

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ - ĐẠI HỌC QUỐC GIA HÀ NỘI



Nhóm 2

**GIÁM SÁT DỊCH VỤ VÀ QUẢN LÝ LOG DOCKER
CONTAINER VỚI ELK STACK**

BÁO CÁO BÀI TẬP LỚN

Học phần: Quản trị mạng INT3310

Hà Nội, 2024

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ - ĐẠI HỌC QUỐC GIA HÀ NỘI

Nhóm 2

**GIÁM SÁT DỊCH VỤ VÀ QUẢN LÝ LOG DOCKER
CONTAINER VỚI ELK STACK**

BÁO CÁO BÀI TẬP LỚN

Học phần: Quản trị mạng INT3310

Giảng viên: Dương Lê Minh

Các thành viên:

Nguyễn Thị Ngọc Mai

Trần Khánh Duy

Đinh Hồng Khanh

Trương Quang Minh

Phạm Xuân Dương

Hoàng Minh Nghĩa

Hà Nội, 2024

Mục lục

Mở đầu	5
1 Giám sát log và giám sát log tập trung	6
1.1 Giám sát log	6
1.1.1 Khái niệm log	6
1.1.2 Khái niệm giám sát log	8
1.1.3 Tính cần thiết của giám sát log	9
1.2 Giám sát log tập trung	10
1.2.1 Khái niệm giám sát log tập trung	10
1.2.2 Ưu điểm và nhược điểm	11
2 Tổng quan về ELK Stack	12
2.1 Giới thiệu chung về ELK Stack	12
2.1.1 Khái niệm và lịch sử phát triển của ELK Stack	12
2.1.2 Ưu điểm và nhược điểm	14
2.1.3 Ứng dụng của ELK Stack và một số trường hợp sử dụng	15
2.2 Các thành phần trong ELK Stack	16
2.2.1 Elasticsearch	16
2.2.2 Logstash	19
2.2.3 Kibana	22
2.2.4 Beats	23
2.3 Vận hành hệ thống ELK Stack	25
2.3.1 Cách thức và hoạt động của các thành phần	25
2.3.2 Cài đặt và cấu hình ELK Stack	25
2.3.3 Các vấn đề cần lưu ý khi vận hành hệ thống ELK Stack	29
3 Docker và Docker Compose	31
3.1 Docker	31
3.1.1 Giới thiệu chung	31
3.1.2 Một số tính năng nổi bật	32
3.1.3 Ứng dụng thực tế	33

3.2	Docker Compose	34
3.2.1	Giới thiệu chung	34
3.2.2	Quy trình hoạt động	35
3.2.3	Lợi ích của Docker Compose	37
3.2.4	Ứng dụng thực tế	37
4	DEMO	39
4.1	Tổng quan	39
4.2	Thực nghiệm	41
4.2.1	Mục tiêu thực nghiệm	41
4.2.2	Cài đặt và cấu hình các thành phần	41
4.2.3	Khởi chạy Demo	48
4.2.4	Kết quả thực nghiệm	52
Tổng kết		60
Tài liệu tham khảo		61
Bảng đóng góp của các thành viên		62

Mở đầu

Trong thời đại chuyển đổi số, các tổ chức và doanh nghiệp ngày càng phụ thuộc vào hệ thống công nghệ thông tin để vận hành và cung cấp dịch vụ. Sự phức tạp ngày càng tăng của hệ thống đòi hỏi một cách tiếp cận toàn diện để giám sát dịch vụ, quản lý log và đảm bảo an toàn cho dữ liệu. Báo cáo này giới thiệu phương pháp tích hợp ELK Stack – bộ công cụ mạnh mẽ gồm Elasticsearch, Logstash, Kibana và Beats – với Docker để xây dựng một hệ thống giám sát log tập trung. ELK Stack cung cấp giải pháp thu thập, phân tích và trực quan hóa dữ liệu log từ nhiều nguồn khác nhau, trong đó Docker đóng vai trò nền tảng để quản lý các ứng dụng container hóa.

Báo cáo không chỉ trình bày lý thuyết về các thành phần của ELK Stack mà còn tập trung vào các bước triển khai thực tế, từ cài đặt, cấu hình cho đến vận hành hệ thống. Qua mô hình này, các quản trị viên có thể tập trung hóa dữ liệu log từ nhiều container Docker, phân tích dữ liệu log theo thời gian thực và phát hiện nhanh chóng các sự cố hệ thống. Báo cáo cũng nhấn mạnh cách mà ELK Stack kết hợp với Docker giúp cải thiện hiệu quả vận hành, tăng cường bảo mật và đảm bảo hiệu suất hệ thống một cách ổn định. Đây là một giải pháp toàn diện, phù hợp với môi trường công nghệ thông tin hiện đại và các doanh nghiệp đang hướng đến mô hình dịch vụ linh hoạt, an toàn và hiệu quả.

Chương 1

Giám sát log và giám sát log tập trung

1.1 Giám sát log

1.1.1 Khái niệm log

Log hay còn gọi là Bản ghi là một hoạt động vô cùng quan trọng trong quản trị hệ thống và phát triển phần mềm, đóng vai trò thiết yếu trong việc ghi lại những thông tin và sự kiện đáng chú ý diễn ra trong suốt quá trình vận hành của hệ thống hoặc ứng dụng. Mục đích chính của logging là tạo ra một cơ chế theo dõi và lưu trữ các bản ghi (log) chi tiết về các hoạt động, trạng thái của hệ thống, cũng như các sự kiện lỗi hoặc cảnh báo cấp thiết. Các bản ghi này không chỉ giúp quản trị viên theo dõi hiệu suất hệ thống, mà còn hỗ trợ quá trình phân tích sự cố, debug lỗi và đảm bảo hệ thống hoạt động ổn định. Đặc biệt khi xảy ra sự cố, dữ liệu log trở thành một tài nguyên quý giá để chẩn đoán vấn đề khi cung cấp một nguồn thông tin phong phú về các dấu hiệu bất thường, những biểu hiện tiêu cực hoặc lỗi. Bằng cách phân tích các bản ghi log, người quản trị có thể xác định nguyên nhân của sự cố và áp dụng biện pháp khắc phục phù hợp và kịp thời.

Thông thường, các bản ghi log được tổ chức theo một định dạng thống nhất và chuẩn hóa, điều này giúp việc truy xuất, đọc và phân tích thông tin trở nên logic, thuận tiện. Định dạng này có thể bao gồm các thông tin cơ bản: thời gian xảy ra sự kiện; nguồn gốc của sự kiện (ví dụ: module, dịch vụ hoặc hệ thống tạo ra sự kiện); mức độ nghiêm trọng của sự kiện (thông thường được phân loại theo các mức độ như INFO, WARNING, ERROR, CRITICAL), và các thông điệp mô tả chi tiết về sự kiện hoặc vấn đề đang xảy ra.

Các bản ghi log không chỉ chứa thông tin về các lỗi hoặc sự cố mà còn ghi lại những hoạt động bình thường trong hệ thống, giúp người quản trị hệ thống dễ dàng theo dõi và phân tích các xu hướng hoạt động, tìm ra các điểm yếu hoặc những chỗ có thể tối ưu hóa hiệu suất. Hơn nữa, thông qua việc phân tích các thông điệp trong log, các chuyên gia có thể xác định được các sự kiện quan trọng, đánh giá mức độ ảnh hưởng của chúng và đưa ra các biện pháp khắc phục kịp thời.

