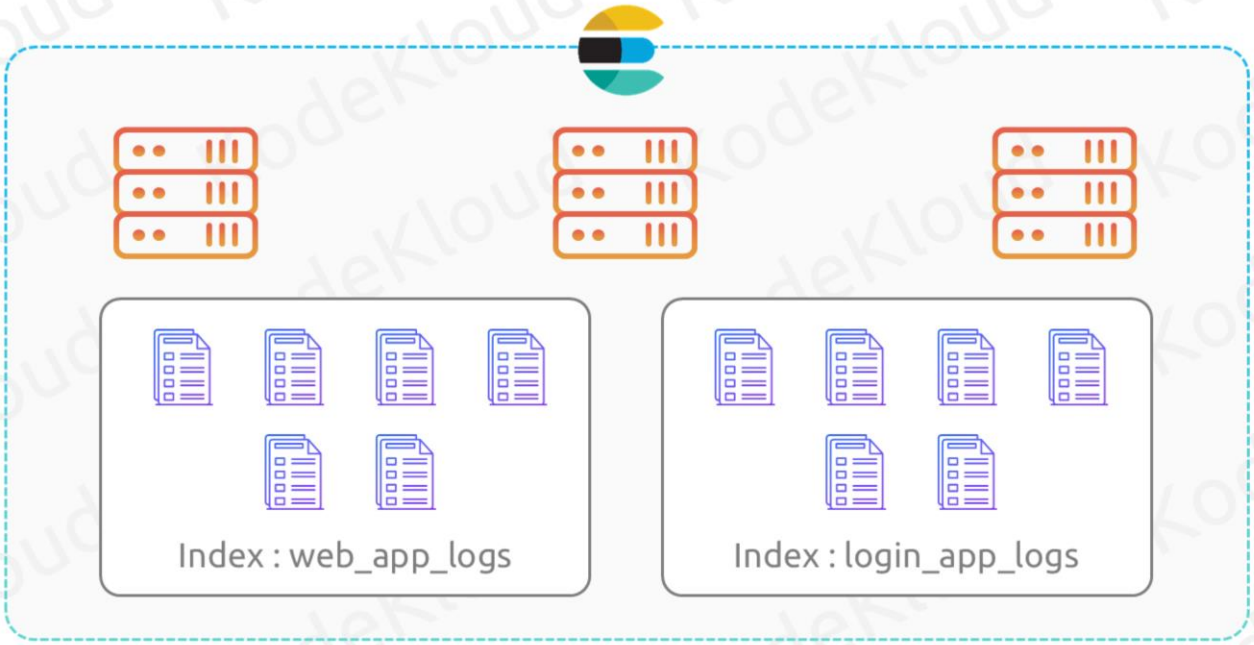
Index in Elasticsearch



Index in Elasticsearch



Index in Elasticsearch



Index in Elasticsearch

01

Logical Container

02

Documents and Fields

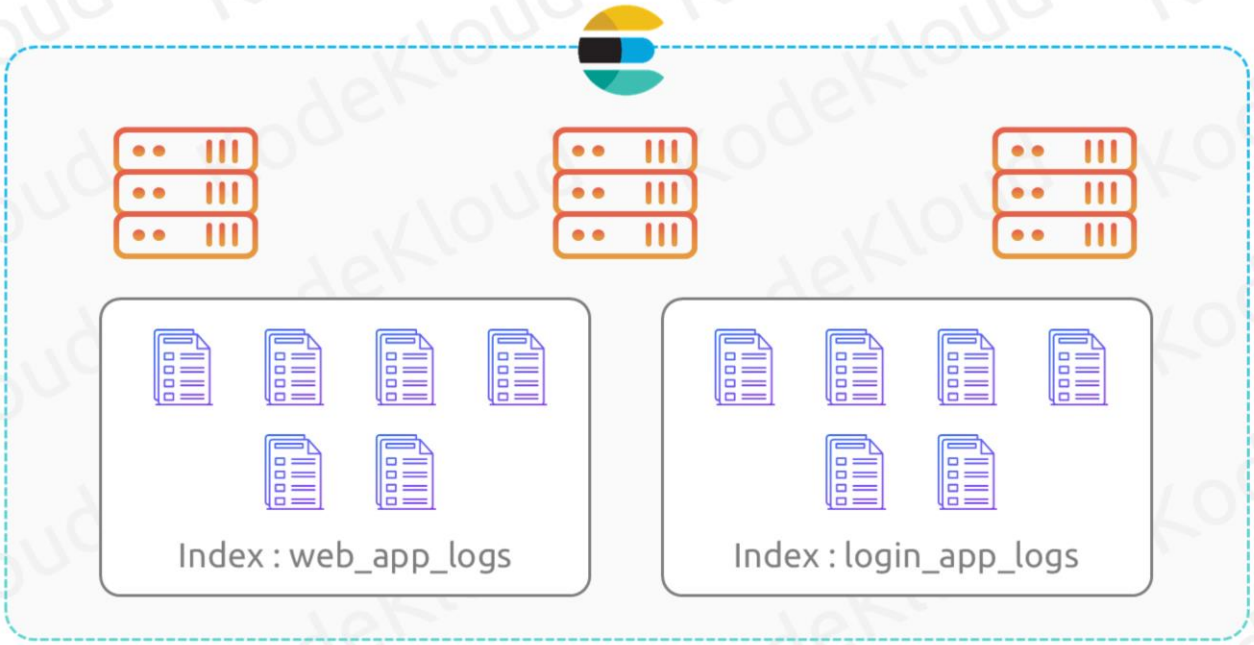
03

Optimized for Search

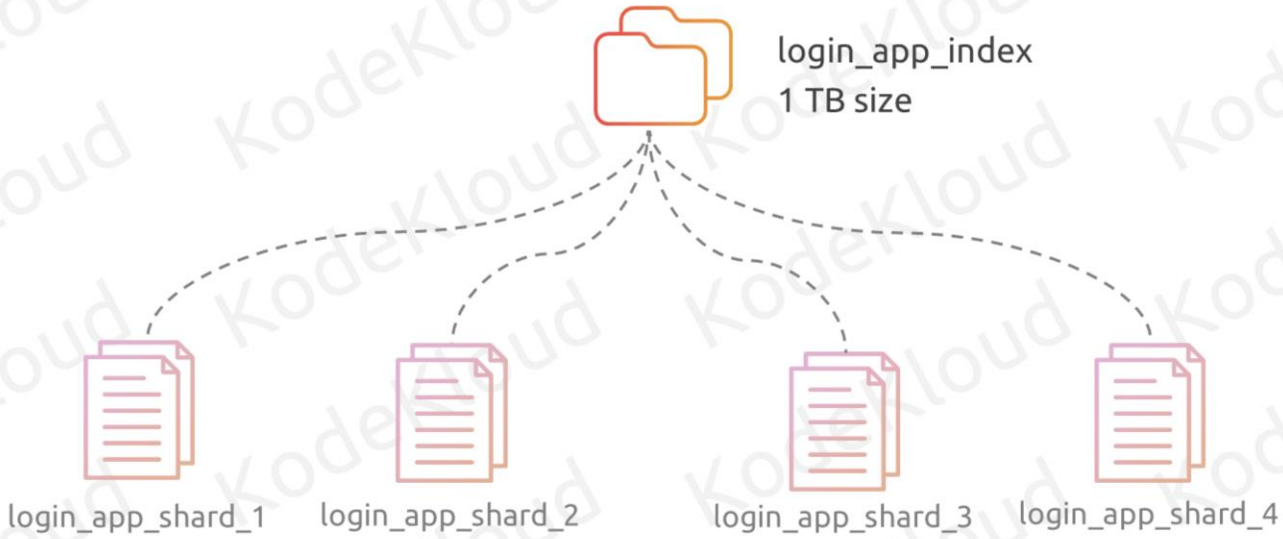
04

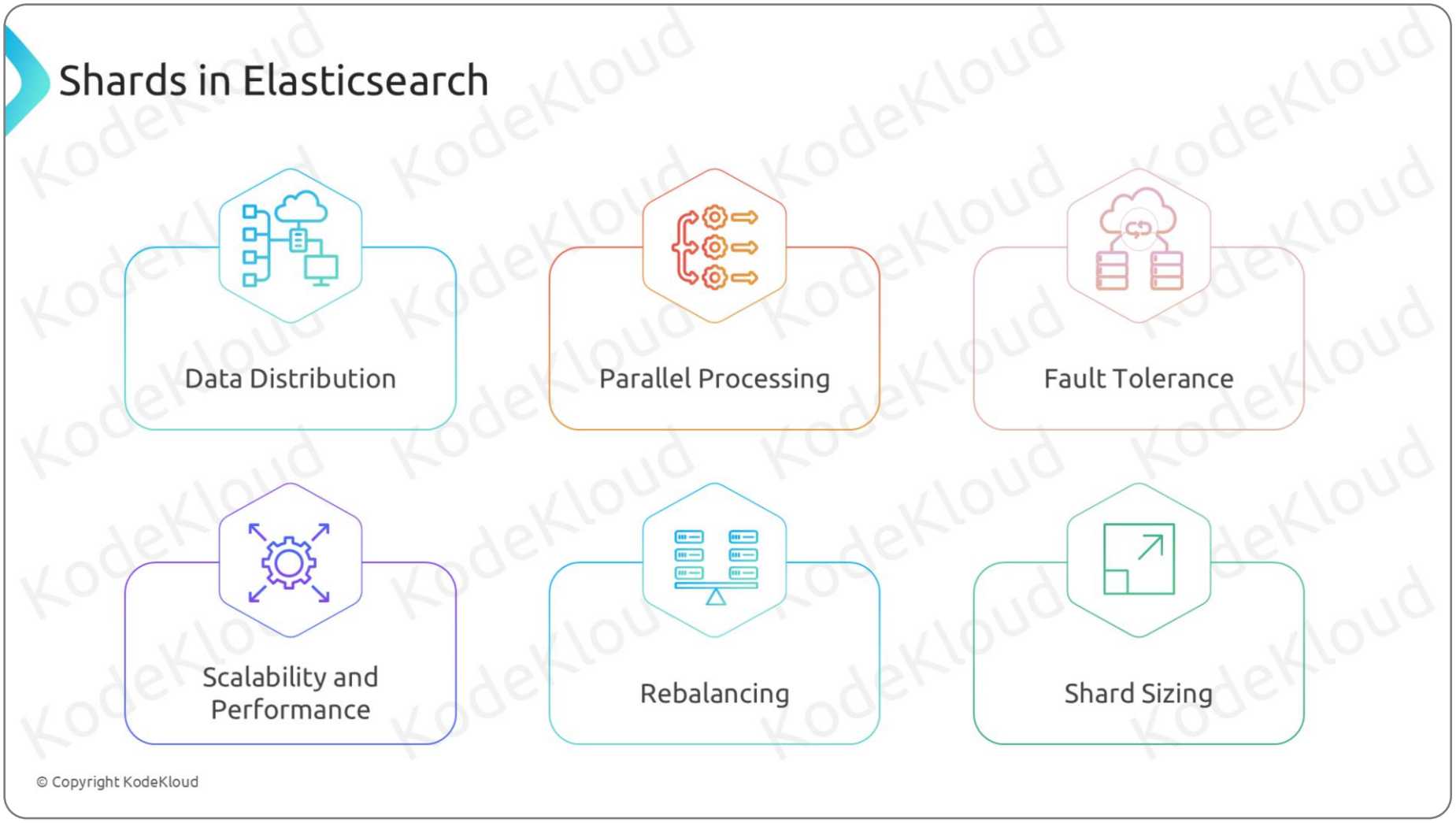
Schema Flexibility

Shards and Replicas in Elasticsearch



Shards and Replicas in Elasticsearch





Data Distribution: Shards are the fundamental building blocks of an Elasticsearch index. They enable horizontal scaling by dividing the index's data into smaller, manageable chunks.

Parallel Processing: Each shard is a self-contained Lucene index, capable of independently performing search and indexing operations. This parallelism boosts performance and allows Elasticsearch to handle large datasets and high query volumes.

Fault Tolerance: Shards can be replicated across multiple nodes for redundancy. If a node fails, the data in its shards can be

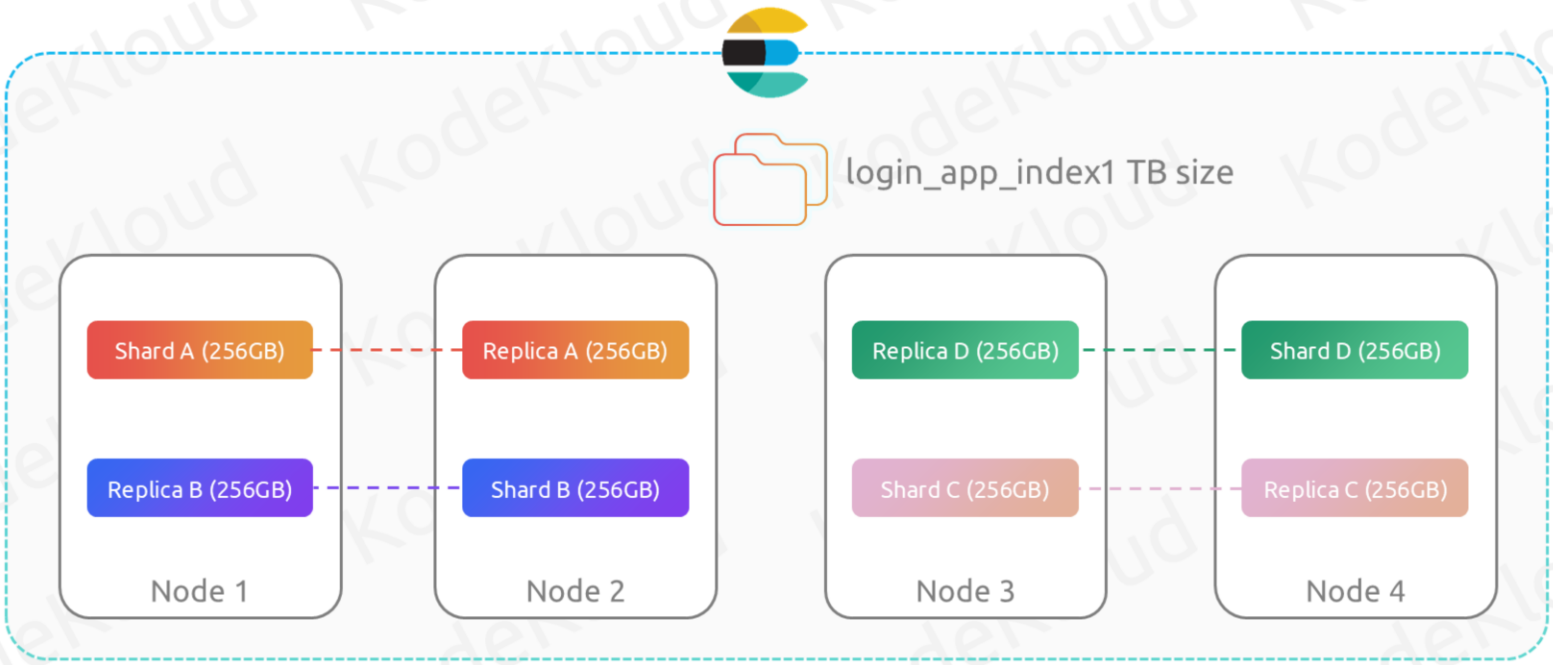
recovered from replicas, ensuring high availability.

