Elastic Stack: (ELK) Elasticsearch, Kibana & Logstash

Mục Lục

1. Tổng quan

1.1. Giới thiệu sơ lược

"Elastic Stack" (còn được gọi là ELK Stack) là một bộ công cụ mã nguồn mở được sử dụng để thu thập, lưu trữ, tìm kiếm, phân tích và trực quan hóa dữ liệu theo thời gian thực. ELK là viết tắt của ba thành phần chính:

1. Elasticsearch: Một công cụ tìm kiếm và phân tích phân tán, được thiết kế để xử lý các lượng lớn dữ liệu một cách nhanh chóng và hiệu quả. Nó lưu trữ dữ liệu dưới dạng các tài liệu JSON và cung cấp khả năng tìm kiếm full-text, phân tích dữ liệu và tổng hợp dữ liệu.

2. Logstash: Một công cụ xử lý dữ liệu pipeline, có nhiệm vụ thu thập dữ liệu từ nhiều nguồn khác nhau, chuyển đổi và làm sạch dữ liệu, sau đó gửi dữ liệu đến Elasticsearch hoặc các kho lưu trữ khác. Logstash hỗ trợ nhiều loại đầu vào (input), bộ lọc (filter) và đầu ra (output), giúp xử lý dữ liệu linh hoạt.

3. Kibana: Một công cụ trực quan hóa dữ liệu, cung cấp giao diện người dùng đồ họa để tương tác với dữ liệu được lưu trữ trong Elasticsearch. Kibana cho phép ta tạo các biểu đồ, bảng điều khiển (dashboard) và báo cáo trực quan từ dữ liệu, giúp dễ dàng phân tích và hiểu được các thông tin quan trọng.

Ngoài ELK, Elastic Stack còn có thêm Beats – một tập hợp các agent nhẹ giúp thu thập dữ liệu từ nhiều nguồn khác nhau và gửi về Logstash hoặc Elasticsearch.

1.2. Ứng dụng

- Giám sát và phân tích log: ELK Stack được sử dụng rộng rãi để thu thập và phân tích log từ các hệ thống, ứng dụng và dịch vụ.

- Tìm kiếm và phân tích dữ liệu: Elasticsearch cung cấp khả năng tìm kiếm nhanh chóng và mạnh mẽ trên các tập dữ liệu lớn.

- Phân tích dữ liệu thời gian thực: Kibana cho phép ta theo dõi và phân tích dữ liệu theo thời gian thực thông qua các biểu đồ và bảng điều khiển.

- Bảo mật và phân tích an ninh: Elastic Stack cũng được sử dụng trong các giải pháp bảo mật để phát hiện và phân tích các mối đe dọa.

2. Kiến Trúc Tổng Quan Của Elastic Stack

Elastic Stack (hay ELK Stack) là sự kết hợp của nhiều thành phần hoạt động cùng nhau để thu thập, xử lý, lưu trữ và trực quan hóa dữ liệu theo thời gian thực. Các thành phần chính gồm Elasticsearch, Logstash, Kibana (và Beats). Hệ thống này được thiết kế theo kiến trúc phân tán, cho phép mở rộng quy mô dễ dàng và xử lý lượng dữ liệu lớn một cách hiệu quả.

2.1. Luồng dữ liệu

Quá trình di chuyển của dữ liệu trong Elastic Stack được xây dựng theo dạng pipeline, đi qua các giai đoạn sau:

Beats → Logstash → Elasticsearch → Kibana

2.1.1. Nguồn Dữ Liệu:

- Các hệ thống và ứng dụng: Dữ liệu log, sự kiện, metric từ server, ứng dụng, thiết bị mạng, cơ sở dữ liệu, v.v.

- Các thiết bị thu thập: Các agent nhẹ (ví dụ: Beats) được cài đặt trực tiếp trên các máy chủ hoặc thiết bị để thu thập dữ liệu.

- Về Beats:

+ Là các agent nhẹ, chuyên dụng để thu thập các loại dữ liệu cụ thể (Filebeat cho file log, Metricbeat cho số liệu metric, Winlogbeat cho log trên Windows, v.v.).

+ Beats có thể gửi dữ liệu **trực tiếp** đến Elasticsearch hoặc thông qua Logstash nếu cần xử lý thêm.

2.1.2. Thu Thập Dữ Liệu (Logstash):

- Đóng vai trò là bộ thu thập và xử lý dữ liệu trung gian.
- Nhận dữ liệu từ nhiều nguồn (bao gồm cả Beats, syslog, file, …).
- Áp dụng các bộ lọc (plugins như grok, mutate, date, …) để làm sạch, chuẩn hóa và chuyển đổi dữ liệu theo định dạng mong muốn.
- Sau khi xử lý, Logstash gửi dữ liệu đã định dạng đến Elasticsearch thông qua plugin output.

2.1.3. Lưu Trữ và Tìm Kiếm (Elasticsearch):

- Là kho lưu trữ dữ liệu chính dưới dạng JSON.
- Dữ liệu được lưu trữ trong các **index** với cấu trúc được xác định bởi **mappings**.
- Hỗ trợ tìm kiếm và truy vấn nhanh chóng nhờ vào khả năng index dữ liệu theo kiểu phân tán (sử dụng các node, shard và replica).