Dựa vào các chức năng, đặc điểm mà ta có thể phân chia Logging theo 4 loại cụ thể:

- **Access Logs:** Đây là loại log ghi lại tất cả các yêu cầu và phản hồi từ người dùng khi họ tương tác với hệ thống, dịch vụ hoặc ứng dụng. Chẳng hạn, khi người dùng truy cập một trang web, gọi API hoặc thực hiện các hành động trên các dịch vụ trực tuyến, các thông tin như địa chỉ IP của người dùng, thời gian truy cập, URL yêu cầu, mã trạng thái HTTP (200, 404, 500...), thời gian phản hồi của máy chủ và thông tin về trình duyệt (User Agent) sẽ được ghi lại trong access logs. Ví dụ, khi một người dùng truy cập vào một trang sản phẩm trên website, thông tin về thời gian truy cập, IP của người dùng, và các phản hồi từ server như mã lỗi (nếu có) sẽ được ghi lại để phân tích hoạt động người dùng và tối ưu hóa hiệu suất.
- **Error Logs:** Loại log này chủ yếu ghi lại các lỗi, sự cố hoặc thông báo cảnh báo trong hệ thống hoặc ứng dụng. Khi hệ thống gặp phải sự cố hoặc không thể thực hiện một yêu cầu nào đó, các thông tin về lỗi như mã lỗi, thông điệp chi tiết, và thông tin về ngữ cảnh xảy ra lỗi (như stack trace) sẽ được ghi lại trong error logs. Điều này giúp các nhà phát triển hoặc quản trị viên hệ thống nhanh chóng phát hiện và khắc phục vấn đề. Ví dụ, nếu một yêu cầu của người dùng không thành công vì cơ sở dữ liệu không thể truy cập, hệ thống sẽ ghi lại một lỗi trong error logs với các chi tiết về nguyên nhân và vị trí xảy ra lỗi.
- **Event Logs:** Đây là loại log dùng để ghi lại các sự kiện hoặc hành động trong hệ thống hoặc ứng dụng. Các sự kiện có thể bao gồm những thay đổi trạng thái hệ thống, các tác vụ được thực thi hoặc các hành động quan trọng của người dùng. Ví dụ, trong một ứng dụng ngân hàng, khi người dùng thực hiện một giao dịch chuyển tiền hoặc thay đổi mật khẩu, các sự kiện đó sẽ được ghi lại trong event logs, bao gồm thông tin như ai thực hiện hành động, vào thời điểm nào, hành động gì, và các thông tin xác thực cần thiết để kiểm tra lại sau này nếu có sự cố.
- **Transaction Logs:** Đây là loại log chủ yếu được sử dụng trong các hệ thống cơ sở dữ liệu để ghi lại mọi thay đổi dữ liệu (các giao dịch) trong quá trình hoạt động của hệ thống. Mỗi khi một giao dịch được thực hiện, chẳng hạn như cập nhật dữ liệu, xóa hoặc thêm bản ghi mới, hệ thống sẽ ghi lại thông tin về giao dịch đó trong transaction logs. Điều này giúp đảm bảo rằng hệ thống có thể khôi phục lại trạng thái ban đầu nếu xảy ra sự cố (ví dụ: mất điện hoặc hệ thống bị sập), vì transaction logs chứa thông tin về các thao tác cần thiết để phục hồi dữ liệu. Ví dụ, nếu một người dùng thực hiện một giao dịch thanh toán trong hệ thống, transaction log sẽ ghi lại các thay đổi trong cơ sở dữ liệu như thông tin người dùng, số tiền giao dịch, thời gian và các bước thực hiện giao dịch đó.

1.1.2 Khái niệm giám sát log

Giám sát log là quá trình liên tục theo dõi, thu thập, phân tích và kiểm tra các file log trong hệ thống hoặc ứng dụng với mục tiêu phát hiện các sự cố, lỗi, hoặc hành vi bất thường. Đây là một phần cấp thiết trong quản trị hệ thống và phát triển phần mềm, giúp đảm bảo tính ổn định và hiệu suất của các hệ thống công nghệ thông tin, đồng thời hỗ trợ quá trình bảo mật và tối ưu hóa hệ thống.

Quá trình giám sát Log bao gồm:

- **Theo dõi (Monitoring):** Quá trình giám sát log bắt đầu bằng việc thiết lập các công cụ và phương pháp để theo dõi các file log trong thời gian thực. Các công cụ giám sát thường xuyên thu thập thông tin từ các file log và phân tích dữ liệu để phát hiện các dấu hiệu của sự cố hoặc hành vi không bình thường. Quá trình này giúp các quản trị viên hệ thống có thể nhận diện vấn đề ngay khi chúng mới xuất hiện, từ đó giảm thiểu rủi ro và tăng khả năng phục hồi của hệ thống.
- **Thu thập dữ liệu:** Quá trình thu thập log là bước tiếp theo trong giám sát log, nơi mà các log từ các nguồn khác nhau sẽ được tập hợp về một nơi duy nhất, thường là một hệ thống lưu trữ hoặc cơ sở dữ liệu chuyên dụng. Các log này có thể được thu thập theo thời gian thực hoặc theo lịch trình định kỳ, tùy thuộc vào yêu cầu của hệ thống và ứng dụng. Các công cụ thu thập log phổ biến như Elastic Stack (ELK Stack), Splunk, hoặc Graylog cung cấp khả năng thu thập và tập hợp log từ nhiều nguồn khác nhau, bao gồm các máy chủ, thiết bị mạng, hoặc các ứng dụng đám mây.
- **Phân tích log:** Sau khi thu thập được dữ liệu từ các file log, bước tiếp là phân tích nội dung của chúng. Phân tích log giúp tìm ra các dấu hiệu của sự cố hoặc các hành vi bất thường, chẳng hạn như lỗi hệ thống, sự cố hiệu suất, hoặc các cuộc tấn công bảo mật. Các công cụ phân tích log thường cung cấp các chức năng như:
 - **Tìm kiếm thông tin:** Cho phép người dùng tìm kiếm các sự kiện cụ thể trong log dựa trên từ khóa, ngày giờ, hoặc các tham số khác.
 - **Phân tích xu hướng:** Giúp nhận diện các xu hướng trong hoạt động của hệ thống, ví dụ như số lượng lỗi hoặc thời gian phản hồi của ứng dụng theo thời gian.
 - **Phát hiện bất thường (Anomaly detection):** Các thuật toán học máy có thể được sử dụng để phát hiện các hành vi không bình thường hoặc khác biệt so với các mẫu hành vi thông thường.

Phân tích log không chỉ dừng lại ở việc tìm lỗi mà còn giúp dự đoán các sự cố tiềm ẩn và cung cấp thông tin quan trọng để tối ưu hóa hệ thống.

- Kiểm tra và phản ứng: Sau khi phân tích, hệ thống giám sát log có thể thiết lập các cảnh báo tự động để thông báo cho các quản trị viên khi có sự cố xảy ra. Các cảnh báo này có thể được cấu hình theo các tiêu chí cụ thể như mức độ nghiêm trọng của sự kiện (ví dụ: lỗi hệ thống, tấn công bảo mật), tần suất lỗi hoặc các sự kiện bất thường khác.

1.1.3 Tính cần thiết của giám sát log

Log được phát triển với mục đích đảm bảo hệ thống luôn hoạt động ổn định, giúp phát hiện kịp thời các vấn đề và hỗ trợ quá trình khắc phục sự cố một cách nhanh chóng và hiệu quả. Giám sát log đóng một vai trò quan trọng trong việc duy trì tính liên tục và hiệu suất của hệ thống. Cụ thể:

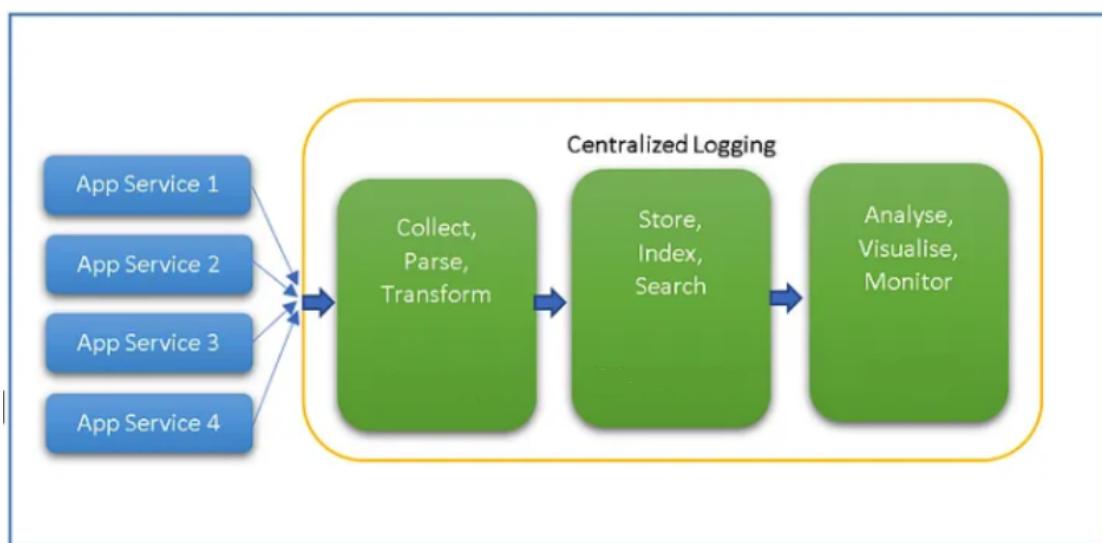
- **Ghi nhận sự kiện hệ thống:** Log ghi lại tất cả các sự kiện trong hệ thống, bao gồm các hành động của người dùng, các lỗi phần mềm, và các sự kiện quan trọng khác. Mỗi sự kiện này đều được ghi lại dưới dạng một bản ghi chi tiết với thông tin về thời gian, nguồn gốc, và các tham số liên quan. Nếu có bất kỳ hành động nào đáng ngờ hoặc có dấu hiệu bất thường, log sẽ là tài liệu quý giá để giúp xác định nguồn gốc của sự cố.
- **Phát hiện lỗi và xác định gốc rễ vấn đề:** Một trong những nhiệm vụ quan trọng của giám sát log là phát hiện lỗi trong hệ thống. Log ghi lại thông tin về các lỗi, sự cố hoặc thông báo từ hệ thống, từ đó giúp nhanh chóng nhận diện những vấn đề nghiêm trọng đang xảy ra. Các thông điệp lỗi trong log có thể cung cấp các chỉ dẫn cụ thể, như mã lỗi, thông báo chi tiết và ngữ cảnh liên quan, giúp xác định được nguyên nhân gốc rễ của sự cố.
- **Cải thiện hiệu suất hệ thống:** Một ứng dụng giám sát log quan trọng khác là việc theo dõi và cải thiện hiệu suất của hệ thống. Các bản ghi log cung cấp cái nhìn chi tiết về các thành phần của hệ thống, bao gồm máy chủ, ứng dụng, cơ sở dữ liệu và các dịch vụ khác. Thông qua việc giám sát log, quản trị viên có thể phát hiện các điểm nghẽn (bottleneck) hoặc các thành phần hoạt động không hiệu quả. Ví dụ, log có thể cung cấp thông tin về thời gian phản hồi của các dịch vụ, mức sử dụng tài nguyên hệ thống như CPU, RAM, I/O và lưu lượng mạng. Việc giám sát thường xuyên những chỉ số này sẽ giúp quản trị viên phát hiện và khắc phục các vấn đề về hiệu suất trước khi chúng trở thành mối đe dọa nghiêm trọng đối với hệ thống.
- **Giám sát thời gian phản hồi và tài nguyên hệ thống:** Giám sát log không chỉ dừng lại ở việc ghi nhận sự kiện, mà còn giúp theo dõi hiệu suất cụ thể của từng thành phần trong hệ thống. Các chỉ số như thời gian phản hồi của ứng dụng, mức độ sử dụng tài nguyên (CPU, RAM, I/O) và lưu lượng mạng có thể được ghi lại trong log để phân tích hiệu quả hoạt động của hệ thống.

- **Dự báo và tối ưu hóa:** Một lợi ích dài hạn của giám sát log là khả năng dự báo các vấn đề tiềm ẩn và xu hướng phát triển của hệ thống. Việc phân tích log trong một khoảng thời gian dài giúp chúng ta nhận diện các mô hình hoạt động và dự đoán trước các sự cố có thể xảy ra.

1.2 Giám sát log tập trung

1.2.1 Khái niệm giám sát log tập trung

Giám sát log tập trung hay Centralized Logging, là một phương pháp quản lý và thu thập dữ liệu log từ các nguồn khác nhau và tập trung chúng vào một hệ thống duy nhất. Trong mô hình này, log từ các máy chủ, ứng dụng, và thiết bị mạng khác nhau được chuyển đến một trung tâm lưu trữ và quản lý. Việc ghi log tập trung có vai trò quyết định trong việc nâng cao khả năng đáp ứng và giảm thời gian giải quyết sự cố. Tập trung dữ liệu log giúp giảm thời gian và công sức cần thiết để tìm kiếm và phân tích thông tin. Thay vì phải kiểm tra từng máy chủ hay từng ứng dụng, đội ngũ quản trị có thể dễ dàng truy xuất log từ trung tâm và sử dụng các công cụ phân tích để hiểu rõ về hoạt động của toàn bộ hệ thống. Đặc biệt, khi các dữ liệu log đã được tập trung tại một điểm, các doanh nghiệp có thể dễ dàng tùy chỉnh cấu trúc, mở rộng hệ thống mà không cần phải lo lắng về việc quản lý log từ các nguồn mới.



Hình 1.1: Giám sát Log tập trung

1.2.2 Ưu điểm và nhược điểm

Vì Centralize logging có tính tập trung cao, nên ta sẽ quản lý log dễ hơn thay vì phải truy cập từng máy chủ hay hệ thống để kiểm tra log. Điều đó dẫn đến 4 lợi ích to lớn sau

- **Dễ dàng phân tích, dự đoán: Rõ ràng**, việc tập trung log vào 1 trung tâm lưu trữ và quản lý giúp ta dễ dàng phân tích dữ liệu, bên cạnh đó cũng dự đoán sớm xu hướng hệ thống hay sự cố tiềm tàng
- **Tăng cường bảo mật do log không bị phân tán ở nhiều nơi:** Việc tập trung toàn bộ log khiến ta dễ dàng kiểm soát, giống như bao bọc 1 đồ vật trong phạm vi ta quản lý được
- **Xử lý sự cố nhanh chóng:** Chính vì có sự tập trung log tại 1 nơi, khi xảy ra sự cố, ta không cần tốn nhiều thời gian để xem xét, truy cứu, tìm kiếm, phân tích các log ở các vị trí khác nhau, tránh được sự xung đột không cần thiết
- **Dễ dàng mở rộng:** Dễ dàng mở rộng hệ thống mà không cần phải lo lắng về việc quản lý log từ các nguồn mới.