Scalability and Performance: By adding more nodes to the cluster, you can increase the number of shards, effectively distributing data and workload, and improving both scalability and performance.

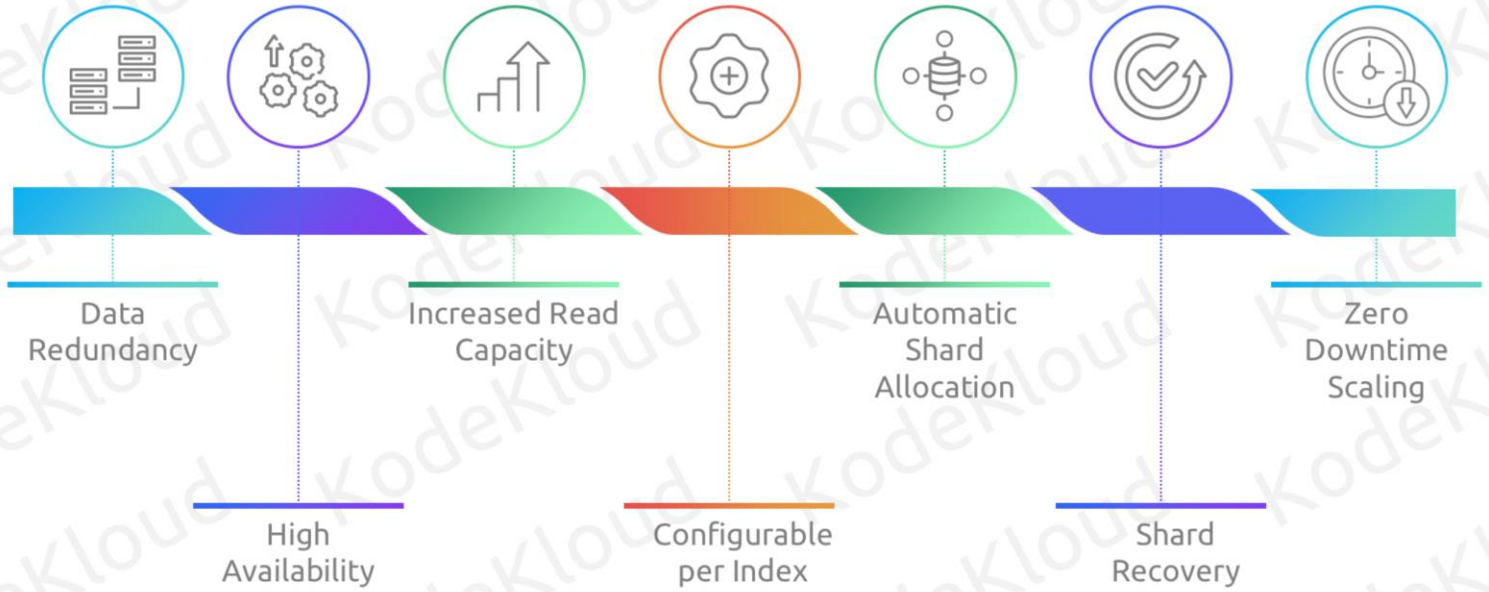
Rebalancing: Elasticsearch automatically rebalances shards across nodes when the cluster configuration changes (e.g., adding or removing nodes). This ensures even data distribution and optimal resource utilization.

Shard Sizing: It's important to choose an appropriate shard size based on your data volume and growth expectations. Too few large shards might hinder performance and scalability, while too many small shards could lead to excessive overhead.

Shards and Replicas in Elasticsearch



Replicas in Elasticsearch



Data Redundancy: Replicas create copies of primary shards, ensuring data availability in case of failures.One-liner: "Replicas: Your data's safety net in Elasticsearch."

High Availability: Replicas allow Elasticsearch to automatically promote a replica to primary if the original fails.One-liner: "Replicas: The unsung heroes of Elasticsearch uptime."

Increased Read Capacity: Replicas can serve search requests, improving performance under heavy read loads.One-liner:

"Replicas: Supercharging your Elasticsearch search speed."

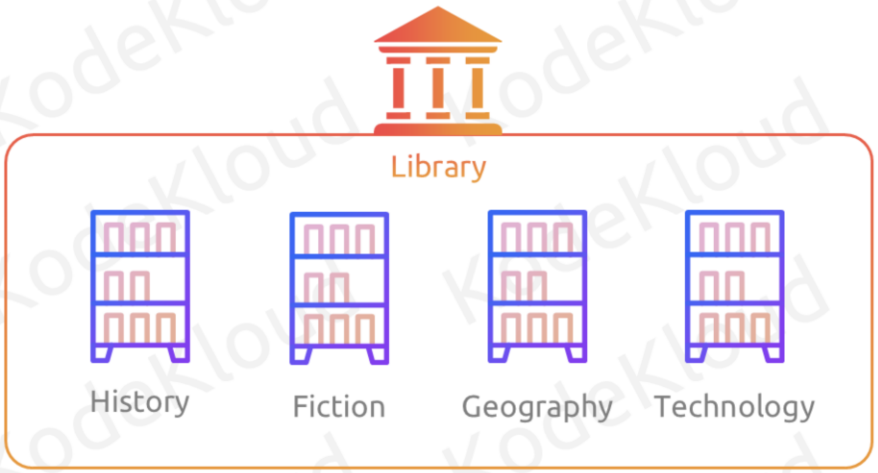
Configurable Per Index: You can customize the number of replicas for each index to match its specific needs.One-liner: "Replicas: Tailor-made data protection for each Elasticsearch index."

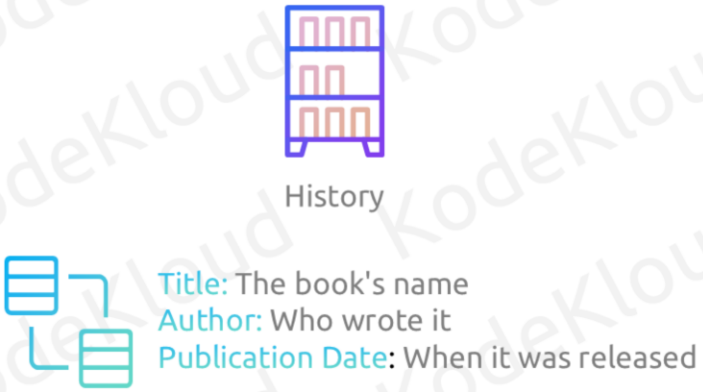
Automatic Shard Allocation: Elasticsearch intelligently manages replica placement across nodes for optimal resilience.One-liner: "Replicas: Elasticsearch's smart data distribution strategy."

Shard Recovery: If a node fails, replicas help to quickly restore data to new shards on healthy nodes.One-liner: "Replicas: The first responders of Elasticsearch data recovery."

Zero Downtime Scaling: Adding replicas allows you to scale read capacity without interrupting ongoing operations.One-liner: "Replicas: Your ticket to seamless Elasticsearch expansion."

Mapping in Elasticsearch





Mapping in Elasticsearch

Mapping is the process of defining how a document and the fields it contains are stored and indexed.

Dynamic Mapping

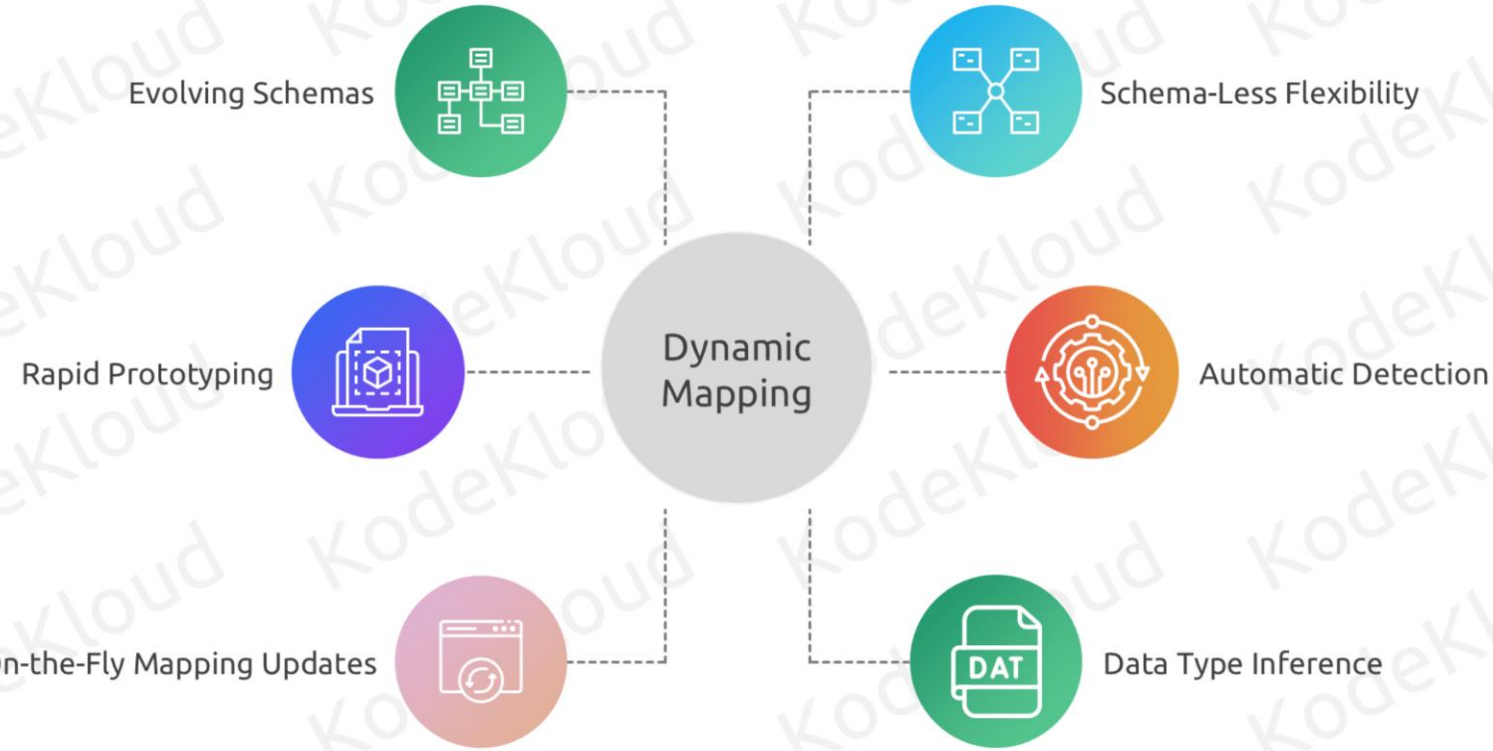
Explicit Mapping

Dynamic Mapping in Elasticsearch

```
{
    "product_id": "12345",
    "name": "Awesome Sneakers",
    "description": "The most comfortable shoes ever!",
    "price": 99.99,
    "in_stock": true,
    "release_date": "2023-12-15"
}
```

```
{
  "mappings": {
    "_doc": {
      "properties": {
        "product_id": { "type": "long" },
        "name": { "type": "text" },
        "description": { "type": "text" },
        "price": { "type": "float" },
        "in_stock": { "type": "boolean" },
        "release_date": { "type": "date" }
      }
    }
  }
}
```

Dynamic Mapping in Elasticsearch

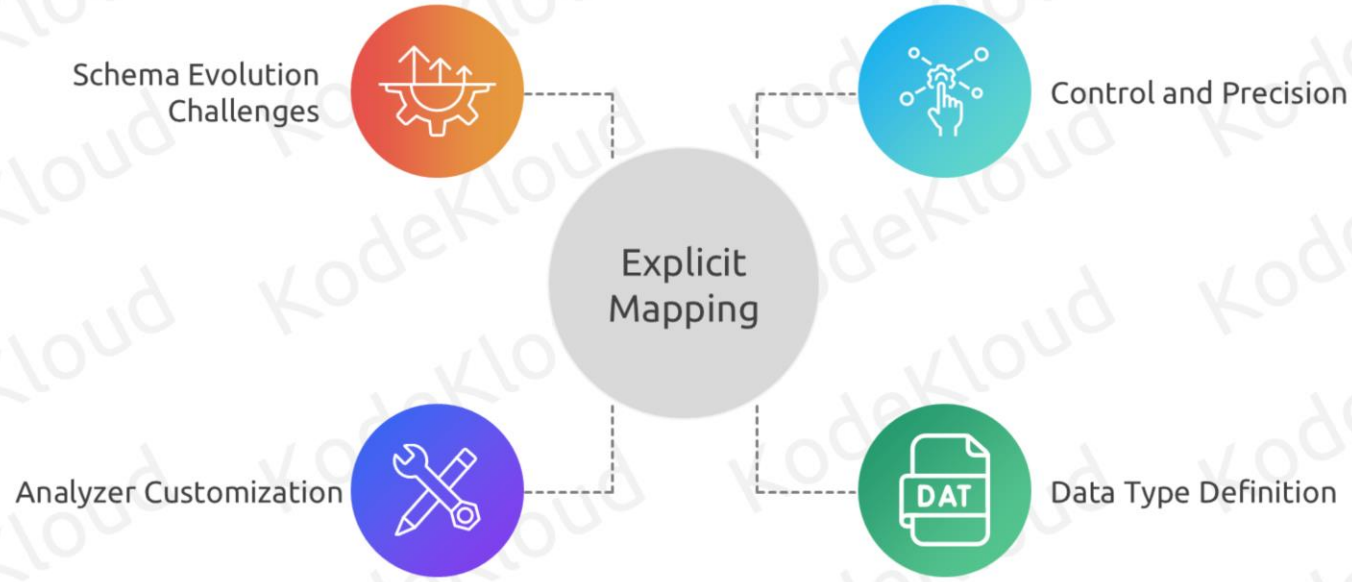


Explicit Mapping in Elasticsearch

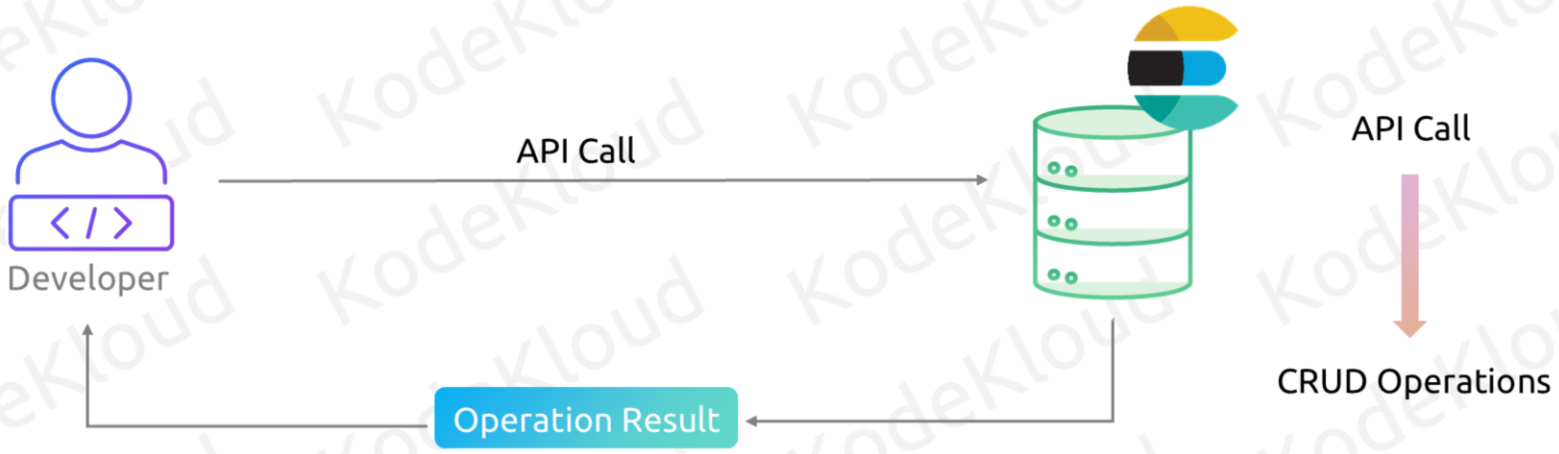
```
PUT /product_index
{
  "mappings": {
    "_doc": {
      "properties": {
        "product_id": { "type": "keyword"
},
        "name": {
          "type": "text",
          "analyzer": "english"
        },
        "description": { "type": "text"
},
        "price": { "type": "float" },
        "in_stock": { "type": "boolean"
},
        "release_date": {
          "type": "date",
          "format": "yyyy-MM-dd"
        }
      }
    }
  }
}
```