2.1.4. Trực Quan Hóa Dữ Liệu (Kibana):

- Kết nối với Elasticsearch thông qua API để truy vấn dữ liệu.
- Cung cấp giao diện đồ họa, cho phép tạo các dashboard, biểu đồ và báo cáo trực quan.
- Hỗ trợ theo dõi dữ liệu thời gian thực, phân tích xu hướng và phát hiện bất thường.

=> Dữ liệu được tạo ra từ các nguồn khác nhau sẽ được **thu thập** bởi Beats, **xử lý** và **chuyển đổi** nếu cần bởi Logstash, sau đó được **lưu trữ** trong Elasticsearch để cuối cùng **trực quan hóa** thông qua Kibana. Quy trình này cho phép dữ liệu di chuyển một cách liền mạch từ nguồn gốc đến khi hiển thị, đảm bảo thông tin luôn được cập nhật và sẵn sàng cho việc phân tích.

2.2. Mối Quan Hệ và Tương Tác Giữa Các Thành Phần

Các thành phần trong Elastic Stack có mối quan hệ chặt chẽ và tương tác qua lại theo các nguyên tắc sau:

2.2.1. Vai Trò và Chức Năng của Từng Thành Phần:

Thành Phần	Chức Năng	Vai Trò
Beats	Thu thập dữ liệu trực tiếp từ các máy chủ, ứng dụng hoặc thiết bị.	Là "người thu thập" nhẹ, đảm bảo dữ liệu được gửi đi ngay từ nguồn với chi phí tài nguyên thấp.
Logstash	Nhận dữ liệu từ nhiều nguồn, xử lý dữ liệu bằng các bộ lọc, và chuyển dữ liệu đến đích cuối cùng.	Là "trung gian xử lý" dữ liệu, cho phép chuẩn hóa, làm sạch và chuyển đổi dữ liệu trước khi lưu trữ.
Elasticsearch	Lưu trữ, index và cung cấp các API tìm kiếm, phân tích dữ liệu	Là "kho dữ liệu" phân tán, hỗ trợ các thao tác tìm kiếm và truy vấn nhanh chóng.
Kibana	Trực quan hóa dữ liệu từ Elasticsearch, tạo dashboard và báo cáo	Là "giao diện người dùng", giúp người dùng cuối dễ dàng tương tác, phân tích và giám sát dữ liệu

2.2.2. Cách Các Thành Phần Giao Tiếp với Nhau:

a. Giao tiếp qua API và Giao thức:

- Beats: Thường sử dụng giao thức TCP hoặc HTTPS để gửi dữ liệu đến Logstash hoặc trực tiếp đến Elasticsearch. Cấu hình của Beats cho phép xác định endpoint (địa chỉ IP và cổng) của đích nhận dữ liệu.

- Elasticsearch API: Các thành phần như Logstash và Kibana sử dụng RESTful API của Elasticsearch (dựa trên giao thức HTTP và dữ liệu JSON) để gửi, truy vấn và quản lý dữ liệu.

- Logstash Pipeline: Logstash được cấu hình với các pipeline (gồm input, filter, output) để nhận và xử lý dữ liệu. Các plugin input có thể nhận dữ liệu qua các giao thức khác nhau (TCP, UDP, HTTP, …).

- **Kibana → Elasticsearch**: **Kibana không lưu dữ liệu** mà chỉ truy vấn từ Elasticsearch qua **HTTP API**. Khi người dùng thao tác trên giao diện Kibana (tìm kiếm log, tạo dashboard), Kibana sẽ gửi truy vấn đến Elasticsearch và hiển thị kết quả.

b. Cấu hình Pipeline và Luồng Dữ Liệu:

- **Elasticsearch**: Elasticsearch nhận dữ liệu từ Logstash và lưu trữ trong **index**. Mỗi index chứa nhiều document, được tổ chức theo các schema (mappings). Khi Kibana truy vấn, Elasticsearch trả về dữ liệu đã được index.

- Logstash: Xác định rõ ràng các bước xử lý (input, filter, output) để đảm bảo dữ liệu được chuyển đổi đúng định dạng và chất lượng trước khi lưu trữ.

- Index Patterns trong Kibana: Kibana cần có cấu hình index patterns để biết cách truy xuất dữ liệu từ Elasticsearch, từ đó hiển thị dữ liệu theo các khía cạnh mong muốn.

c. Tích Hợp và Đồng Bộ Hóa:

- Các thành phần cần được cấu hình sao cho đồng bộ về mặt định dạng dữ liệu và thời gian (ví dụ: định dạng timestamp) để đảm bảo quá trình phân tích và trực quan hóa được chính xác.

- Sự tương tác liên tục giữa các thành phần thông qua các API và cấu hình đồng bộ giúp Elastic Stack luôn có dữ liệu mới nhất, cho phép theo dõi thời gian thực và phản ứng kịp thời với các sự kiện.