Tuy nhiên, cũng chính vì tính tập trung cao của, giám sát log tập trung cần chi phí vận hành và duy trì cao để có thể tập trung log với số lượng lớn không lồ tại cùng 1 nơi, cũng cần phải có cơ sở hạ tầng vững, chịu được traffic cao mà không gây xung đột. Nếu xử lý không triệt để, không hiệu quả, sẽ gây ra giảm hiệu suất hệ thống lớn do xung đột hệ thống kéo dài

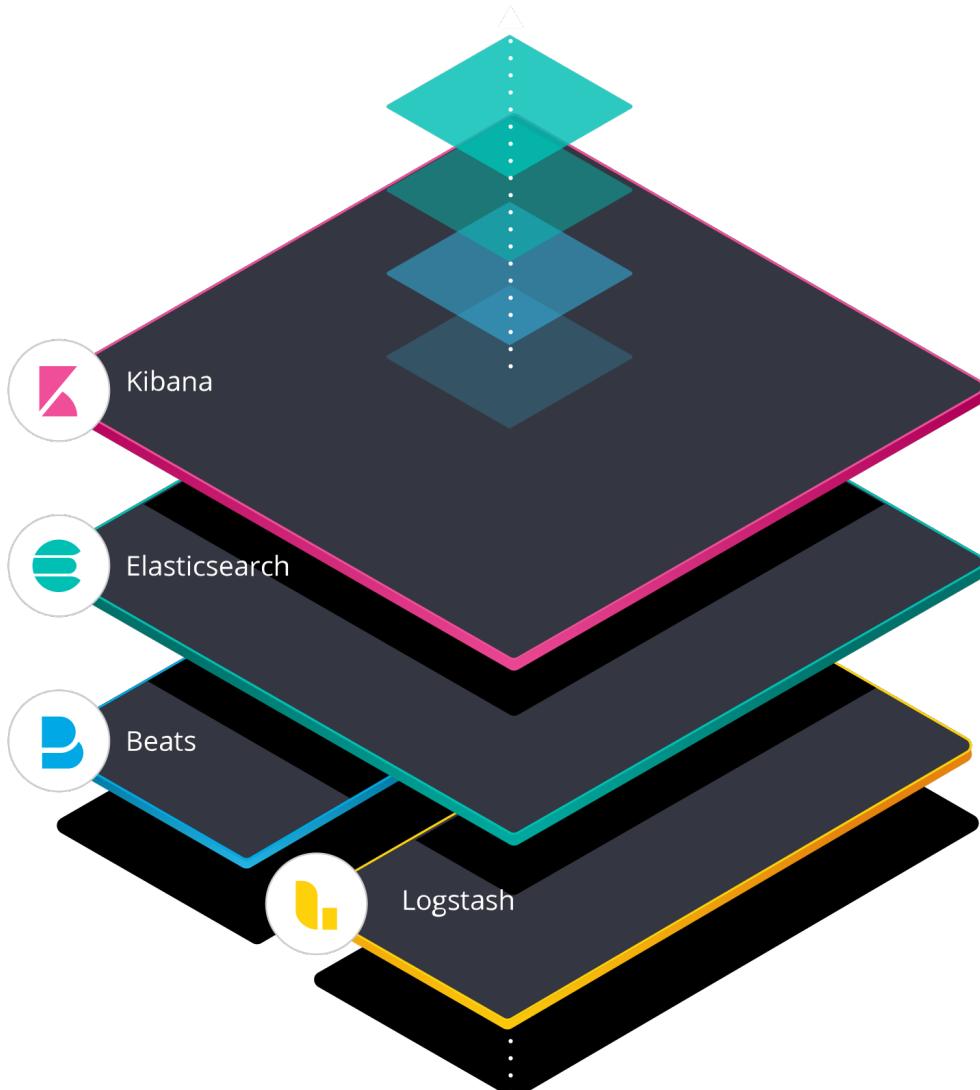
Chương 2

Tổng quan về ELK Stack

2.1 Giới thiệu chung về ELK Stack

2.1.1 Khái niệm và lịch sử phát triển của ELK Stack

ELK Stack, một bộ công cụ bao gồm Elasticsearch, Logstash, và Kibana, xuất phát từ việc phát triển Elasticsearch vào năm 2010 bởi Shay Banon. Elasticsearch, hệ thống tìm kiếm phân tán, sử dụng công nghệ Apache Lucene, nhanh chóng trở thành cột mốc quan trọng trong khả năng tìm kiếm và truy xuất dữ liệu. Logstash, dự án cá nhân của Jordan Sissel, sau đó được tích hợp để giải quyết thách thức về thu thập và xử lý log từ nhiều nguồn khác nhau. Kibana, giao diện người dùng đồ họa, là sự bổ sung cuối cùng, tạo nên ELK Stack như ngày nay.



Hình 2.1: Mô hình hệ thống ELK Stack

Tầm quan trọng của ELK Stack thể hiện rõ trong việc quản lý log một cách hiệu quả, giúp doanh nghiệp theo dõi và hiểu rõ về sự kiện trong hệ thống. Trong khi Elasticsearch đảm bảo khả năng lưu trữ và truy xuất dữ liệu log nhanh chóng, mang lại tính linh hoạt cho việc quản trị và phân tích dữ liệu, Logstash chịu trách nhiệm thu thập và chuẩn hóa dữ liệu log. Mặt khác, Kibana với giao diện đồ họa giúp trực quan hóa dữ liệu, từ biểu đồ đến bảng điều khiển, giúp nhận diện vấn đề và xu hướng một cách dễ dàng.

Đặc biệt, ELK Stack không chỉ đóng vai trò trong lĩnh vực quản lý log, mà còn có ứng dụng rộng rãi trong giám sát hệ thống, an ninh thông tin, và quản lý sự kiện. Sự tích

hợp và linh hoạt của nó làm cho ELK Stack trở thành công cụ đa nhiệm, đáp ứng nhu cầu ngày càng đa dạng của doanh nghiệp.

Bên cạnh đó, ELK Stack nhận được sự hỗ trợ mạnh mẽ từ cộng đồng người dùng và nhà phát triển trên khắp thế giới, đảm bảo rằng nó không chỉ đáp ứng hiệu quả với nhu cầu hiện tại mà còn luôn tiếp tục phát triển để đổi mới với các thách thức mới trong quản lý log và dữ liệu. Tóm lại, ELK Stack không chỉ là công cụ quản lý log, mà là một trợ thủ đáng tin cậy, giúp doanh nghiệp “đàm phán được con đường thông tin” của mình một cách hiệu quả và thông minh.

Cơ bản các thành phần chính của ELK Stack gồm:

- **Elasticsearch:** Là thành phần chính chịu trách nhiệm lưu trữ và tìm kiếm dữ liệu log trong ELK Stack.
- **Logstash:** Là công cụ thu thập, xử lý và chuyển đổi dữ liệu trước khi gửi đến Elasticsearch. Logstash có thể thu thập dữ liệu từ nhiều nguồn khác nhau và xử lý theo nhiều cách khác nhau trước khi truyền đến Elasticsearch.
- **Kibana:** Là công cụ trực quan hóa dữ liệu (cung cấp giao diện), giúp người dùng xem và phân tích dữ liệu log một cách trực quan thông qua các biểu đồ, bảng và dashboard.
- **Beats:** Là thành phần được bổ sung sau để tối ưu việc thu thập dữ liệu từ các nguồn khác nhau và gửi dữ liệu đến Elasticsearch hoặc Logstash. Chúng tối ưu hóa việc thu thập dữ liệu cho ELK Stack, giúp giám sát hạ tầng dễ dàng và hiệu quả hơn.

2.1.2 Ưu điểm và nhược điểm

ELK Stack là một nền tảng mã nguồn mở mạnh mẽ, được thiết kế để quản lý và phân tích log một cách hiệu quả.

- **Mã nguồn mở:** Với tính chất mã nguồn mở, ELK không chỉ tiết kiệm chi phí mà còn sở hữu cộng đồng người dùng lớn, tài liệu phong phú, và khả năng tùy chỉnh linh hoạt, đáp ứng đa dạng nhu cầu của doanh nghiệp.
- **Tính mở rộng cao:** cho phép xử lý dữ liệu lớn và đáp ứng nhu cầu ngày càng tăng về giám sát hệ thống trong các môi trường hạ tầng phức tạp.
- **Tìm kiếm và phân tích mạnh mẽ:** Elasticsearch mang lại khả năng tìm kiếm nhanh chóng và phân tích dữ liệu thời gian thực, giúp phát hiện lỗi và các bất thường một cách hiệu quả.
- **Quản lý log tập trung:** ELK Stack cung cấp khả năng tập hợp log từ nhiều nguồn, giúp quản lý tập trung và loại bỏ sự phân tán log, hỗ trợ giám sát toàn diện.

- **Giao diện trực quan:** Với các dashboard hiện đại và sinh động từ Kibana, giúp người dùng dễ dàng theo dõi tình trạng hệ thống, ngay cả khi không chuyên về kỹ thuật.