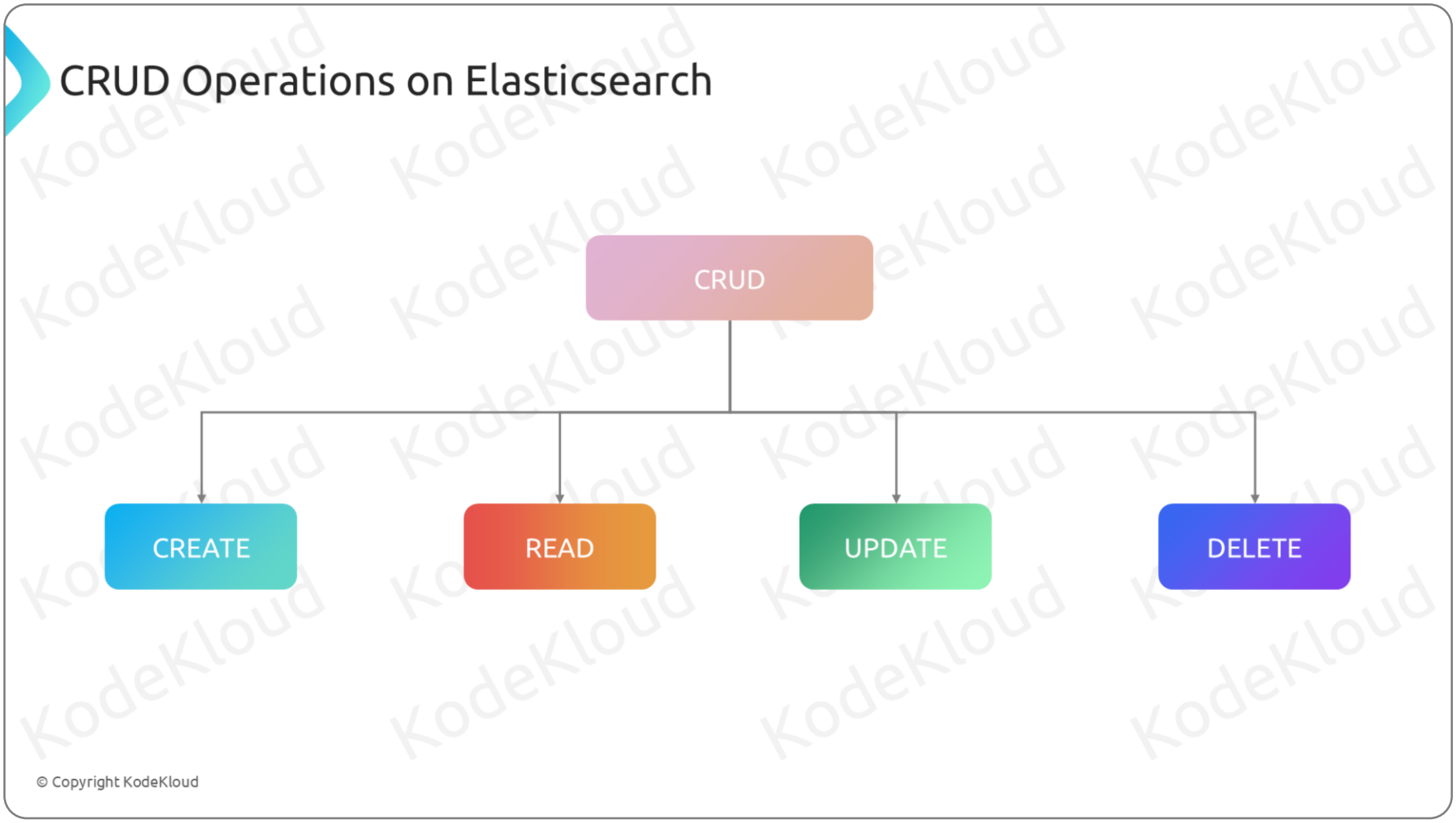
```
{
    "product_id": "12345",
    "name": "Awesome Sneakers",
    "description": "The most comfortable shoes ever!",
    "price": 99.99,
    "in_stock": true,
    "release_date": "2023-12-15"
}
```

Mapping in Elasticsearch



CRUD Operations on Elasticsearch

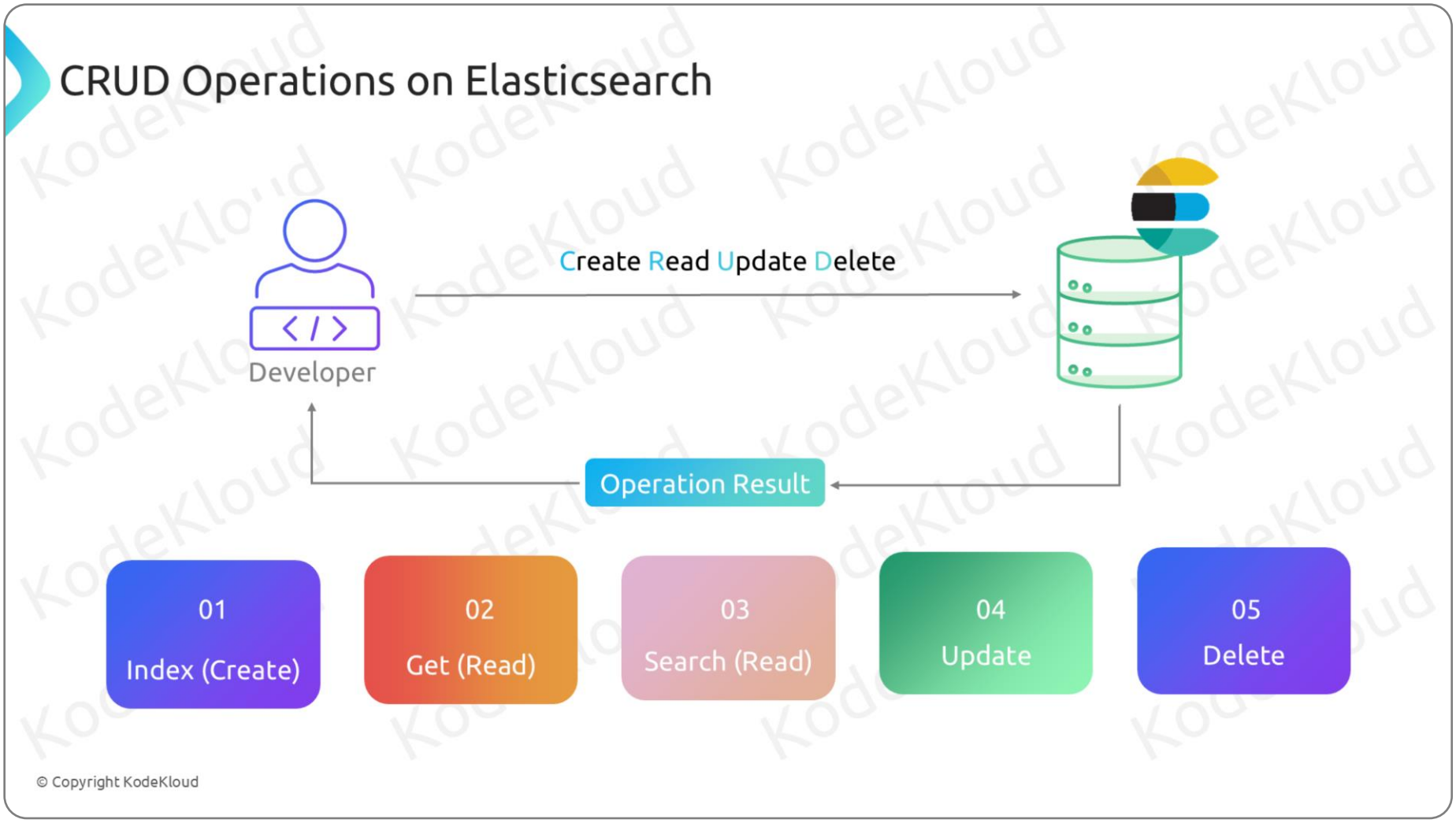




What CRUD Stands For

CRUD is an acronym that describes the four basic operations you can perform on data in a database or search engine like Elasticsearch:

- **C**reate
- **R**ead
- **U**pdate
- **D**elete



Index (Create): Adds new JSON documents to a specific index, like adding books to a library shelf. Each document gets a unique ID for later retrieval.
Get (Read): Retrieves a specific document by its unique ID from the relevant index, similar to finding a book in a library by its call number.
Search (Read): Finds documents based on their content using flexible queries. This is like searching the library catalog by keywords or author.
Update: Modifies existing documents, either by updating specific fields or by changing multiple documents that match a

search query.
Delete: Removes documents from the index either by their unique ID or by matching a query. This is equivalent to removing a book from the library.

Demo

Elasticsearch CRUD commands

Demo

Difference between POST and PUT operations

Demo

Cluster Information – Elasticsearch CRUD commands

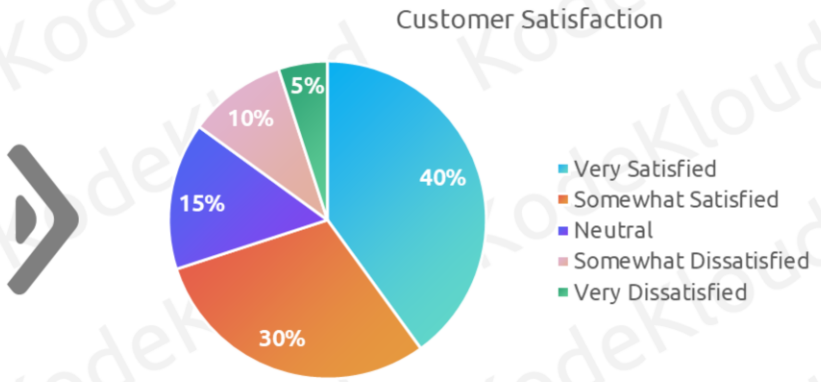
Understanding Kibana

Importance of Data Visualization

Rating	Number of Customers	Percentage
Very Satisfied	80	40%
Somewhat Satisfied	60	30%
Neutral	30	15%
Somewhat Dissatisfied	20	10%
Very Dissatisfied	10	5%
Total	200	100%

Importance of Data Visualization

Rating	Number of Customers	Percentage
Very Satisfied	80	40%
Somewhat Satisfied	60	30%
Neutral	30	15%
Somewhat Dissatisfied	20	10%
Very Dissatisfied	10	5%
Total	200	100%



Importance of Data Visualization

Improved Understanding: Data visualization transforms raw data into intuitive formats like charts, graphs, and maps. This makes complex trends, patterns, and outliers much easier to grasp.
Faster Decision Making: Visualizations provide a quick overview of the data, allowing decision-makers to identify key insights and act upon them swiftly. This is crucial in fast-paced business environments.
Effective Communication: Visuals are a powerful way to communicate findings to both technical and non-technical audiences. They make data more engaging and memorable, aiding in conveying complex information effectively.
Identifying Hidden Patterns: Data visualization can reveal hidden relationships and correlations that might not be apparent

in raw data. This can lead to new discoveries and opportunities.
Data-Driven Storytelling: Visualizations help weave compelling narratives around data, making it easier to communicate the story behind the numbers. This is especially useful for presentations and reports.

Kibana Visualizations and Dashboards

Basic Charts | Time Series Visualizations | Geospatial Visualizations

Table: Organizes data in rows and columns for detailed inspection

Month	Product A Sales	Product B Sales	Website Traffic	Customer Satisfaction (1-10)	Marketing Spend
January	5,000	3,500	20,000	7.5	$500
February	6,200	4,200	25,000	8	$650
March	7,800	5,500	32,000	8.2	$800
April	6,500	4,800	28,000	7.8	$700
May	8,200	6,000	35,000	8.5	$850

Basic Charts:

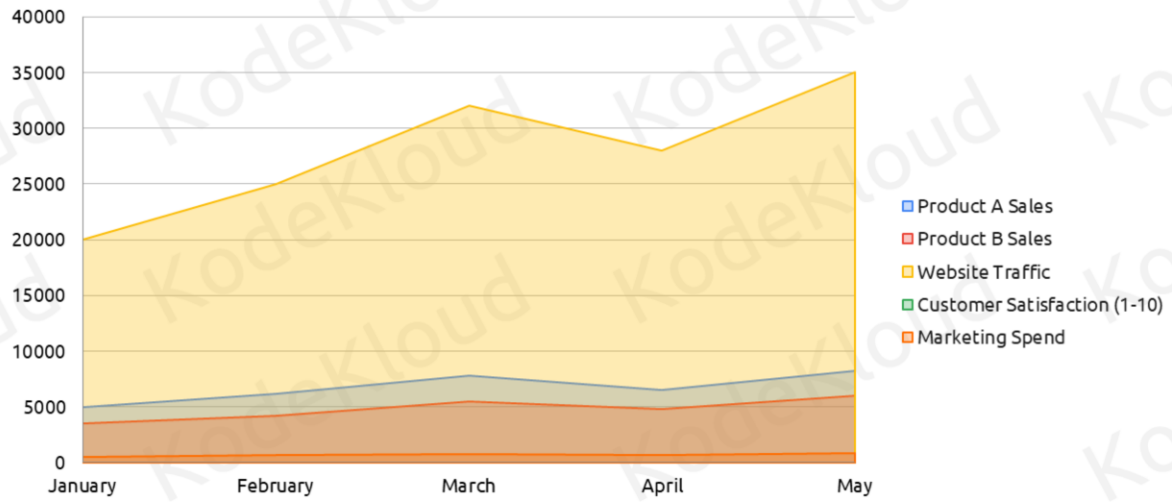
- **Area Chart:** Displays trends over time, with the area between the line and the axis filled.
- **Bar Chart:** Shows comparisons between categories or values.
- **Heatmap:** Visualizes data density across two dimensions using colors.
- **Line Chart:** Tracks changes in values over time.
- **Metric:** Displays a single numeric value.
- **Pie Chart:** Shows proportions of a whole.