=> Mỗi thành phần trong Elastic Stack đều đảm nhận một vai trò cụ thể và quan trọng trong chu trình xử lý dữ liệu. Sự kết hợp của Beats, Logstash, Elasticsearch và Kibana tạo thành một hệ thống liền mạch, trong đó dữ liệu được thu thập, xử lý, lưu trữ và trực quan hóa một cách hiệu quả. Các thành phần này giao tiếp với nhau thông qua các API, giao thức truyền tải dữ liệu (như HTTP, TCP) và cấu hình pipeline được thiết lập rõ ràng, giúp duy trì sự nhất quán và đồng bộ của toàn bộ hệ thống.

3. Elasticsearch – Lý Thuyết Và Cơ Chế Hoạt Động

Elasticsearch là một công cụ tìm kiếm và phân tích dữ liệu phân tán, được xây dựng trên nền tảng **Apache Lucene**. Nó thường được sử dụng để lưu trữ và tìm kiếm dữ liệu thời gian thực với tốc độ cao.

3.1. Kiến Trúc Phân Tán

Elasticsearch được thiết kế dựa trên kiến trúc phân tán để xử lý lượng lớn dữ liệu và đảm bảo tính sẵn sàng cao (high availability). Dưới đây là các thành phần chính:

3.1.1. Cách xây dựng cluster

- Cluster: Là một tập hợp các máy chủ (node) làm việc cùng nhau để lưu trữ dữ liệu và thực hiện các thao tác tìm kiếm, phân tích. Mỗi cluster có một tên duy nhất để phân biệt.

- Node: Một máy chủ vật lý hoặc ảo tham gia vào cluster. Mỗi node có một vai trò cụ thể:

+ Master Node:

- Quản lý các hoạt động của cluster như tạo/xóa index, theo dõi trạng thái các node, phân bổ shard.
- Chỉ một master node hoạt động tại một thời điểm (tránh split-brain bằng thuật toán quorum).
- Cấu hình: node.roles: [master].

+ Data Node:

- Lưu trữ dữ liệu (shard) và thực hiện các thao tác tìm kiếm, phân tích.
- Cấu hình: node.roles: [data].

+ Coordinating Node:

- Điều phối các yêu cầu từ client, chuyển chúng đến data node phù hợp.
- Mặc định, tất cả node đều có thể đóng vai trò này.

+ Ingest Node:

- Xử lý dữ liệu trước khi lưu trữ (như chuyển đổi định dạng, thêm trường).
- Cấu hình: node.roles: [ingest].

3.1.2. Sharding (Phân mảnh) và Replication (Sao lưu):

- Shard:

+ Một index được chia thành các phân mảnh (shard) để phân tán dữ liệu và tăng hiệu suất.

+ Shard có thể là **primary shard** (lưu dữ liệu gốc) hoặc **replica shard** (bản sao của primary shard).

+ Số lượng primary shard được xác định khi tạo index và **không thể thay đổi** sau đó.

- Replica:

+ Là bản sao của primary shard, giúp đảm bảo tính sẵn sàng cao và cân bằng tải.

+ Số lượng replica có thể thay đổi động.

+ Ví dụ: Nếu một node chết, replica shard trên node khác sẽ được promote thành primary shard.

3.2. Lưu Trữ Và Quản Lý Dữ Liệu

3.2.1. Index, Document, và Mapping

- **Index**: Tương tự một "database" trong hệ thống cơ sở dữ liệu, dùng để nhóm các document liên quan. Ví dụ: Index users lưu thông tin người dùng, index products lưu thông tin sản phẩm.

- **Document**: Đơn vị dữ liệu cơ bản trong Elasticsearch, được lưu dưới dạng JSON. Mỗi document thuộc một index và có một ID duy nhất. Ví dụ:

```
{
  "id": "1",
  "name": "John Doe",
  "email": "john@example.com"
}
```

- **Mapping**: Định nghĩa cấu trúc dữ liệu của index, bao gồm kiểu dữ liệu các trường (text, keyword, date, v.v.) và cách xử lý chúng. Elasticsearch tự động tạo mapping nếu không được khai báo trước (**dynamic mapping**). Ví dụ mapping tùy chỉnh:

```
PUT /users
{
  "mappings": {
    "properties": {
      "name": { "type": "text" },
      "email": { "type": "keyword" }
    }
  }
}
```

3.2.2. Cơ chế lưu trữ dữ liệu theo định dạng JSON

Tất cả dữ liệu trong Elasticsearch đều được lưu trữ dưới dạng JSON. Khi document được thêm vào index, Elasticsearch:

1. **Phân tích (Analyze)**: Áp dụng các bộ phân tích (analyzers) để tách văn bản thành các token (ví dụ: tách từ, loại bỏ stopwords).
2. **Lập chỉ mục (Indexing)**: Xây dựng cấu trúc dữ liệu inverted index để tối ưu tìm kiếm.
3. **Lưu trữ**: Dữ liệu được lưu vào các shard dựa trên ID của document.