Tuy nhiên, việc quản lý một cụm ELK Stack lớn có thể trở nên rất phức tạp, đặc biệt là đối với người mới sử dụng còn ít kinh nghiệm.

- Để quản lý hiệu quả và duy trì các thành phần của ELK Stack đòi hỏi sự hiểu biết sâu rộng và các kỹ năng chuyên sâu. ELK Stack yêu cầu tài nguyên đáng kể, đặc biệt là Elasticsearch, từ đó phải tốn chi phí triển khai và duy trì.
- Nếu tài nguyên hạn chế, khả năng mở rộng của nó có thể sẽ gặp thách thức dẫn đến hiệu suất của tổng thể hệ thống.
- Ngoài ra về vấn đề bảo mật, người dùng cần phải đặc biệt chú ý đến cấu hình bảo mật để đảm bảo an toàn cho dữ liệu log. Nếu không sẽ dễ tạo ra các lỗ hổng bảo mật mà có thể bị tận dụng bởi bên thứ ba.
- Đồng thời, việc sử dụng hiệu quả ELK Stack đòi hỏi kiến thức sâu sắc về các công nghệ và khái niệm liên quan. Điều này đòi hỏi người dùng phải đầu tư thời gian và nỗ lực để hiểu rõ về cách ELK Stack hoạt động và làm thế nào để tối ưu hóa sự tích hợp của nó vào môi trường hệ thống cụ thể.

Tóm lại, ELK Stack là một công cụ mạnh mẽ, nhưng việc triển khai và duy trì nó đòi hỏi sự chuyên sâu và quản lý cẩn thận từ phía người sử dụng.

2.1.3 Úng dụng của ELK Stack và một số trường hợp sử dụng

ELK Stack là một công cụ mạnh mẽ trong việc giám sát hiệu suất của trang web, giúp các doanh nghiệp nắm bắt kịp thời tình trạng hệ thống. Nền tảng này cho phép theo dõi thời gian phản hồi của các dịch vụ, tỷ lệ lỗi phát sinh, và mức độ sử dụng tài nguyên như CPU, RAM, và băng thông mạng. Nhờ phân tích log chi tiết, các doanh nghiệp có thể xác định được giờ cao điểm truy cập và lên kế hoạch tối ưu hóa hoặc triển khai các chương trình khuyến mãi phù hợp, từ đó cải thiện trải nghiệm người dùng và duy trì hiệu suất hệ thống.

Trong môi trường hạ tầng phức tạp, ELK Stack hỗ trợ phát hiện sớm các sự cố, đảm bảo tính ổn định cho các hệ thống lớn với nhiều dịch vụ hoặc microservices. Công cụ này cung cấp khả năng giám sát log từ các máy chủ, container, hoặc ứng dụng phân tán, đồng thời gửi cảnh báo tức thời khi phát hiện các vấn đề như bottleneck hay server downtime. Việc ứng dụng ELK giúp doanh nghiệp tránh được tình trạng gián đoạn dịch vụ, bảo vệ danh tiếng và sự hài lòng của khách hàng.

Về bảo mật, ELK Stack đóng vai trò quan trọng trong việc phát hiện và ứng phó với các mối đe dọa. Nền tảng này có thể theo dõi các hoạt động bất thường như IP lạ truy

cập, đăng nhập không hợp lệ, hoặc các cuộc tấn công brute force. Thông qua việc phân tích log bảo mật, doanh nghiệp có thể phát hiện và ngăn chặn sớm các nỗ lực truy cập trái phép, từ đó giảm thiểu nguy cơ bị tấn công mạng và bảo vệ dữ liệu quan trọng.

Ngoài ra, ELK Stack còn hỗ trợ ghi nhận và phân tích log bảo mật để điều tra các sự cố sau khi xảy ra. Hệ thống lưu lại các nỗ lực truy cập trái phép, log đăng nhập không thành công, hoặc các hành vi vi phạm chính sách. Từ các dữ liệu này, doanh nghiệp có thể tối ưu hóa chính sách bảo mật, cải thiện firewall, và tăng cường các biện pháp bảo vệ hệ thống, đảm bảo an toàn thông tin lâu dài.

Cuối cùng, ELK Stack giúp phân tích log ứng dụng, một phần quan trọng trong việc phát triển và duy trì các ứng dụng hiện đại. Thông qua phân tích log chi tiết, doanh nghiệp có thể nhanh chóng tìm và khắc phục lỗi runtime, tối ưu hóa các truy vấn cơ sở dữ liệu, và cải thiện hiệu suất tổng thể của ứng dụng. Điều này không chỉ giúp nâng cao chất lượng dịch vụ mà còn tối ưu hóa trải nghiệm người dùng một cách đáng kể.

2.2 Các thành phần trong ELK Stack

2.2.1 Elasticsearch

Khái niệm và đặc điểm

Elasticsearch (ES) là một hệ thống cơ sở dữ liệu tìm kiếm mạnh mẽ trong việc xử lý, tìm kiếm thông tin đối với các ứng dụng hiện đại. Elasticsearch không chỉ là một công cụ tìm kiếm mà còn là một nền tảng có khả năng lưu trữ và truy xuất dữ liệu ở tốc độ cao hiệu quả. Các đặc điểm nổi bật của Elasticsearch có thể kể đến như:

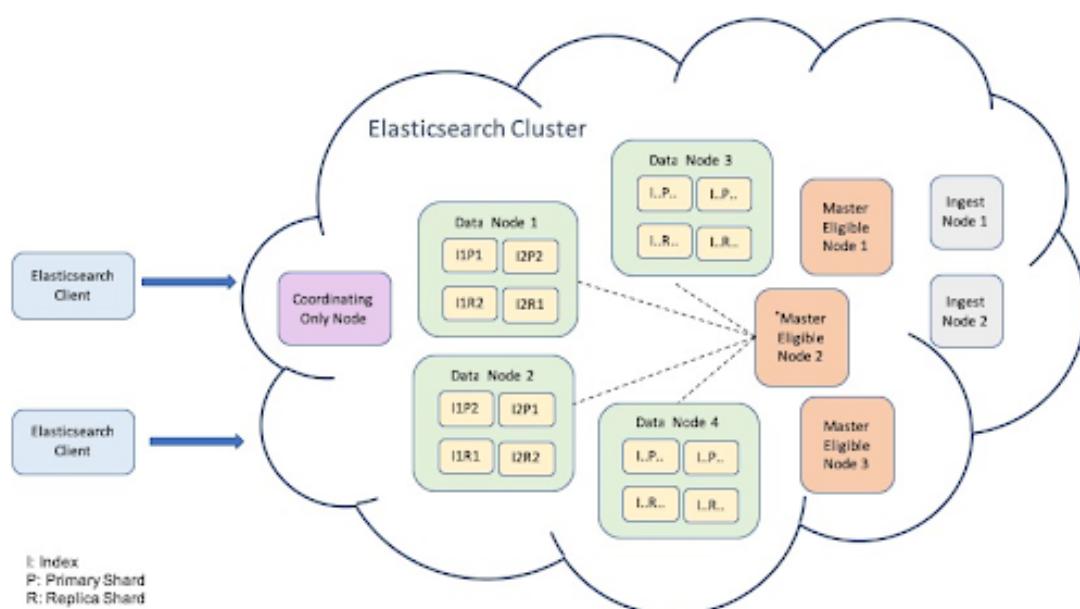
- **Công cụ tìm kiếm tối ưu hơn truy vấn dữ liệu SQL:** Nếu search bằng truy vấn SQL như LIKE %one% thì kết quả sẽ chỉ cần chứa one là ra, như phone, zone, money... dẫn đến nhiều kết quả không mong muốn. Còn khi search bằng ES thì gõ one sẽ chỉ có one được trả về. Truy vấn LIKE không thể truy vấn có dấu,... ES sử dụng JSON-based DSL(Domain Specific Language) để xây dựng các truy vấn. Truy vấn này có thể tìm kiếm theo các điều kiện, sắp xếp hay là lọc. Kết quả trả về sẽ được xếp hàng dựa theo độ liên quan với truy vấn dựa trên thuật toán TF-IDF (Term Frequency-Inverse Document Frequency) và các yếu tố khác;
- **Phân tích dữ liệu:** ES có khả năng phân tích dữ liệu theo thời gian với Time Series Analytics, hay phân tích hồi quy, xu hướng và mô hình hóa dữ liệu với Machine Learning. Nhờ đó người dùng có thể dễ dàng phân tích và hiểu rõ insight từ dữ liệu lớn để hỗ trợ đưa ra quyết định;
- **Phân tán dữ liệu:** Các node trong cluster liên lạc với nhau để chia sẻ thông tin về dữ liệu, trạng thái của cluster. ES sử dụng các giải thuật như Zen Discovery để phát hiện và quản lý các nút trong mạng lưới. Cluster cũng có thể sẵn sàng chịu

lỗi vì có các bản replicas của các node được phân bố trong cluster và thay thế lúc cần thiết mà không làm gián đoạn công việc;