- **Scatterplot:** Plots points based on two variables to reveal relationships.
- **Table:** Organizes data in rows and columns for detailed inspection.

Kibana Visualizations and Dashboards

Basic Charts | Time Series Visualizations | Geospatial Visualizations

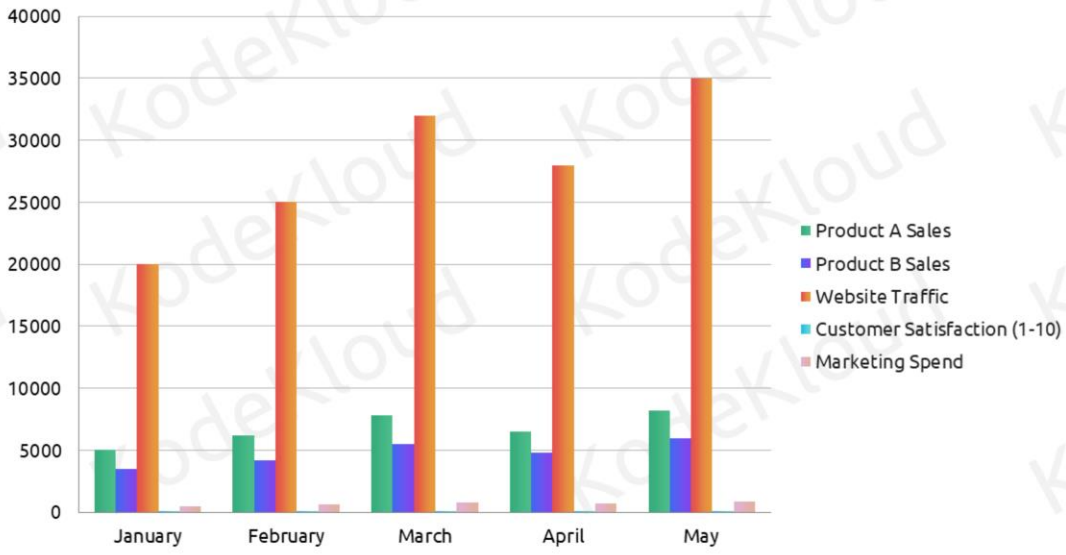
Area Chart: Displays trends over time, with the area between the line and the axis filled



Kibana Visualizations and Dashboards

Basic Charts | Time Series Visualizations | Geospatial Visualizations

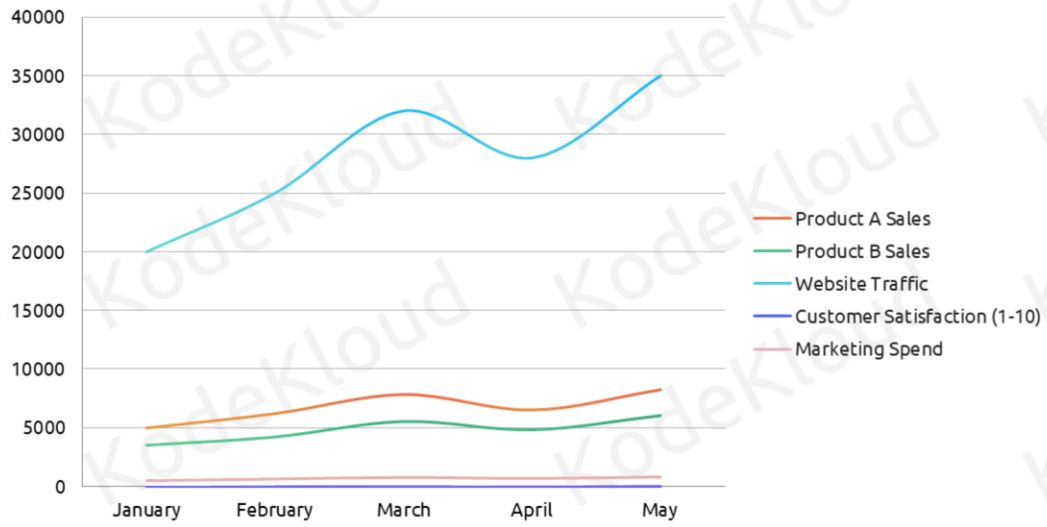
Bar Chart: Shows comparisons between categories or values



Kibana Visualizations and Dashboards

Basic Charts | Time Series Visualizations | Geospatial Visualizations

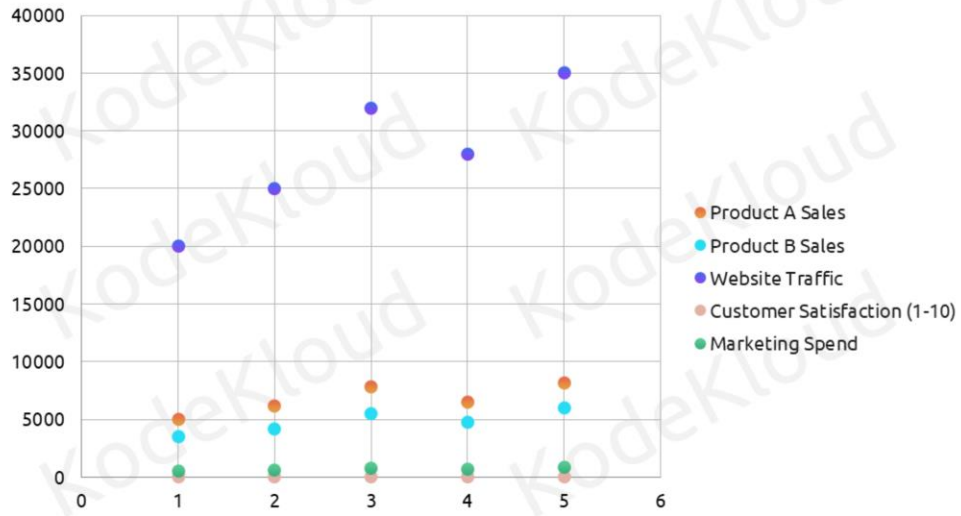
Line Chart: Tracks changes in values over time



Kibana Visualizations and Dashboards

Basic Charts | Time Series Visualizations | Geospatial Visualizations

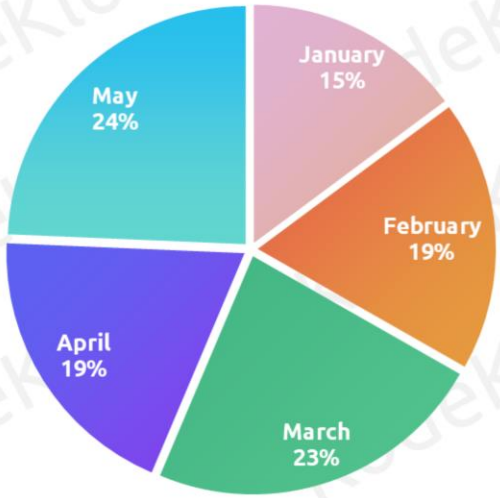
Scatterplot: Plots points based on two variables to reveal relationships: Tracks changes in values over time



Kibana Visualizations and Dashboards

Basic Charts | Time Series Visualizations | Geospatial Visualizations

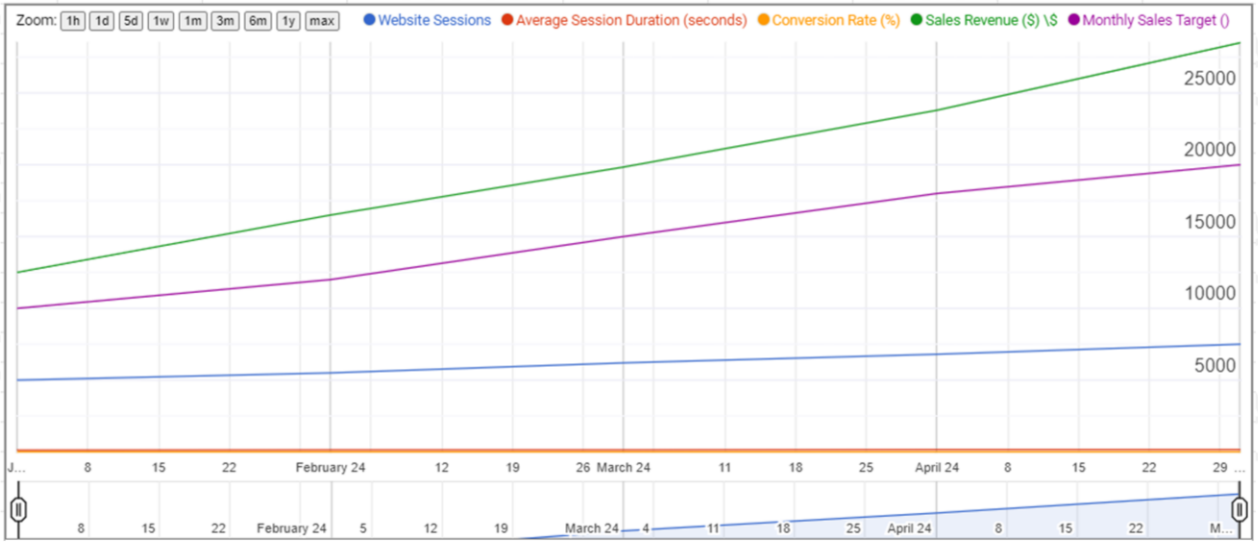
Pie Chart: Shows proportions of a whole



Kibana Visualizations and Dashboards

Basic Charts | Time Series Visualizations | Geospatial Visualizations

Time Series Visual Builder (TSVB): Creates sophisticated visualizations with multiple metrics and complex aggregations over time; ideal for analyzing trends, seasonality, and anomalies



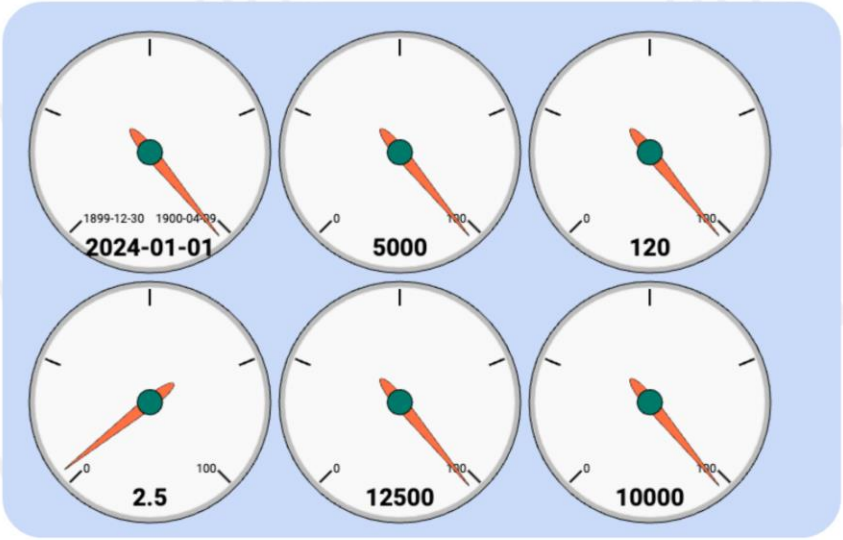
Time Series Visualizations:

- **Time Series Visual Builder (TSVB):** Creates sophisticated visualizations with multiple metrics and complex aggregations over time. Ideal for analyzing trends, seasonality, and anomalies.
- **Gauge:** Displays a single value on a circular scale, often used for performance metrics.
- **Goal:** Tracks progress towards a target value.

Kibana Visualizations and Dashboards

Basic Charts | Time Series Visualizations | Geospatial Visualizations

Gauge: Displays a single value on a circular scale, often used for performance metrics



Time Series Visualizations:

•**Time Series Visual Builder (TSVB):** Creates sophisticated visualizations with multiple metrics and complex aggregations over time. Ideal for analyzing trends, seasonality, and anomalies.

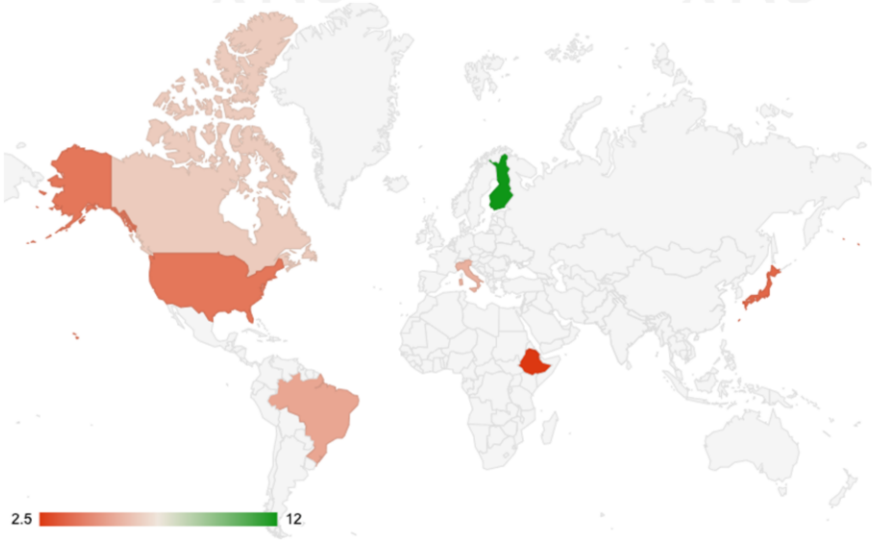
•**Gauge:** Displays a single value on a circular scale, often used for performance metrics.

•**Goal:** Tracks progress towards a target value.

Kibana Visualizations and Dashboards

Basic Charts | Time Series Visualizations | Geospatial Visualizations

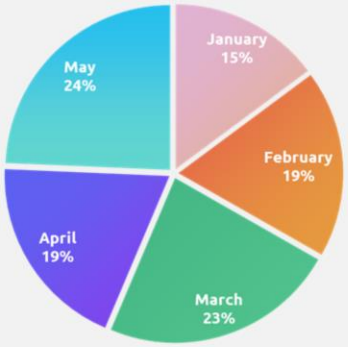
Region Map: Colors geographical regions based on data values

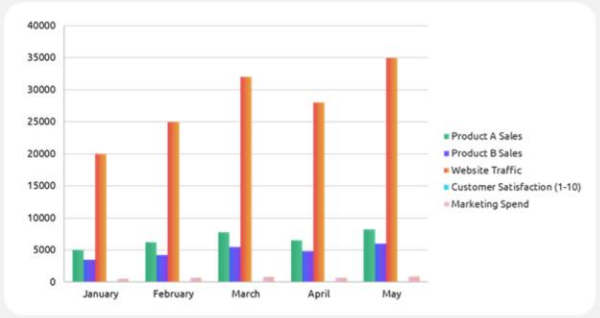


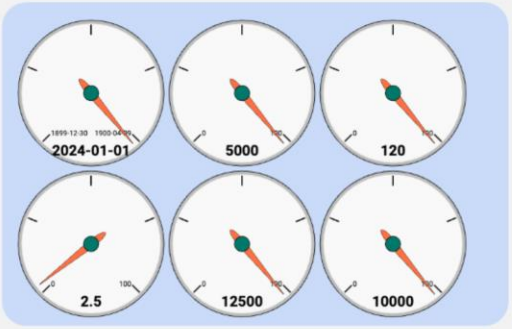
Geospatial Visualizations:

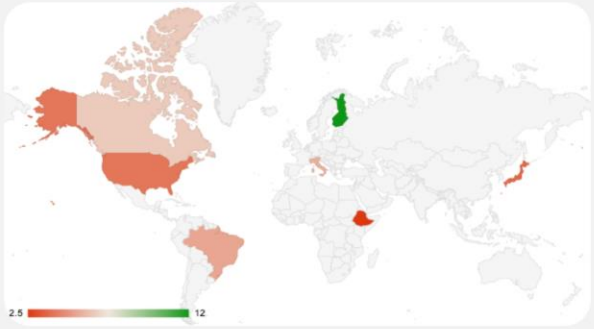
•**Region Map:** Colors geographical regions based on data values.