3.3. Tìm Kiếm Và Phân Tích Dữ Liệu

3.3.1. Elasticsearch Query DSL

Ngôn ngữ truy vấn dựa trên JSON để tương tác với Elasticsearch. Một số loại truy vấn phổ biến:

a. Truy vấn cơ bản:

- **Match Query**: Tìm kiếm dựa trên phân tích văn bản.

```
GET /users/_search
{
  "query": {
    "match": { "name": "john" }
  }
}
```

- **Term Query**: Tìm kiếm chính xác giá trị (không phân tích văn bản).

```
GET /users/_search
{
  "query": {
    "term": { "email.keyword": "john@example.com" }
  }
}
```

b. Truy vấn nâng cao:

- **Bool Query**: Kết hợp nhiều truy vấn với must, should, must_not.

```
GET /users/_search
{
  "query": {
    "bool": {
      "must": [
        { "match": { "name": "john" } },
        { "range": { "age": { "gte": 30 } } }
      ]
    }
  }
}
```

- **Aggregations**: Phân tích dữ liệu theo nhóm (ví dụ: thống kê, tổng hợp).

```
GET /orders/_search
{
  "aggs": {
    "total_sales": { "sum": { "field": "price" } },
    "avg_price": { "avg": { "field": "price" } }
  }
}
```

3.3.2. Các bộ phân tích (Analyzers)

Analyzers xử lý văn bản trước khi lập chỉ mục, bao gồm 3 thành phần:

1. **Character Filters**: Loại bỏ hoặc thay thế ký tự (ví dụ: xóa HTML tags).
2. **Tokenizer**: Tách văn bản thành các token (ví dụ: standard tokenizer tách theo khoảng trắng).
3. **Token Filters**: Chỉnh sửa token (ví dụ: chuyển thành chữ thường, loại bỏ stopwords).

Ví dụ về quy trình phân tích:

- Văn bản: "Elasticsearch is FAST!"
- Kết quả sau phân tích (với standard analyzer): ["elasticsearch", "is", "fast"].

4. Logstash – Cơ Chế Thu Thập Và Xử Lý Dữ Liệu

Logstash là một công cụ mạnh mẽ dùng để thu thập, xử lý và chuyển đổi dữ liệu từ nhiều nguồn khác nhau trước khi gửi đến hệ thống lưu trữ (ví dụ: Elasticsearch) hoặc các đích khác. Việc sử dụng Logstash giúp ta kiểm soát và chuyển đổi dữ liệu theo ý muốn, từ việc làm sạch thông tin đến việc định dạng lại dữ liệu phù hợp với các yêu cầu phân tích.

4.1. Kiến Trúc Pipeline

Một pipeline của Logstash là chuỗi các bước xử lý dữ liệu, được chia làm ba phần chính: **Input, Filter, và Output**.

4.1.1. Input

Chức năng: Nhận dữ liệu từ các nguồn khác nhau. **Ví dụ về nguồn dữ liệu:**

- **File:** Đọc dữ liệu từ các file log.
- **Syslog:** Nhận dữ liệu log từ các hệ thống qua giao thức Syslog.
- **TCP/UDP:** Nhận dữ liệu qua kết nối mạng (TCP hoặc UDP).
- **Beats:** Nhận dữ liệu từ các Beats hoặc Elastic Agent gửi về.
- **HTTP, Kafka, JDBC:** Các nguồn dữ liệu đa dạng khác.

4.2.2. Filter

Chức năng: Xử lý, chuyển đổi, và làm sạch dữ liệu trước khi chuyển đến đích. **Các thao tác phổ biến:**

- **Phân tích dữ liệu:** Sử dụng plugin **grok** để tách các thành phần của chuỗi log theo các pattern đã định nghĩa.
- **Biến đổi dữ liệu:** Plugin **mutate** dùng để thay đổi tên trường, chuyển đổi kiểu dữ liệu, hoặc loại bỏ các trường không cần thiết.
- **Xử lý thời gian:** Plugin **date** chuyển đổi định dạng thời gian trong dữ liệu thành dạng chuẩn, thường ánh xạ vào trường @timestamp.
- **Các plugin bổ sung:** Các plugin như **geoip** để thêm thông tin vị trí từ địa chỉ IP, **csv**, **json** hoặc **xml** để xử lý dữ liệu ở các định dạng đặc biệt.

4.2.3. Output

Chức năng: Gửi dữ liệu đã được xử lý đến đích cuối cùng. **Ví dụ về đích đến:**

- **Elasticsearch:** Để lưu trữ và truy vấn dữ liệu.
- **File:** Ghi dữ liệu ra file, phục vụ cho mục đích lưu trữ hoặc kiểm thử.
- **Kafka:** Đẩy dữ liệu đến hệ thống message queue.
- **HTTP:** Gửi dữ liệu đến một API endpoint.
- **Các đích khác:** Email, database, hay hệ thống SIEM,...

4.2. Các Plugin Và Công Cụ Xử Lý Dữ Liệu

Logstash được mở rộng thông qua các plugin, cho phép xử lý và chuyển đổi dữ liệu một cách linh hoạt.