- **Lưu trữ dữ liệu:** Khi một tài liệu được thêm mới vào ES, nó được đánh index để cho phép tìm kiếm nhanh chóng. Quá trình này bao gồm việc tách văn bản thành các từ và lưu trữ chúng trong một cấu trúc tối ưu cho việc tìm kiếm. Dữ liệu cũng sẽ được chia thành các phân vùng để phân tán dữ liệu trên nhiều node từ đó giúp tăng khả năng mở rộng và tăng hiệu suất;

Cách tổ chức dữ liệu

Cách tổ chức dữ liệu của Elasticsearch có thể chia thành những bộ phận sau:



Hình 2.2: Cách tổ chức dữ liệu của Elasticsearch

Cluster: Trong ES, mỗi một cluster là một tập các Node, cluster chứa toàn bộ dữ liệu đồng thời cung cấp khả năng đánh index và tìm kiếm trên toàn bộ các node thuộc cluster đó. Mỗi cluster được định danh bởi 1 tên duy nhất. Clustering cho phép sử dụng nhiều máy chủ kết hợp với nhau tạo thành 1 cụm có khả năng chịu đựng hay chấp nhận lỗi để nâng cao tính sẵn sàng. Nếu một máy chủ ngừng hoạt động do sự cố hoặc để nâng cấp hay bảo trì thì toàn bộ công việc do máy chủ này hoạt động hoàn toàn có thể chuyển sang máy chủ khác mà không làm cho hệ thống bị ngắt hay gián đoạn.

Node: là một single server, tham gia vào quá trình đánh index và tìm kiếm của cluster. Cũng giống như cluster mỗi node được định danh bởi một tên và được sinh ngẫu nhiên

tại thời điểm khởi động hệ thống. Mỗi node chỉ có thể thuộc 1 cluster do nhất được chỉ định, nếu không nó sẽ chỉ thuộc elastics cluster được tạo tự động từ đầu.

Indexing: một tập hợp các document có cấu trúc tương tự nhau, và mỗi tài liệu bao gồm các trường (fields) và giá trị tương ứng. Mỗi tài liệu lưu trữ một đơn vị dữ liệu như một bản ghi hay một đối tượng. Elasticsearch cho phép tìm kiếm và trả về kết quả cực nhanh bởi thay vì tìm kiếm text 1 cách thủ công thì ES sử dụng inverted index. Nghĩa là thay vì đọc lần lượt để tìm kiếm thì ES sẽ tìm kiếm dựa trên các keywords trong các index và việc tạo và quản lý inverted index được xử lý bởi search engine Apache Lucene.

Document: Là đơn vị dữ liệu được tìm kiếm và đánh index. Một index có thể bao gồm 1 hay nhiều document, 1 Document có thể chứa 1 hay nhiều Fields và được lưu dưới dạng JSON. Trong ES, thì 1 document không cần phải có 1 cấu trúc rõ ràng không giống như 1 DB thông thường.

Shards và Replicas: Một index có thể được chia nhỏ thành các mảnh khác nhau và phân tán trên các node khác nhau được gọi là shards. Khi tạo một index, quản trị viên có thể cấu hình số lượng shards để lưu trữ index này. Mỗi shards, bản thân nó là 1 index đầy đủ chức năng và độc lập, do đó chúng có thể được host bởi bất kỳ node nào. Việc làm như vậy có thể giúp ES có khả năng horizontal scale và tính toán phân tán và song song đồng thời giúp tăng hiệu năng. Và replicas chính là bản sao của shards. Replicas được tạo ra với mục đích tăng tính sẵn sàng và khả năng chịu lỗi của hệ thống. Bởi nếu 1 shard bị lỗi thì có thể sử dụng bản copy của nó để thay thế ngay lập tức.

Mapping là quá trình định nghĩa làm thế nào một document và các fields của nó được lưu trữ và đánh index. Có thể nói đến như fields này có kiểu dữ liệu là gì, hay liệu giá trị của tất cả các fields trong document được đánh index hay không.

Mapping type được sử dụng để định nghĩa cấu trúc dữ liệu của các document khi lưu chúng vào 1 index. Mỗi mapping type định nghĩa cấu trúc dữ liệu riêng, bao gồm các fields, kiểu dữ liệu và 1 số thuộc tính khác. Ở phiên bản mới nhất, khi tạo mới 1 index, ES sẽ tự động định nghĩa một mapping type tương ứng cho index đó, và tất cả các document được thêm vào index đều phải tuân theo mapping type này. Điều này giúp ES đơn giản hóa cấu trúc dữ liệu và giảm bớt sự phức tạp cũng như tối ưu hóa được performance cho hệ thống

Cơ chế hoạt động

Với việc dữ liệu gửi đến ES được lưu dưới dạng các document JSON được đánh index và phân bố trên các node khác nhau thì khi cần tìm kiếm người dùng dùng có thể cần gửi 1 HTTP request đến ES. HTTP request sẽ bao gồm từ khóa tìm kiếm, cơ chế tìm kiếm, hay là các tham số tìm kiếm. Từ đó truy vấn sẽ được gửi tới 1 node đầu tiên. Node đầu tiên này sẽ chuyển truy vấn tới tất cả các node có chứa các shards liên quan đến index được truy vấn. Mỗi node nhận được truy vấn sẽ thực thi truy vấn đối với các mảnh dữ liệu ở trên node đó và kết quả trả về từ các node sẽ được gộp lại và sắp xếp theo độ liên quan và một số yếu tố khác để tạo nên tập kết quả cuối cùng. Kết quả trả về cũng sẽ được

cached lại cho các lần truy vấn sau.

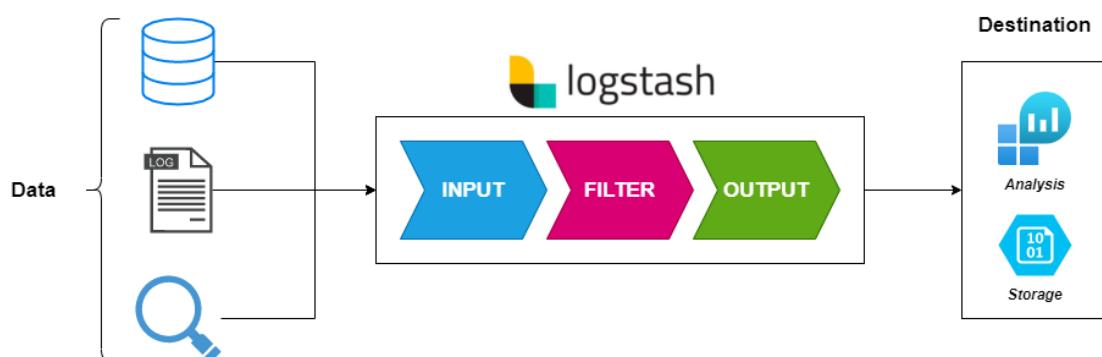
2.2.2 Logstash

Khái niệm và đặc điểm

Logstash được sử dụng để tạo cấu trúc cho log, giúp tối ưu hóa khả năng tìm kiếm và trực quan hóa dữ liệu. Một trong những đặc điểm nổi bật của Logstash là khả năng xử lý đa dạng các loại dữ liệu. Với hơn 200 plugin, Logstash là một công cụ linh hoạt, cho phép người dùng thu thập dữ liệu từ nhiều nguồn khác nhau một cách dễ dàng. Khả năng này mang lại cho người dùng không chỉ sức mạnh trong việc thu thập thông tin mà còn khả năng chuyển đổi và tinh chỉnh dữ liệu nhằm đáp ứng đúng yêu cầu và mục đích phân tích cụ thể.