Kibana Visualizations and Dashboards





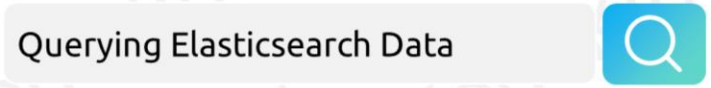




Demo

Building our first dashboard using Kibana

Kibana Query Language (KQL) – Mastering Syntax and Advanced Querying



KQL is integrated into Kibana for querying Elasticsearch data.

KQL queries are translated into Elasticsearch Query DSL requests.

Elasticsearch processes these requests and returns matching data to Kibana.

- Querying Elasticsearch Data
  KQL is a query language built into Kibana, which is a visualization and exploration tool for data stored in Elasticsearch.
- When a user enters a KQL query in Kibana, it translates this query into an Elasticsearch Query DSL (Domain Specific Language) request. This request is then sent to the Elasticsearch cluster.
- Elasticsearch processes the request, searches through the indexed data, and returns the matching documents to Kibana.

Kibana Query Language (KQL) – Mastering Syntax and Advanced Querying

Enter a Query in Kibana:

```
status: "200" AND
extension: "php"
```

Translation to Elasticsearch Query DSL:

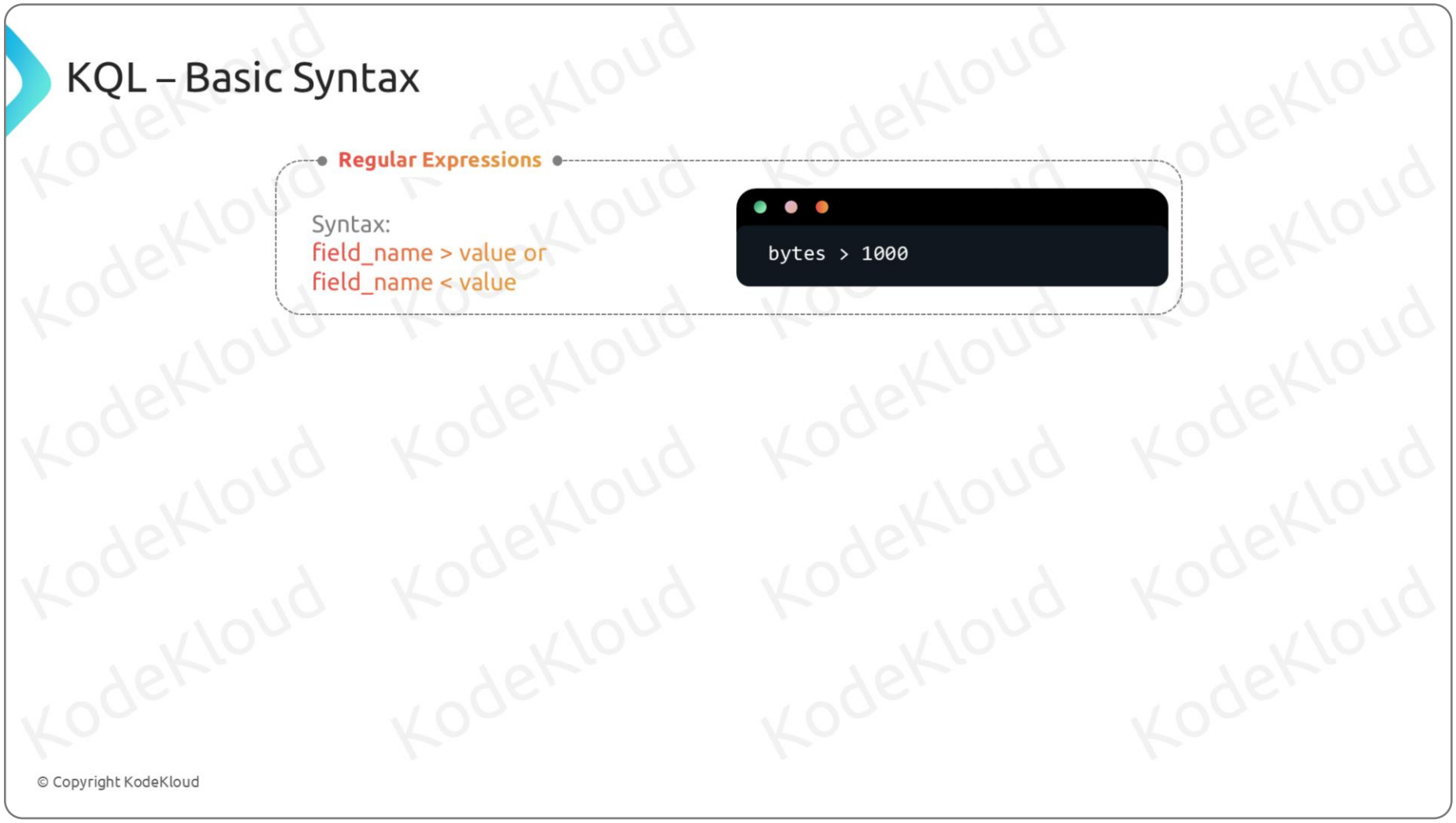
```
{
  "query": {
    "bool": {
      "must": [
        { "match": { "status": "200" }},
        { "match": { "extension": "php" }}
      ]
    }
  }
}
```

KQL – Basic Syntax

1. **Field Queries**:
   - Syntax: `field_name: "value"`
   - Example:
     ```kql
     status: "200"
     ``` This query will return documents where the `status` field has the value `200`.


2. **Wildcard Queries**:
   - Syntax: `field_name: "val*"`
   - Example:
     ```kql
     user: "joh*"
     ``` This query will return documents where the `user` field starts with "joh", such as "john" or "johnny".


3. **Logical Operators**:
   - Syntax: `condition1 AND/OR/NOT condition2`
   - Example:
     ```kql
     status: "404" OR status: "500"
     ``` This query will return documents where the `status` field is either `404` or `500`.




4. **Range Queries**:
   - Syntax: `field_name > value` or `field_name < value`
   - Example:
     ```kql
     bytes > 1000
     ``` This query will return documents where the `bytes` field has a value greater than `1000`.


5. **Existence Queries**:
   - Syntax: `_exists_: field_name`
   - Example:
     ```kql
     _exists_: user_agent
     ``` This query will return documents that contain the `user_agent` field.


### Examples Combined in One Query

Combining different syntax elements, here's a more complex example:

```kql
(status: "200" AND extension: "php") OR (bytes > 1000 AND _exists_: user_agent)
```

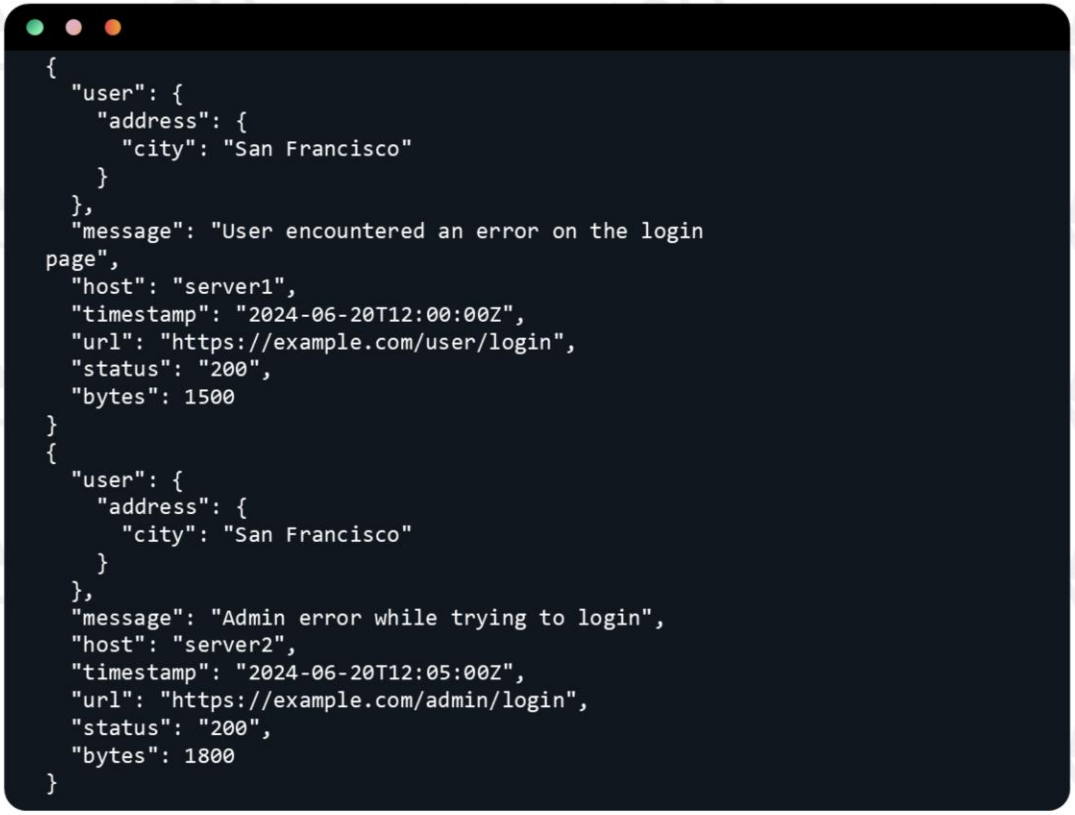
This query will return documents that either have a `status` of `200` and an `extension` of "php", or have more than `1000` bytes and contain the `user_agent` field.

KQL – Basic Syntax

Combined in One Query

```
(status: "200" AND extension: "php") OR
(bytes > 1000 AND _exists_: user_agent)
```

KQL – Advanced Capabilities



```
{
  "user": {
    "address": {
      "city": "San Francisco"
    }
  },
  "message": "User encountered an error on the login
page",
  "host": "server1",
  "timestamp": "2024-06-20T12:00:00Z",
  "url": "https://example.com/user/login",
  "status": "200",
  "bytes": 1500
}
{
  "user": {
    "address": {
      "city": "San Francisco"
    }
  },
  "message": "Admin error while trying to login",
  "host": "server2",
  "timestamp": "2024-06-20T12:05:00Z",
  "url": "https://example.com/admin/login",
  "status": "200",
  "bytes": 1800
}
```

KQL – Advanced Capabilities

Nested Fields

Syntax:
parent_field.child_field: "value"

```
user.address.city: "San Francisco"
```

Lucene Syntax Support

Syntax:
field_name: "value" AND (field_name2: "value2"
OR field_name3: "value3")

```
message: "error" AND (host: "server1" OR host:
"server2")
```

Proximity Searches

Syntax:
"term1 term2"~number

```
"quick brown"~2
```

KQL – Advanced Capabilities

Regular Expressions

Syntax:
field_name: /regex/

```
url: /.*login.*/
```

Boosting

Syntax:
field_name: "value"^number

```
status: "200"^2
```

KQL – Advanced Capabilities

Combined in One Query

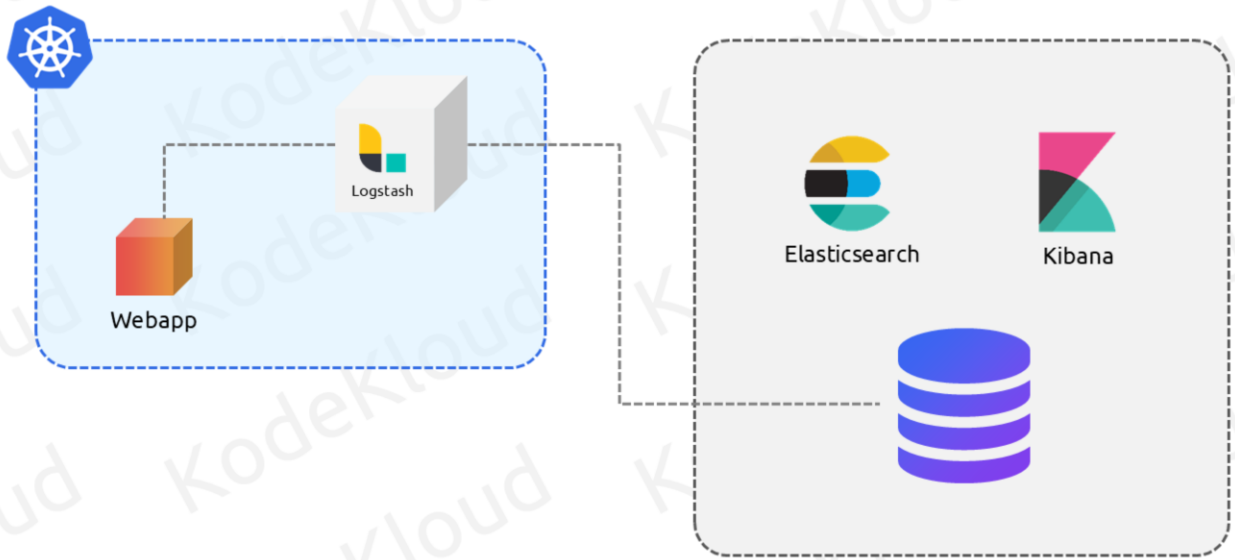
```
(user.address.city: "San Francisco" AND
(message: "error" AND (host: "server1"
OR host: "server2")) AND "quick brown"~2
AND url: /.*login.*/ AND status: "200"^2
```