4.2.1. Plugin Input

Mục đích: Nhận dữ liệu từ nhiều nguồn khác nhau. **Các ví dụ điển hình:**

- **File:** Đọc dữ liệu từ file trên hệ thống.
- **Syslog:** Nhận dữ liệu từ hệ thống syslog.

- **Beats:** Nhận dữ liệu từ các agent như Filebeat, Metricbeat, hoặc Elastic Agent.
- **TCP/UDP:** Nhận dữ liệu qua giao thức mạng.
- **Kafka, JDBC:** Nhận dữ liệu từ các hệ thống message queue hoặc cơ sở dữ liệu.

4.2.2. Plugin Filter

Mục đích: Xử lý, chuyển đổi và làm sạch dữ liệu sau khi nhận được. **Các plugin tiêu biểu:**

- **Grok:** Phân tích và trích xuất các phần thông tin từ chuỗi log dựa trên các pattern định sẵn. *Ví dụ:*

```
filter {
  grok {
    match => { "message" => "%{COMBINEDAPACHELOG}" }
  }
}
```

- **Mutate:** Thay đổi, chuyển đổi hoặc loại bỏ các trường trong dữ liệu.

```
filter {
  mutate {
    rename => { "old_field" => "new_field" }
  }
}
```

- **Date:** Chuyển đổi các giá trị thời gian về định dạng chuẩn và ánh xạ vào trường @timestamp.

```
filter {
  date {
    match => [ "timestamp", "ISO8601" ]
  }
}
```

- **GeoIP:** Tự động thêm thông tin vị trí địa lý từ địa chỉ IP.
- **Các plugin xử lý khác:** CSV, JSON, XML, v.v.

4.2.3. Plugin Output

Mục đích: Chuyển dữ liệu đã xử lý đến hệ thống lưu trữ hoặc các đích khác. **Các plugin phổ biến:**

- **Elasticsearch:** Gửi dữ liệu vào Elasticsearch để lưu trữ và tìm kiếm. *Ví dụ:*

```
output {
  elasticsearch {
    hosts => ["http://localhost:9200"]
    index => "logs-%{+YYYY.MM.dd}"
  }
}
```

- **File:** Ghi dữ liệu ra file.
- **Kafka:** Gửi dữ liệu đến một topic trên Kafka.
- **HTTP:** Đẩy dữ liệu qua giao thức HTTP đến endpoint khác.
- **Stdout:** In dữ liệu ra console (thường dùng cho mục đích kiểm thử).

4.3. Lợi Ích Và Hạn Chế Của Việc Xử Lý Dữ Liệu Qua Logstash

4.3.1. Lợi Ích

Tính linh hoạt cao: Với khả năng sử dụng nhiều plugin, Logstash cho phép xử lý dữ liệu theo nhiều cách khác nhau, từ phân tích, làm sạch đến chuyển đổi dữ liệu theo yêu cầu của hệ thống.

Khả năng tích hợp đa dạng: Logstash có thể kết nối với nhiều nguồn dữ liệu (file, syslog, database, message queue, v.v.) và gửi dữ liệu đến các đích như Elasticsearch, Kafka, hoặc hệ thống lưu trữ khác.

Xử lý dữ liệu phức tạp: Nhờ các plugin như grok, mutate, date, và geoip, Logstash có thể chuyển đổi các dữ liệu phức tạp thành dạng có cấu trúc, dễ dàng truy vấn và phân tích.

Chuẩn hóa dữ liệu: Giúp chuẩn hóa dữ liệu từ nhiều nguồn khác nhau, đảm bảo tính nhất quán trước khi lưu trữ và phân tích trên Elasticsearch hoặc các hệ thống khác.

4.3.2. Hạn Chế

Chi phí tài nguyên: Việc xử lý dữ liệu thông qua Logstash có thể tiêu tốn tài nguyên (CPU, bộ nhớ) đáng kể, đặc biệt trong các hệ thống có khối lượng dữ liệu lớn hoặc khi thực hiện nhiều thao tác xử lý phức tạp.

Độ phức tạp trong cấu hình: Việc cấu hình pipeline với nhiều plugin khác nhau có thể trở nên phức tạp, đòi hỏi kiến thức sâu về cấu hình cũng như cách tối ưu hóa để đảm bảo hiệu suất hoạt động cao.

Độ trễ xử lý: So với việc gửi dữ liệu trực tiếp từ Beats/Elastic Agent đến Elasticsearch, sử dụng Logstash có thể tạo ra độ trễ thêm do bước xử lý dữ liệu trung gian.

Khó khăn trong debug: Khi pipeline trở nên phức tạp, việc debug hoặc theo dõi các lỗi phát sinh trong quá trình xử lý dữ liệu có thể gặp nhiều khó khăn, đòi hỏi các công cụ giám sát và log chi tiết.

5. Kibana – Trực Quan Hóa Và Phân Tích Dữ Liệu

Kibana là giao diện trực quan của Elastic Stack, giúp người dùng tương tác, phân tích và hiển thị dữ liệu được lưu trữ trong Elasticsearch thông qua các biểu đồ, dashboard và báo cáo. Dưới đây là các thành phần và cách thức sử dụng Kibana để tối ưu hóa việc trực quan hóa và phân tích dữ liệu.