Cơ chế hoạt động

Luồng dữ liệu đi qua Logstash sẽ được xử lý theo cơ chế pipeline.



Hình 2.3: Luồng hoạt động của Logstash

Pipeline gồm có 3 stage (giai đoạn): input → filter → output. Input: Trong stage này, dữ liệu từ các source sẽ được Logstash chuyển thành các event. Mỗi một input stage sẽ chạy bên trong thread riêng của mình, và ghi các event vào trong một queue trung tâm nằm trong bộ nhớ (mặc định) hoặc ổ cứng. Trong queue này, các event sẽ được đóng gói thành các batch.

Các Input Plugin quan trọng và phổ biến nhất được sử dụng là: Files, Beats, Syslog, HTTP. Trong đó:

File Plugin: File Plugin là một công cụ mạnh mẽ để theo dõi và thu thập dữ liệu từ các tệp tin log. Khả năng tự động cập nhật dữ liệu mới khi có thay đổi trong tệp tin giúp Logstash duy trì sự đồng bộ với các thông tin mới nhất từ hệ thống. Điều này làm cho

File Plugin trở thành một lựa chọn lý tưởng cho việc theo dõi và thu thập dữ liệu từ các tệp log trên hệ thống.

Beats Plugin: Beats Plugin là một trong những công cụ quan trọng giúp Logstash tích hợp dễ dàng với các Beats agent. Được thiết kế nhẹ nhàng, Beats giảm bớt gánh nặng và đơn giản hóa quá trình thu thập dữ liệu từ các máy chủ, container và các nguồn dữ liệu phân tán khác. Sự linh hoạt và khả năng mở rộng của Beats Plugin khiến cho nó phù hợp với việc thu thập dữ liệu từ các môi trường đa dạng.

Syslog Plugin: Syslog Plugin là một công cụ chuyên dụng để thu thập dữ liệu từ các thiết bị hoạt động theo chuẩn syslog. Với khả năng tương thích với nhiều thiết bị mạng như Router và Switch, Syslog Plugin là sự lựa chọn tốt để giám sát và thu thập dữ liệu từ chúng.

HTTP Plugin: HTTP Plugin cung cấp khả năng lắng nghe trên một cổng HTTP và thu thập dữ liệu thông qua giao thức HTTP. Điều này làm cho nó trở thành công cụ hữu ích cho việc thu thập dữ liệu từ các nguồn web hoặc các ứng dụng tương tác qua HTTP, mở rộng khả năng tích hợp và tính linh hoạt của Logstash.

Filter: trong stage này, các worker thread sẽ lấy các batch ra khỏi queue. Sau đó tùy theo filter mà chỉnh sửa nội dung của các event.

Các Filter Plugin phổ biến được sử dụng nhiều hiện nay gồm có: Grok, Mutate, Date, JSON, GeoIP, KV.

Grok Filter Plugin: Grok cho đến nay là filter plugin được sử dụng phổ biến nhất trong Logstash. Nó là một công cụ mạnh mẽ để phân tích cú pháp và tách dữ liệu thành các trường riêng lẻ. Với các mẫu Grok, nó có thể nhận diện và phân tách các mẫu dữ liệu phức tạp, không có cấu trúc thành các phần nhỏ hơn, tạo ra cấu trúc cho dữ liệu log và làm cho nó trở nên dễ đọc và hiểu hơn.

Mutate Filter Plugin: Mutate Filter Plugin cho phép người dùng điều chỉnh các dữ liệu log bằng cách “biến đổi” các trường khác nhau. Ví dụ, người dùng có thể sử dụng bộ lọc này để thay đổi các trường, nối chúng lại với nhau, đổi tên chúng hay thậm chí là xóa chúng. Điều này tạo ra sự linh hoạt trong việc tinh chỉnh cấu trúc của dữ liệu và chuẩn hóa nó để phù hợp với các yêu cầu cụ thể.

Date Filter Plugin: Date Filter Plugin có thể được sử dụng để lấy ngày và giờ từ dữ liệu log và xác định nó làm trường “dấu thời gian” (timestamp) cho log. Bộ lọc này sẽ cung cấp khả năng chuyển đổi các trường thời gian trong dữ liệu thành định dạng chuẩn hoặc các trường thời gian khác nhau. Nó giúp cho việc đồng bộ và hiểu rõ thời gian trở nên thuận tiện, hỗ trợ rất nhiều trong quá trình phân tích và giám sát sự kiện.

JSON Filter Plugin: JSON là một định dạng cực kỳ phổ biến cho log vì nó cho phép người dùng viết các dữ liệu log được chuẩn hóa có cấu trúc để có thể dễ dàng đọc và phân tích. Để duy trì cấu trúc JSON của toàn bộ dữ liệu hoặc một trường cụ thể, JSON Filter Plugin cho phép người dùng trích xuất và duy trì cấu trúc JSON trong dữ liệu log. Ngoài ra, bộ lọc này còn giúp giải mã các trường JSON trong dữ liệu, tạo ra sự hiểu biết sâu sắc về cấu trúc và thông tin trong log.

GeoIP Filter Plugin: GeoIP Filter Plugin giúp mở rộng thông tin địa lý bằng cách thêm dữ liệu vị trí địa lý từ địa chỉ IP. Điều này làm cho việc phân tích log với khía cạnh địa lý trở nên phong phú, đặc biệt hữu ích khi theo dõi nguồn gốc và vị trí của các sự kiện.

KV Filter Plugin: Cặp giá trị-khoa hoặc KVP (Key Value Pair) là một định dạng ghi nhặt ký thường được sử dụng khác. Giống như JSON, định dạng này phổ biến vì nó dễ đọc và KV Filter Plugin cho phép người dùng tự động phân tích cú pháp các dữ liệu hoặc các trường cụ thể được định dạng theo cách này.

Output: trong stage này, các worker thread sẽ chuyển các event tới các destination. Dưới đây là mô tả về một số Output Plugin quan trọng trong Logstash:

File Output Plugin: File Output Plugin cho phép người dùng lưu trữ dữ liệu đã được xử lý vào các tệp tin log, tạo nên một giải pháp thuận tiện để giữ lại và quản lý thông tin log trực tiếp trên máy chủ hoặc hệ thống tệp tin.

CSV Output Plugin: CSV Output Plugin được sử dụng để xuất dữ liệu log dưới định dạng CSV (Comma-Separated Values), tổ chức dữ liệu thành các cột và hàng, thuận tiện cho việc nhập vào các bảng tính và các ứng dụng đòi hỏi định dạng CSV.

S3 Output Plugin: S3 Output Plugin cho phép người dùng đẩy dữ liệu log trực tiếp đến Amazon S3, mang lại sự linh hoạt và tiện ích trong việc lưu trữ lượng lớn dữ liệu log trên nền tảng đám mây.

Stdout Output Plugin: Stdout Output Plugin hiển thị dữ liệu xử lý trực tiếp trên cửa sổ console, giúp người dùng có thể theo dõi thông tin log ngay tại thời điểm xử lý, đặc biệt hữu ích trong quá trình phát triển và kiểm thử.