Demo

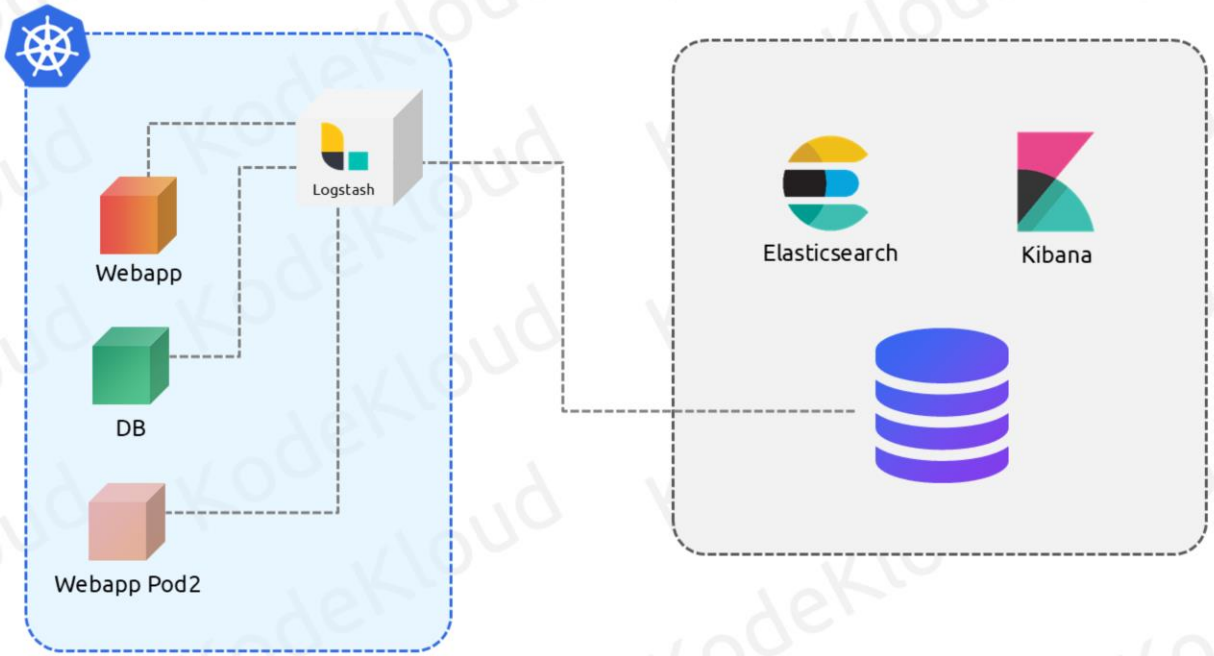
Kibana Query Language (KQL)

FluentBit

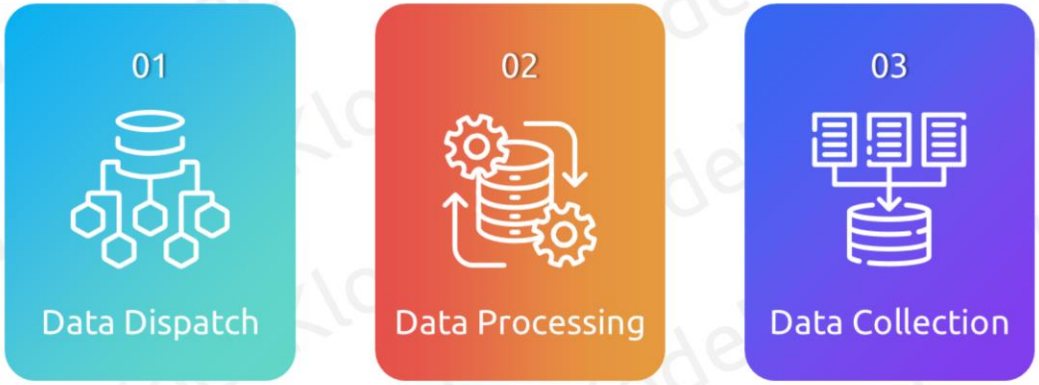
Logstash – Introduction



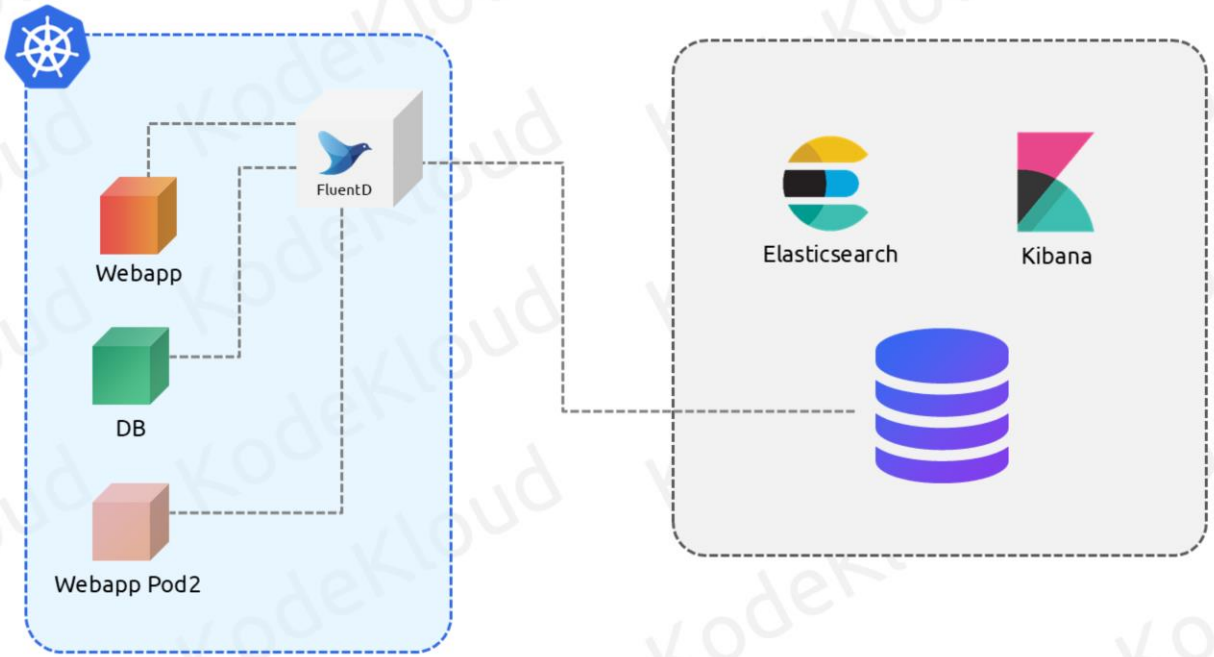
Logstash – Introduction



Logstash's Role Within the ELK Stack



Replacing Logstash With FluentD



Logstash vs FluentD

Features	Logstash	FluentD
Language	Written in JRuby; requires a Java runtime	Written in C and Ruby; no Java dependencies
Ecosystem and Plugins	200 plugins; centralized by Elastic	500 plugins; decentralized in various repositories
Data Transport	No built-in buffering; uses external queues like Redis or Kafka for persistence	Built-in buffering system; reliable without external dependencies
Performance	Consumes more memory than FluentD	More memory-efficient than Logstash
Event Routing	Generally higher due to durability features and multi-AZ deployments	Uses tagging for routing; simpler and more straightforward
Log Parsing	Uses if-then-else statements for routing; more complex configuration	Built-in parsers for formats (JSON, regex, CSV, etc.); less need for external plugins

Logstash
Language: Written in JRuby, requiring a Java runtime on the host machine.
Ecosystem and Plugins: Has around 200 plugins available in a centralized repository managed by Elastic.
Data Transport: Lacks an built-in buffering system, relying on external queues like Redis or Kafka for data persistence across restarts.
Performance: Generally consumes more memory than FluentD.
Event Routing: Uses if-then-else conditional statements for routing events, which can be more complex to configure.
Log Parsing: Relies on plugins for log parsing, requiring additional configuration.

plugins.

FluentD vs FluentBit

Criteria	FluentD	FluentBit
Resource Consumption	Heavier resource usage; requires more memory and CPU; suitable for server environments	Lightweight; minimal resource usage; ideal for edge and embedded systems
Deployment and Usage	More complex setup; requires Ruby runtime; rich feature set and plugins	Easier to deploy with a smaller binary size; quick setup with minimal configuration
Performance	Generally slower due to additional features and Ruby-based implementation	High performance with low-latency data collection and forwarding
Configuration	More complex with multiple files; supports advanced features like routing, buffering, and retry logic	Simpler; uses a single configuration file with straightforward syntax
Data Processing Capabilities	Advanced features for parsing, flexible routing, and data transformation	Basic capabilities for filtering, parsing, and buffering
Use Cases	Ideal for centralized logging, complex log aggregation, and extensive customization	Ideal for edge computing, IoT devices, and scenarios needing lightweight, high-performance log forwarding

FluentD vs Fluent Bit

1.Origin and Development

1. **FluentD**: Developed by Treasure Data, now a part of the Cloud Native Computing Foundation (CNCF).
2. **Fluent Bit**: Also developed by the Fluent community, designed as a lightweight and high-performance log processor and forwarder.

2.Performance

1. **FluentD**: More resource-intensive due to its feature set and design, which can be demanding on memory and

CPU.

2. **Fluent Bit**: Optimized for performance and low resource usage, making it suitable for edge devices and high-volume log processing.

1.Language and Architecture

1. **FluentD**: Written in C and Ruby, leveraging Ruby for its plugin ecosystem and flexibility.
2. **Fluent Bit**: Written in C, designed to be lightweight and efficient, with a minimalistic architecture aimed at high throughput.

2.Plugins and Extensibility

1. **FluentD**: Offers a rich ecosystem of plugins (over 900), covering a wide range of inputs, filters, and outputs, allowing extensive customization.
2. **Fluent Bit**: Has a smaller but growing set of plugins (over 150), focusing on essential functionalities and integration with cloud-native environments.

3.Configuration

1. **FluentD**: Uses a configuration file format based on a JSON-like syntax, which some users find complex, especially for large configurations.
2. **Fluent Bit**: Employs a straightforward configuration syntax, often simpler and easier to manage, making it user-friendly for basic and advanced setups.

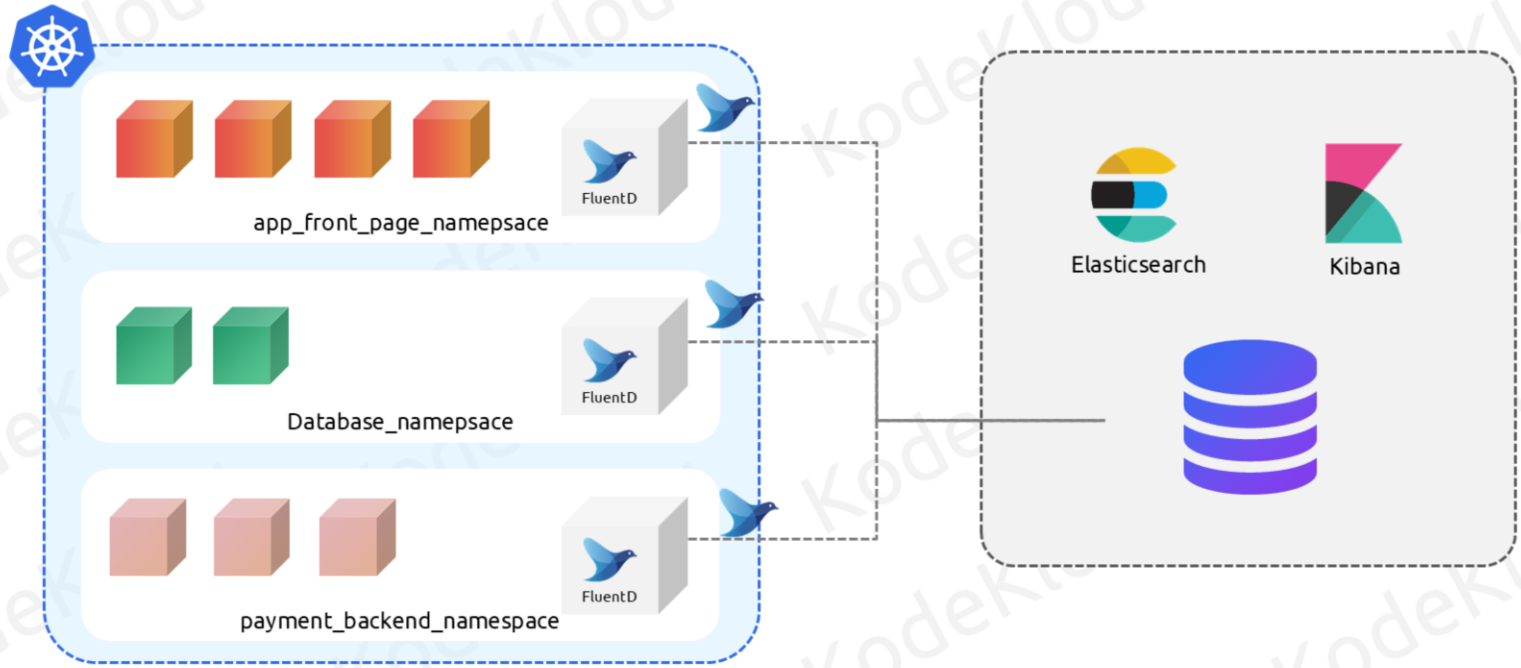
4.Deployment and Use Cases

1. **FluentD**: Typically used in environments where complex log processing and transformation are required, especially within the Elastic ecosystem.
2. **Fluent Bit**: Designed for lightweight log forwarding and collection, ideal for edge computing, IoT, and as a log processor for Kubernetes and cloud-native setups.

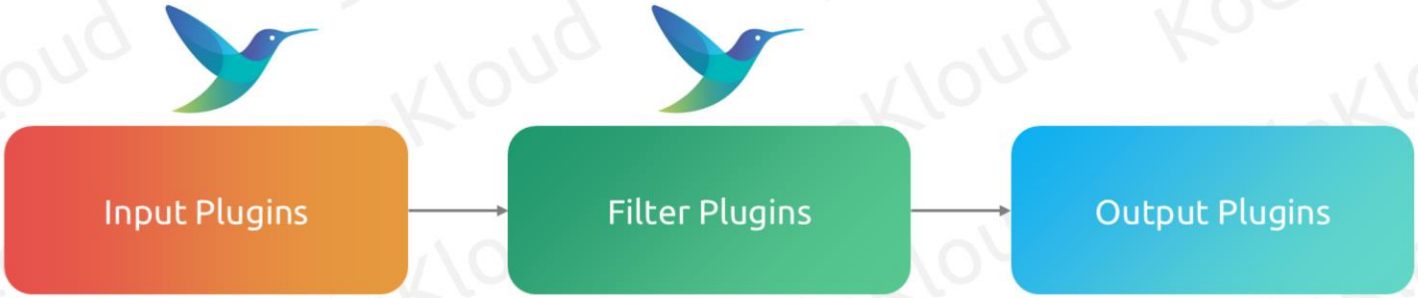
5.Community and Ecosystem

1. **FluentD**: Strong community support with extensive documentation and integration within the Kubernetes ecosystem and other cloud services.
2. **Fluent Bit**: Growing community with a focus on performance and simplicity, gaining traction in the cloud-native space for its efficiency and ease of use.