5.1. Các Thành Phần Cơ Bản Của Kibana

5.1.1. Index Patterns

Định nghĩa: Index pattern là định nghĩa cho phép Kibana biết được những chỉ mục (index) nào trong Elasticsearch sẽ được sử dụng để truy vấn và trực quan hóa dữ liệu.

Chức năng: Xác định phạm vi dữ liệu (ví dụ: logs-* cho tất cả các index chứa log). Chỉ định trường thời gian (timestamp field) để hỗ trợ các truy vấn dựa trên khoảng thời gian.

Cách cấu hình: Thông qua giao diện **Management → Stack Management → Index Patterns** trong Kibana, người dùng có thể tạo mới hoặc chỉnh sửa các index pattern.

5.1.2. Dashboards

Định nghĩa: Dashboard là bộ sưu tập của các visualizations, cho phép người dùng tổng hợp và theo dõi các số liệu, biểu đồ cùng lúc.

Chức năng: Tạo không gian hiển thị tổng hợp dữ liệu từ các nguồn khác nhau. Cung cấp khả năng theo dõi và giám sát dữ liệu thời gian thực. Cho phép người dùng tương tác (zoom, filter, drill-down) để phân tích sâu hơn.

Cách sử dụng: Dashboard có thể được tạo mới bằng cách kéo thả các visualization đã có sẵn hoặc tạo trực tiếp trên giao diện Dashboard của Kibana.

5.1.3. Visualizations

Định nghĩa: Visualizations là các biểu đồ, đồ thị, bảng số liệu được tạo ra từ dữ liệu truy xuất từ Elasticsearch.

Chức năng: Hiển thị dữ liệu dưới dạng trực quan, giúp người dùng dễ dàng nhận diện xu hướng, bất thường và mối quan hệ giữa các dữ liệu. Các loại visualizations có thể bao gồm: bar chart, pie chart, line chart, area chart, heat map, data table, scatter plot, v.v.

Cách tạo: Thông qua mục **Visualize Library** trong Kibana, người dùng có thể tạo visualizations dựa trên các truy vấn đã được định nghĩa hoặc dựa trên index pattern.

5.2. Ngôn Ngữ Truy Vấn Trong Kibana

5.2.1. Kibana Query Language (KQL)

Định nghĩa: Kibana Query Language (KQL) là ngôn ngữ truy vấn được tích hợp sẵn trong Kibana, cho phép người dùng thực hiện các truy vấn tìm kiếm một cách trực quan và dễ hiểu.

Đặc điểm nổi bật:

- Cú pháp đơn giản: Cho phép kết hợp các toán tử như AND, OR, NOT, và sử dụng dấu ngoặc để nhóm các điều kiện.

- **Tìm kiếm dạng full-text:** Hỗ trợ tìm kiếm theo văn bản, số, hoặc các giá trị định danh.

- **Tích hợp với các filter:** Người dùng có thể áp dụng các bộ lọc (filter) để thu hẹp phạm vi dữ liệu truy vấn.

5.2.2. Cách thức truy vấn dữ liệu với KQL

Ví dụ 1: Tìm kiếm các document có trường user bằng "admin":

```
user: "admin"
```

Ví dụ 2: Tìm kiếm các log trong khoảng thời gian cụ thể:

```
@timestamp >= "2025-02-10T00:00:00" and @timestamp <= "2025-02-11T23:59:59"
```

Ví dụ 3: Kết hợp nhiều điều kiện:

```
user: "admin" and response: 200 and path: "/api/*"
```

5.3. Các Loại Biểu Đồ Và Báo Cáo

5.3.1. Lựa chọn biểu đồ phù hợp cho từng loại dữ liệu và mục tiêu phân tích

- Bar Chart: Sử dụng khi so sánh dữ liệu giữa các nhóm (ví dụ: số lượng log theo từng loại error).

- Line Chart: Sử dụng khi theo dõi xu hướng của dữ liệu theo thời gian (ví dụ: lượng truy cập website hàng ngày).

- Pie Chart: Sử dụng khi hiển thị tỷ lệ phần trăm của các thành phần trong tập hợp (ví dụ: phân chia thị phần, loại log).

- **Area Chart:** Sử dụng khi hiển thị xu hướng tổng hợp, đặc biệt là khi cần thể hiện tổng giá trị theo thời gian.

- Heat Map: Sử dụng khi hiển thị mật độ dữ liệu, phát hiện vùng dữ liệu tập trung hay bất thường.

- **Data Table:** Sử dụng kh cung cấp chi tiết dữ liệu dạng bảng, dễ dàng sắp xếp và lọc thông tin.

- **Gauge, Metric Visualizations:** Sử dụng khi hiển thị các chỉ số đơn lẻ, biểu thị hiệu suất hoặc mục tiêu đạt được.

5.3.2. Báo cáo trong Kibana

- Tạo báo cáo động: Người dùng có thể tạo các dashboard báo cáo tích hợp nhiều visualizations để cung cấp bức tranh toàn diện về dữ liệu.