FluentBit's Role as a Lightweight Log Shipper



Input, Filter, and Output Plugins in FluentBit



Input Plugins

Purpose: Input plugins collect data from various sources.

tail

Reads data from a text file as it is written (tailing a log file).

```
[INPUT]
  Name    tail
  Path    /var/log/nginx/access.log
  Tag     nginx.access
  Parser  nginx
```

systemd

Collects logs from the systemd journal.

```
[INPUT]
  Name    systemd
  Tag     host.*
```

tcp

Listens for TCP connections and reads the data sent to it.

```
[INPUT]
  Name    tcp
  Listen  0.0.0.0
  Port    5170
  Tag     tcp.input
```

Filter Plugins

Purpose: Filter plugins modify, enrich, or drop records based on specific rules.

grep

Filters records by matching a regular expression.

```
[FILTER]
  Name     grep
  Match    nginx.access
  Regex    message  error
```

modify

Adds, removes, or modifies fields in the log records.

```
[FILTER]
  Name     modify
  Match    *
  Add      service nginx
```

parser

Parses specific fields in the logs.

```
[FILTER]
  Name     parser
  Match    nginx.access
  Key_Name   message
  Parser    json
```

Output Plugins

Purpose: Output plugins send the collected and processed data to various destinations.

es

Sends data to Elasticsearch.

```
[OUTPUT]
  Name    es
  Match   *
  Host    127.0.0.1
  Port    9200
  Index   fluentbit
  Type    _doc
```

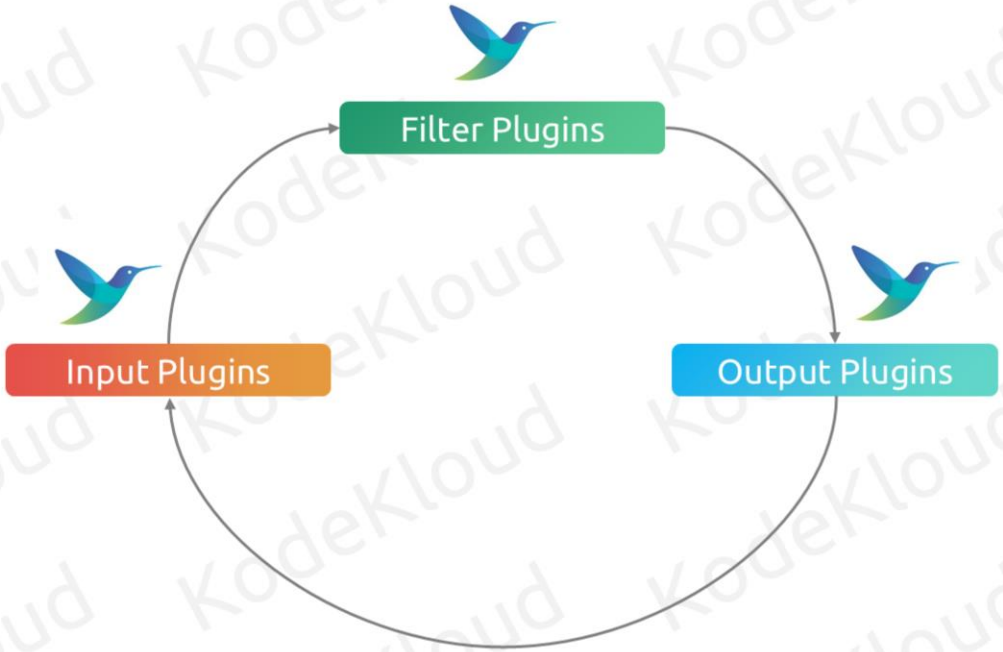
http

Sends data to an HTTP endpoint.

```
[OUTPUT]
  Name    http
  Match   *
  Host    example.com
  Port    80
  URI     /data
  Format  json
```

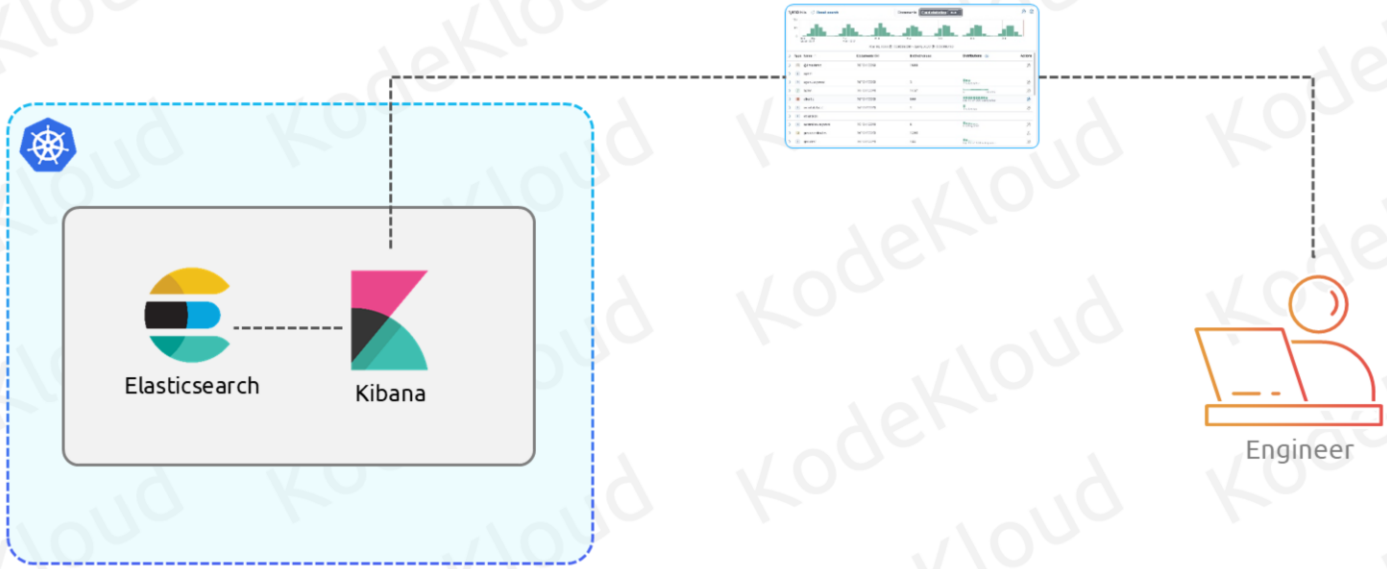
Putting It All Together

```
# Input section
[INPUT]
    Name        tail
    Path        /var/log/nginx/access.log
    Tag         nginx.access
    Parser      nginx

# Filter section
[FILTER]
    Name        grep
    Match       nginx.access
    Regex       message  error

# Output section
[OUTPUT]
    Name        es
    Match       *
    Host        127.0.0.1
    Port        9200
    Index       fluentbit
    Type        _doc
```



Elasticsearch and Kibana Deployment on Kubernetes

Elasticsearch and Kibana on Kubernetes



Demo

Deploy Elasticsearch on Kubernetes

Demo

Deploy Kibana on Kubernetes

Demo

Resource allocation (CPU, memory) for Elasticsearch and Kibana

Security Considerations for Elasticsearch and Kibana on Kubernetes

1. Secure Kubernetes Cluster
2. Enable SSL/TLS Encryption
3. Elastic Stack Security Features
4. Restrict Elasticsearch Access

```
networkPolicy:
  - name: allow-kibana
    podSelector:
      matchLabels:
        app: kibana
    ingress:
      - from:
          - podSelector:
              matchLabels:
                app: kibana
    egress:
      - to:
          - podSelector:
              matchLabels:
                app: elasticsearch
```


Secure Kubernetes Cluster
Remember that the Elasticsearch cluster is only as secure as the underlying Kubernetes cluster, so ensure the Kubernetes environment is properly secured to prevent unauthorized access.

Enable SSL/TLS Encryption
Use SSL/TLS encryption to secure traffic between browsers and Kibana server, preventing eavesdropping and tampering.

Elastic Stack Security Features

ensuring that only necessary users can access sensitive data.

Security Considerations for Elasticsearch and Kibana on Kubernetes

1. Secure Kubernetes Cluster
2. Enable SSL/TLS Encryption
3. Elastic Stack Security Features
4. Restrict Elasticsearch Access

```
# Enable SSL/TLS encryption for
Kibana
kibana:
  server:
    ssl:
      enabled: true
      certificate:
/path/to/cert.pem
      key: /path/to/key.pem
```

Security Considerations for Elasticsearch and Kibana on Kubernetes

1. Secure Kubernetes Cluster
2. Enable SSL/TLS Encryption
3. Elastic Stack Security Features
4. Restrict Elasticsearch Access

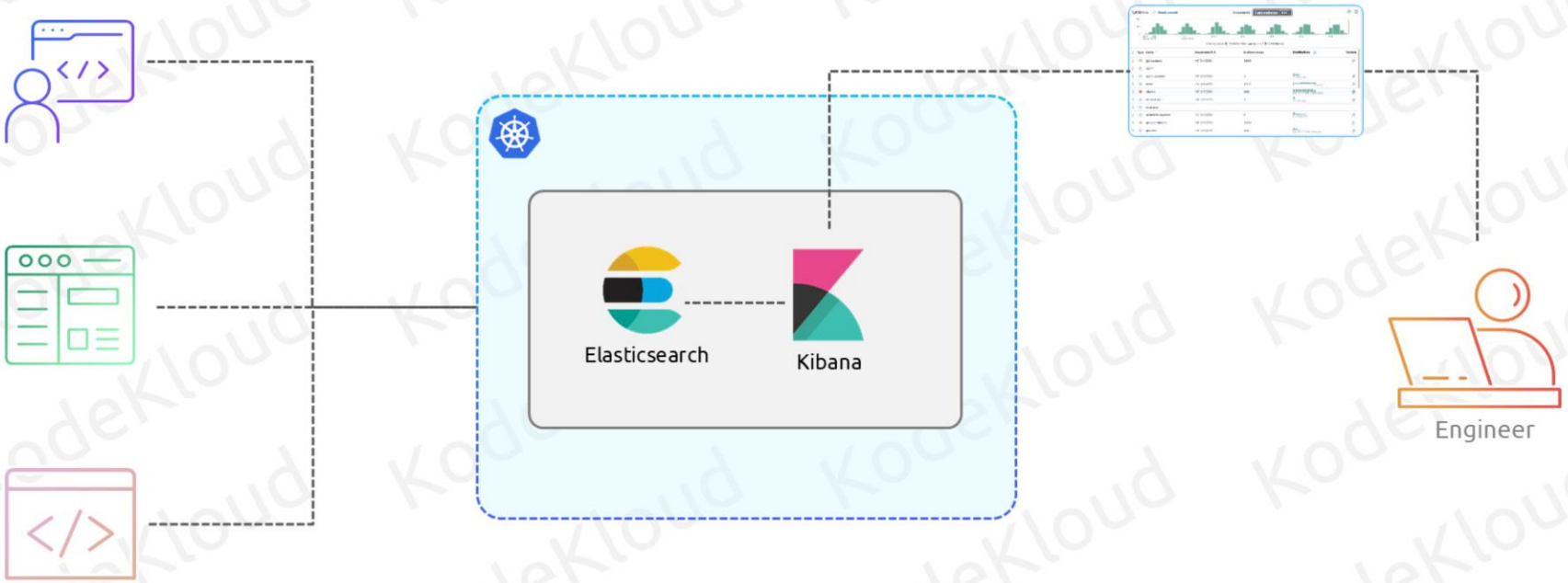
```
# Enable Elastic Stack security
features
elasticsearch:
  security:
    enabled: true
    authc:
      realms:
        - type: basic
          order: 0
    authz:
      roles:
        - type: basic
          order: 0
```

Security Considerations for Elasticsearch and Kibana on Kubernetes

1. Secure Kubernetes Cluster
2. Enable SSL/TLS Encryption
3. Elastic Stack Security Features
4. Restrict Elasticsearch Access

```
# Restrict access to Elasticsearch
by only allowing traffic from
Kibana
elasticsearch:
  network:
    host: kibana
    port: 9200
```

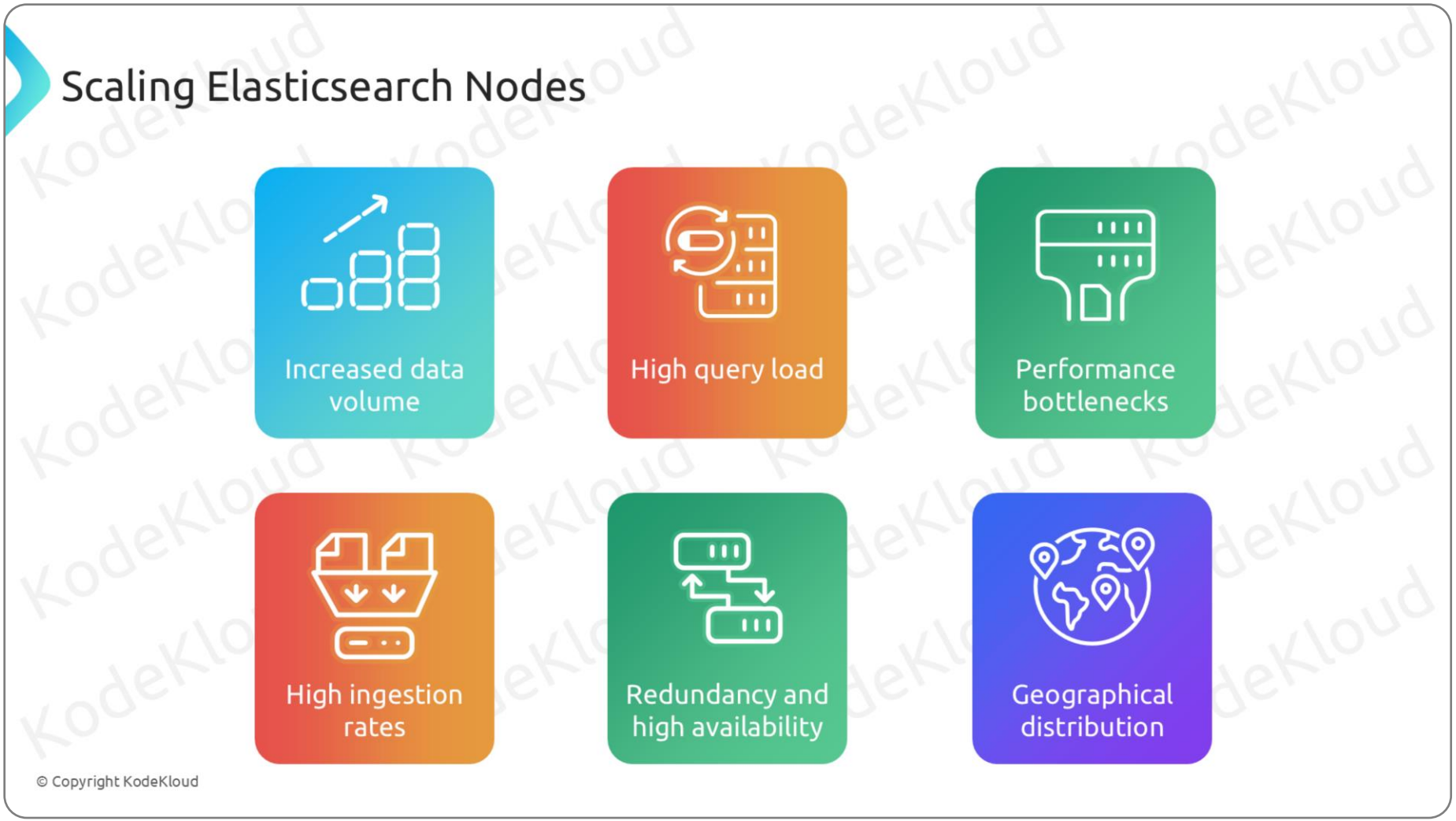
Scaling Elasticsearch Nodes



Scaling Elasticsearch Nodes

When should we consider scaling Elasticsearch nodes?

When Should We Consider Scaling Elasticsearch Nodes?



Increased Data Volume

Explanation: When the amount of stored data grows, more nodes are needed to handle the increased storage requirements efficiently.
High Query Load

Explanation: To maintain performance during high search or query load, adding nodes can distribute the query processing

workload.
Performance Bottlenecks

Explanation: When nodes are frequently experiencing high CPU or memory usage, scaling can alleviate these bottlenecks.
High Ingestion Rates

Explanation: With an increased rate of data ingestion, additional nodes help manage and index the incoming data more effectively.

Redundancy and High Availability

Explanation: To ensure data redundancy and high availability, scaling nodes provides more replicas and fault tolerance.

Geographical Distribution

Explanation: For globally distributed applications, scaling nodes across different regions can reduce latency and improve access speed.

Scaling Elasticsearch Nodes

What are some ways recommended to scale Elasticsearch nodes?

What Are Some Ways Recommended to Scale Elasticsearch Nodes?

Scaling Elasticsearch Nodes

01 Vertical Scaling

02 Horizontal Scaling

03 Node Types Specialization

04 Sharding

05 Index Lifecycle Management (ILM)

06 Snapshot and Restore

Vertical Scaling

Explanation: Increasing the resources (CPU, memory, storage) of existing nodes to handle more data and queries.
Horizontal Scaling

Explanation: Adding more nodes to the cluster to distribute the workload and increase capacity.
Node Types Specialization

Explanation: Using specialized nodes for different roles such as master, data, and ingest nodes to optimize cluster performance.
Sharding

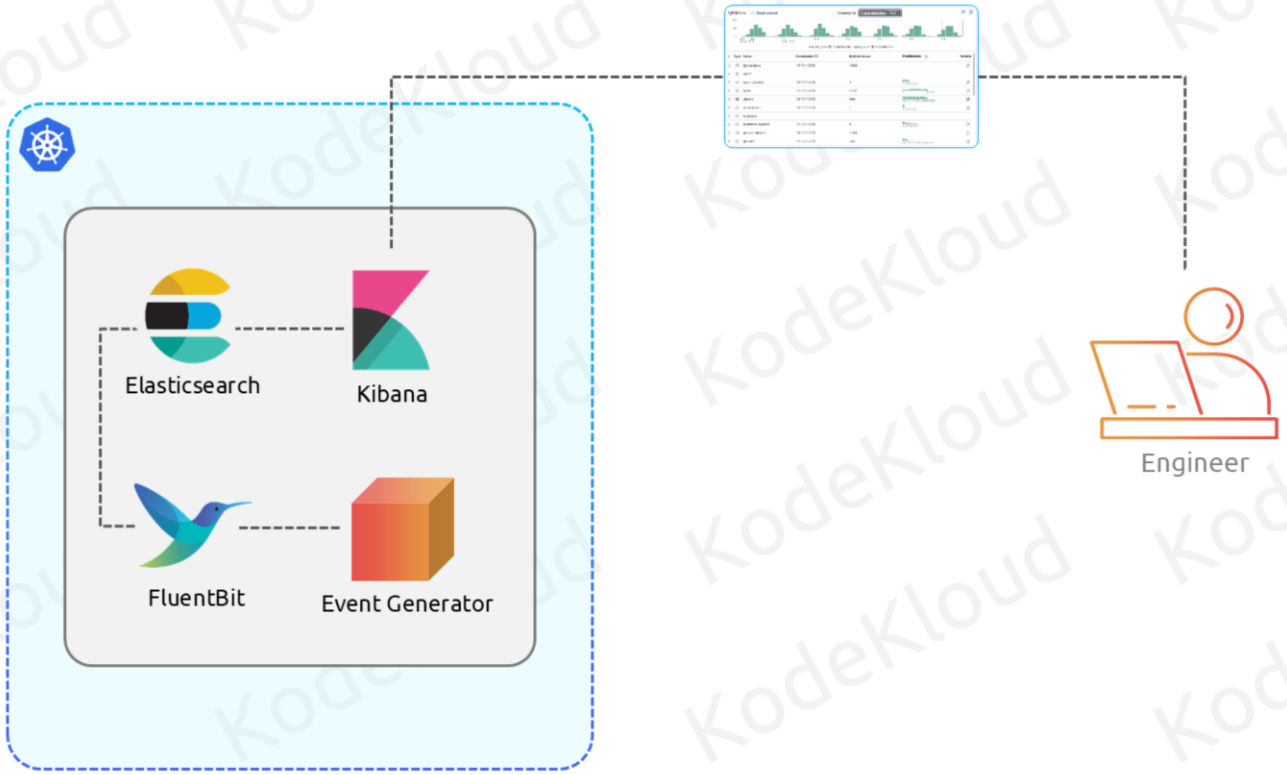
Explanation: Dividing indices into smaller pieces (shards) that can be distributed across multiple nodes to balance the load.
Index Lifecycle Management (ILM)

Explanation: Implementing ILM to automate the management of indices through different phases, reducing the load on nodes.
Snapshot and Restore

Explanation: Using snapshots for backup and disaster recovery, enabling easy restoration and scaling of nodes when needed.
These sections and sub-points can be elaborated into detailed explanations or visualizations on your slide to help your audience understand when and how to scale Elasticsearch nodes effectively.

Deploying Mock Ecommerce Application on K8

Deploying Mock Ecommerce Application and Kibana on K8s



Demo

Setting up Event Generator App

Demo

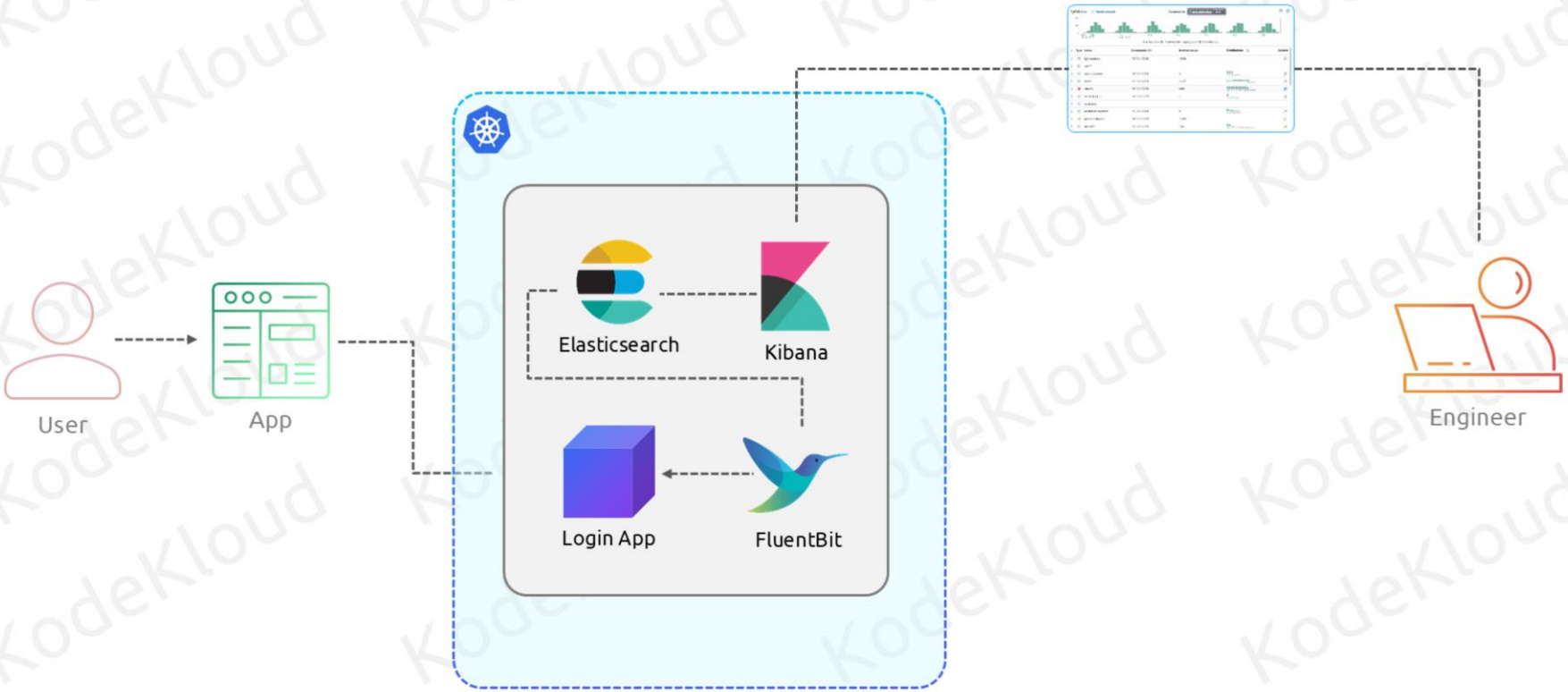
Setting up FluentBit to monitor Application Logs

Demo

Exploring Application Logs in Kibana

Monitoring a Python Application With FluentBit and Elasticsearch

Deploying Login App and Kibana on Kubernetes



Demo

Deploying and validating the login app on Kubernetes

Demo

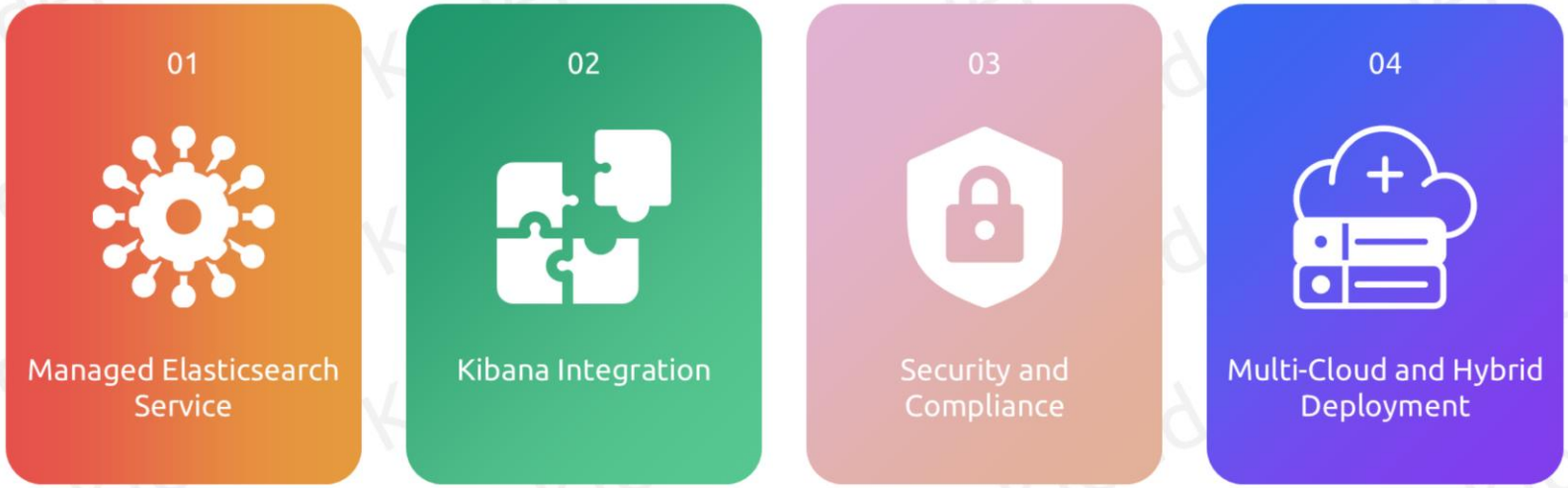
Configuring FluentBit to collect Python Application Logs

Demo

Building Kibana Dashboards to visualize our application

Elastic Cloud

Elastic Cloud – Offerings



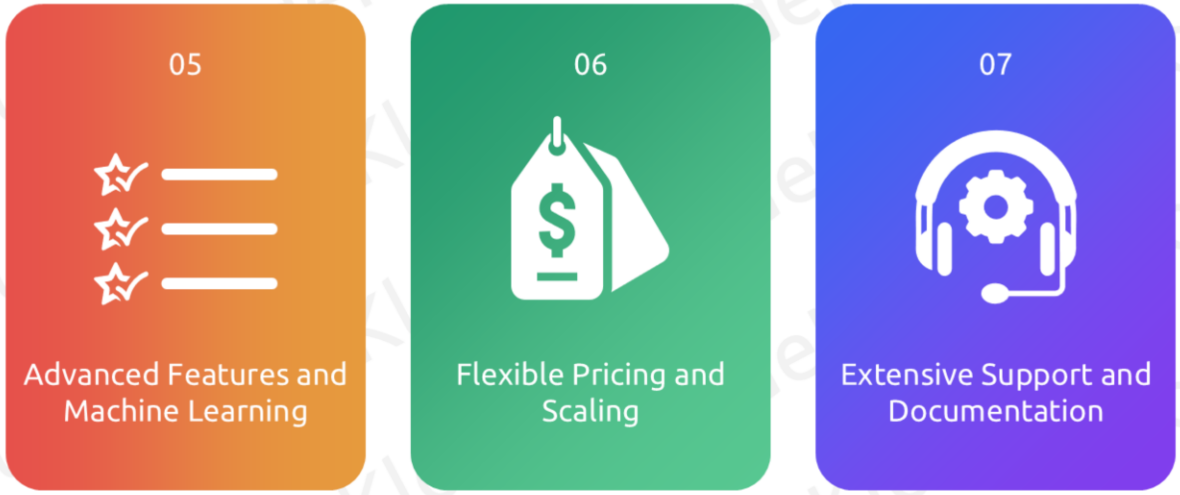
1. Managed Elasticsearch Service
Elastic Cloud offers a fully managed Elasticsearch service, allowing users to deploy, manage, and scale Elasticsearch clusters without worrying about the underlying infrastructure. This service includes automatic updates, backups, and monitoring.
2. Kibana Integration
Elastic Cloud integrates seamlessly with Kibana, providing powerful visualization and analysis tools for Elasticsearch data. Users can create interactive dashboards, perform real-time analysis, and gain insights from their data effortlessly.
3. Security and Compliance

Elastic Cloud provides robust security features, including encryption at rest and in transit, role-based access control, and single sign-on (SSO) capabilities. It also complies with various industry standards and regulations, ensuring data protection and privacy.

4. Multi-Cloud and Hybrid Deployment

Elastic Cloud supports multi-cloud and hybrid deployment options, allowing users to deploy their Elasticsearch clusters on their preferred cloud providers, such as AWS, Azure, and Google Cloud, or on-premises.

Elastic Cloud – Offerings



5. Advanced Features and Machine Learning
Elastic Cloud offers advanced features like machine learning, anomaly detection, and alerting. These features help users to detect unusual patterns, forecast trends, and receive notifications about critical events in their data.

6. Flexible Pricing and Scaling
Elastic Cloud provides flexible pricing plans based on the resources used, making it cost-effective for various use cases. Users can scale their clusters up or down based on their needs, ensuring they only pay for what they use.

7. Extensive Support and Documentation

Elastic Cloud offers extensive support options, including 24/7 technical support, community forums, and comprehensive documentation. Users have access to resources and assistance to ensure their deployments run smoothly and efficiently.

https://cloud.elastic.co/pricing

Demo

Setting up free
Elastic Cloud account

Demo

Monitoring Kubernetes Cluster using ElasticAgent

KodeKloud

Follow us on https://kodekloud.com/ to learn more about us.