- **Tùy chỉnh thời gian và bộ lọc:** Báo cáo có thể được tùy chỉnh theo khoảng thời gian cụ thể và có các bộ lọc để hiển thị thông tin phù hợp với mục tiêu phân tích.

- **Chia sẻ và xuất báo cáo:** Kibana cho phép người dùng chia sẻ dashboard qua URL hoặc xuất ra các định dạng báo cáo (PDF, CSV) để gửi cho các bên liên quan.

6. Beats – Thu Thập Dữ Liệu Từ Các Nguồn Khác Nhau

6.1. Giới Thiệu Về Beats

- Các loại Beats:

1. **Filebeat:** Thu thập dữ liệu từ file log.
2. **Metricbeat:** Thu thập số liệu hiệu năng, metric của hệ thống.
3. **Winlogbeat:** Thu thập log từ hệ điều hành Windows.
4. **Packetbeat:** Phân tích lưu lượng mạng.

- Vai trò: Beats là các agent nhẹ, được cài đặt trên các máy chủ để thu thập dữ liệu từ nhiều nguồn khác nhau và gửi về Logstash hoặc Elasticsearch, giúp hệ sinh thái Elastic Stack có được nguồn dữ liệu phong phú và kịp thời.

6.2. Cách Beats Tương Tác Với Logstash Và Elasticsearch

Giao thức: Beats sử dụng các giao thức như TCP hoặc HTTPS để truyền dữ liệu.

Định dạng dữ liệu: Dữ liệu được định dạng theo dạng JSON, đảm bảo tính nhất quán và dễ dàng xử lý.

Truyền tải: Có thể gửi trực tiếp đến Elasticsearch để tối ưu hiệu suất và giảm độ trễ. Hoặc gửi qua Logstash để áp dụng các bước xử lý, lọc và chuyển đổi dữ liệu trước khi lưu trữ.

6.3. Lợi Ích Khi Sử Dụng Beats

Tính nhẹ: Tiêu thụ tài nguyên tối thiểu, dễ cài đặt trên nhiều hệ thống.

Dễ triển khai: Cấu hình đơn giản, có thể nhanh chóng tích hợp vào hệ thống hiện có.

Tối ưu hoá quá trình thu thập dữ liệu: Cho phép thu thập dữ liệu từ nhiều nguồn đa dạng một cách hiệu quả và đồng bộ, tạo tiền đề cho quá trình phân tích và giám sát trên Elastic Stack.

7. Bảo Mật Trong Elastic Stack

Elastic Stack không chỉ tập trung vào thu thập và phân tích dữ liệu mà còn chú trọng đến việc bảo vệ dữ liệu và đảm bảo an toàn cho toàn bộ hệ thống. Việc triển khai các biện pháp bảo mật phù hợp giúp ngăn chặn truy cập trái phép, bảo vệ dữ liệu khi truyền tải và lưu trữ, đồng thời phát hiện sớm các hành vi bất thường.

7.1. Các Yếu Tố Bảo Mật Cơ Bản

Xác Thực (Authentication):

- Xác minh danh tính của người dùng hoặc hệ thống trước khi cho phép truy cập vào Elastic Stack.

- Có thể sử dụng các phương pháp như LDAP, Active Directory, SAML, hoặc token-based authentication để đảm bảo chỉ những đối tượng hợp lệ mới được truy cập.

Ủy Quyền (Authorization):

- Phân quyền truy cập dựa trên vai trò (role-based access control – RBAC) nhằm đảm bảo người dùng chỉ có thể truy cập các tài nguyên và dữ liệu mà họ được phép.

- Các chính sách ủy quyền cho phép kiểm soát chi tiết hành động của người dùng (đọc, ghi, xóa, thay đổi cấu hình, …).

Mã Hóa Dữ Liệu:

- Khi truyền tải (in transit): Sử dụng SSL/TLS để bảo vệ dữ liệu khi truyền qua mạng giữa các thành phần Elastic Stack (ví dụ: giữa Elasticsearch, Kibana, Logstash, Beats/Elastic Agent).

- Khi lưu trữ (at rest): Mã hóa dữ liệu được lưu trữ trên đĩa để bảo vệ thông tin nhạy cảm trong trường hợp có sự xâm nhập vật lý hoặc vi phạm hệ thống lưu trữ.

7.2. Bảo Vệ Cluster Elasticsearch Và Kibana

Cấu Hình SSL/TLS:

- Thiết lập chứng chỉ SSL/TLS cho các node trong cluster Elasticsearch nhằm mã hóa dữ liệu trao đổi giữa các node.

- Cấu hình SSL/TLS cho kết nối giữa Elasticsearch và các client (như Kibana, Logstash, Beats/Elastic Agent) giúp ngăn chặn nghe lén và tấn công trung gian.

Role-Based Access Control (RBAC):

- Sử dụng RBAC để định nghĩa rõ ràng các vai trò và quyền hạn truy cập vào các tài nguyên của Elasticsearch và Kibana.

- Đảm bảo rằng người dùng chỉ được phép thực hiện các hành động phù hợp với vai trò của họ, hạn chế rủi ro do truy cập trái phép hoặc thao tác ngoài phạm vi cho phép.

Việc sử dụng ABAC trong Elastic Stack:

- Mặc dù Elastic Stack không cung cấp hỗ trợ ABAC tích hợp sẵn, nhưng ta có thể xây dựng hoặc tích hợp các giải pháp tùy chỉnh để mô phỏng hoặc kết hợp ABAC vào hệ thống của mình.

- Điều này có thể đòi hỏi thêm công việc về cấu hình, phát triển hoặc sử dụng các giải pháp bảo mật bên ngoài để thực hiện các chính sách ABAC.

- Một số doanh nghiệp có thể triển khai ABAC bằng cách sử dụng metadata hoặc các trường bổ sung trong tài liệu để xác định các thuộc tính cần thiết, sau đó tích hợp với hệ thống quản lý truy cập tùy chỉnh.

8. Ứng Dụng Lý Thuyết Trong Thực Tế

Phần này tập trung vào việc chuyển giao các khái niệm lý thuyết đã được trình bày vào các ứng dụng cụ thể, từ việc giám sát log hệ thống đến tối ưu hóa tìm kiếm dữ liệu, đồng thời so sánh ELK với các giải pháp tương tự khác trên thị trường.

8.1. Phân Tích Các Case Study

(Để làm sau)

8.2. Áp Dụng Các Nguyên Tắc Lý Thuyết Vào Thiết Kế Hệ Thống

Chiến Lược Tối Ưu Hóa Tìm Kiếm Và Phân Tích Dữ Liệu

Thiết kế index và mapping: Áp dụng các nguyên tắc của Elasticsearch trong việc tạo index phù hợp với loại dữ liệu cần lưu trữ, tối ưu hóa các trường tìm kiếm (keyword vs. text) để đảm bảo hiệu suất truy vấn.

Quản lý cluster: Xây dựng một cluster phân tán với sharding và replication phù hợp để cân bằng tải và đảm bảo tính sẵn sàng cao.

Tối ưu hóa pipeline của Logstash: Thiết lập pipeline hiệu quả, sử dụng các bộ lọc phù hợp (grok, mutate, date) để làm sạch dữ liệu, đồng thời cấu hình ILM (Index Lifecycle Management) để quản lý vòng đời của index.

Trực quan hóa và báo cáo: Xây dựng các dashboard trên Kibana phản ánh các chỉ số chính của hệ thống, kết hợp với bộ lọc thời gian và các truy vấn KQL để người dùng có thể tương tác và phân tích dữ liệu một cách trực quan.

Tích hợp hệ thống cảnh báo: Áp dụng các quy tắc và alerts trong Kibana/Elastic SIEM để theo dõi dữ liệu thời gian thực, từ đó phát hiện sớm các sự cố hoặc hành vi bất thường.

8.3. So Sánh Với Các Giải Pháp Khác

8.3.1. Ưu Điểm Của ELK

- Tính mở rộng và linh hoạt: Elastic Stack được thiết kế theo kiến trúc phân tán, dễ dàng mở rộng khi khối lượng dữ liệu tăng.

- **Khả năng tìm kiếm và phân tích mạnh mẽ:** Sử dụng Elasticsearch với khả năng tìm kiếm full-text và các kỹ thuật phân tích dữ liệu mạnh mẽ giúp xử lý lượng lớn dữ liệu thời gian thực.

- **Trực quan hóa dữ liệu:** Kibana cung cấp giao diện trực quan, dễ sử dụng cho việc tạo dashboard và báo cáo, hỗ trợ người dùng trong việc ra quyết định.

- **Tích hợp bảo mật và cảnh báo:** Elastic SIEM tích hợp giúp giám sát an ninh mạng, phát hiện sớm các hành vi bất thường và tấn công.

8.3.2. Nhược Điểm Và Hạn Chế So Với Các Giải Pháp Khác

- **Độ phức tạp trong cấu hình:** Việc triển khai và cấu hình một hệ thống Elastic Stack hoàn chỉnh (bao gồm cả bảo mật, xử lý dữ liệu và quản lý cluster) có thể phức tạp, đặc biệt đối với các tổ chức nhỏ hoặc thiếu kinh nghiệm.

- **Yêu cầu tài nguyên cao:** Đối với các hệ thống có khối lượng dữ liệu lớn, việc duy trì hiệu suất cao của Elasticsearch, Logstash và Kibana đòi hỏi tài nguyên phần cứng đáng kể.

- **Cạnh tranh với các giải pháp SIEM chuyên dụng:** Các giải pháp SIEM khác như Splunk, Sumo Logic hoặc Graylog có thể cung cấp giao diện và tính năng chuyên sâu cho mục đích phân tích và giám sát an ninh, mặc dù chi phí của chúng thường cao hơn.

- **Hỗ trợ tích hợp ABAC:** Trong khi ELK chủ yếu hỗ trợ RBAC, các hệ thống sử dụng ABAC (Attribute-Based Access Control) có thể cần các giải pháp tùy chỉnh để tích hợp, điều này có thể gây thêm độ phức tạp cho hệ thống bảo mật.