Logstash Codecs: Logstash Codecs đóng vai trò quan trọng trong quá trình xử lý dữ liệu, mang lại khả năng định dạng và mã hóa thông tin log để làm cho nó dễ đọc và tương thích với các hệ thống khác nhau. Trong khi Input Codec hỗ trợ để giải mã dữ liệu một cách thuận tiện trước khi dữ liệu được đưa vào Input Plugin thì Output Codec là một phương tiện để mã hóa dữ liệu trước khi nó rời khỏi Output Plugin. Sau đây là một số codecs phổ biến nhất trong Logstash:

Plain codec: Plain codec là một lựa chọn đơn giản nhưng hiệu quả, giữ nguyên định dạng ban đầu của thông điệp log mà không thực hiện bất kỳ biến đổi hay mã hóa nào. Điều này làm cho nó phù hợp với các dữ liệu log đã ở định dạng chuẩn và không cần sự thay đổi.

JSON codec: JSON codec được dùng để mã hóa các sự kiện JSON ở đầu vào và giải mã các thông báo json ở đầu ra. Tuy nhiên, nếu dữ liệu nhận được không ở một định dạng JSON hợp lệ thì JSON codec sẽ trả về định dạng ban đầu.

JSON Lines codec: JSON Lines codec chuyển đổi mỗi sự kiện log thành một dòng trong định dạng JSON Lines. Điều này giúp tách biệt mỗi sự kiện và thuận tiện cho việc xử lý dữ liệu hàng loạt, đặc biệt là trong các tình huống với lượng lớn sự kiện log.

Rubydebug codec: Rubydebug codec xuất dữ liệu log dưới định dạng đối tượng dữ liệu Ruby (data Ruby objects) được định nghĩa bởi Ruby's pp (pretty print), giúp hiển

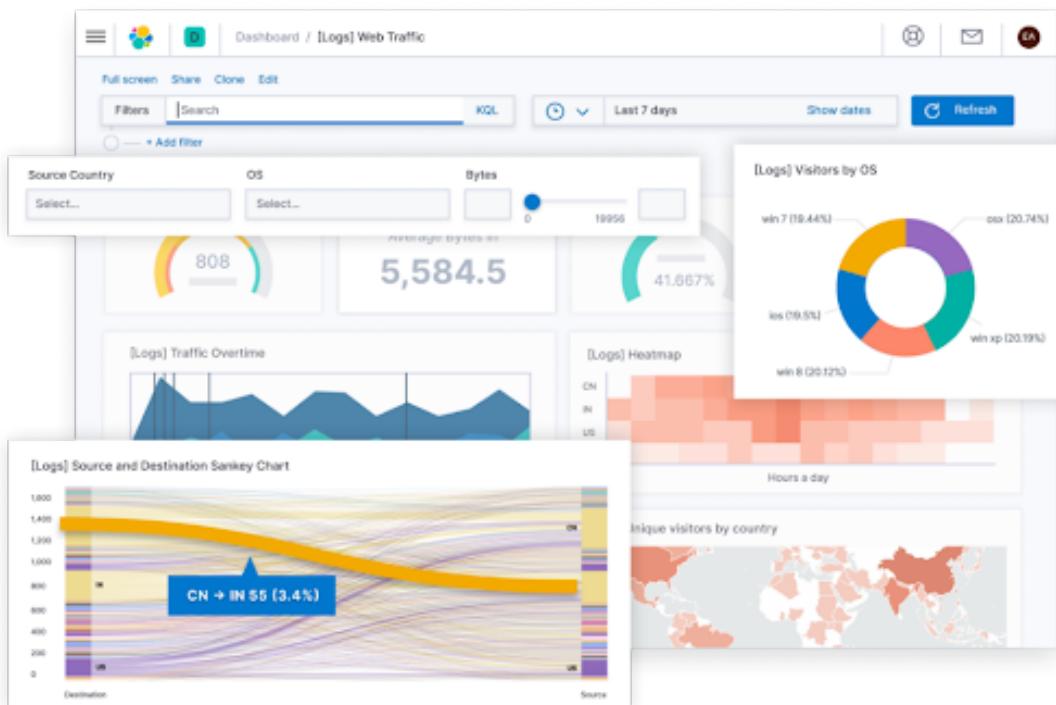
thị thông tin log một cách có cấu trúc và dễ đọc. Rubydebug codec thường được sử dụng trong quá trình phát triển và kiểm thử để dễ dàng theo dõi và kiểm tra cấu trúc dữ liệu log.

Sự lựa chọn và kết hợp linh hoạt của các codecs này giúp Logstash trở thành một công cụ mạnh mẽ trong việc chuyển đổi sự kiện log, đáp ứng linh hoạt với nhiều định dạng và cấu trúc dữ liệu khác nhau.

2.2.3 Kibana

Khái niệm và đặc điểm

Kibana là một phần mềm chạy trên nền tảng web được sử dụng để tìm kiếm, phân tích và trực quan hóa dữ liệu được lưu trữ trong Elasticsearch (Kibana không thể được sử dụng với các DB khác mà chỉ được phát triển để sử dụng với Elasticsearch). Nó được biết đến như 1 công cụ nổi tiếng vì cung cấp các công cụ trực quan hóa mạnh mẽ trên nền tảng dữ liệu lớn bao gồm các biểu đồ, bảng, hay hơn nữa là bản đồ cho phép người dùng tạo các bản đồ để trực quan hóa dữ liệu theo vị trí địa lý



Hình 2.4: Giao diện của Kibana

Cơ chế hoạt động

Kibana sẽ phải dùng kết hợp với Elasticsearch:

- Khi người dùng thực hiện truy vấn, Kibana gửi yêu cầu đến Elasticsearch để lấy dữ liệu cần thiết.
- Sau khi nhận được dữ liệu từ Elasticsearch, Kibana cho phép người dùng tạo ra nhiều loại biểu đồ khác nhau như biểu đồ cột, biểu đồ đường, biểu đồ tròn và bản đồ nhiệt.

Các tính năng và thao tác chính

Một số tính năng nổi bật của Kibana gồm có:

Discover: Cho phép người dùng tìm kiếm và lọc dữ liệu từ các chỉ mục Elasticsearch nhanh chóng và hiệu quả. Người dùng có thể thực hiện tìm kiếm toàn văn trên dữ liệu, lọc kết quả bằng cách sử dụng các truy vấn KQL, xem chi tiết các trường dữ liệu.

Visualize: Tận dụng sức mạnh của dữ liệu bằng cách tạo ra nhiều loại biểu đồ, bảng biểu và hình thức trực quan khác nhau. Linh hoạt trong việc lựa chọn kiểu hiển thị dữ liệu. Dễ dàng lựa chọn hình thức phù hợp nhất để diễn đạt ý nghĩa của dữ liệu đang phân tích. Khả năng tùy chỉnh từng yếu tố của trực quan hóa, từ màu sắc, đến kích thước và loại hình.

Dashboard: Kết hợp nhiều trực quan hóa thành một giao diện tổng quát, giúp dễ dàng theo dõi và phân tích. Cho phép người dùng có cái nhìn tổng quát, từ đó nhanh chóng phát hiện các xu hướng hoặc mẫu hình nổi bật. Cung cấp khả năng tùy chỉnh cao, người dùng có thể điều chỉnh dữ liệu hiển thị, từ việc thay đổi phạm vi thời gian đến việc tùy chỉnh các bộ lọc.

Canvas: Cho phép người dùng tạo ra các báo cáo và bản trình bày với thiết kế nghệ thuật. Người dùng có thể kéo thả các thành phần, thêm văn bản, hình ảnh và kết nối với dữ liệu để tạo ra sản phẩm có các thông tin cần thiết và đẹp mắt. **Machine Learning:** Hỗ trợ người dùng phát hiện ra các vấn đề tiềm ẩn nhanh chóng mà đôi khi con người khó nhận ra. Định dạng và phát hiện các mẫu trưởng đột biến, xu hướng giảm giá trị, hay các trường hợp ngoại lệ.

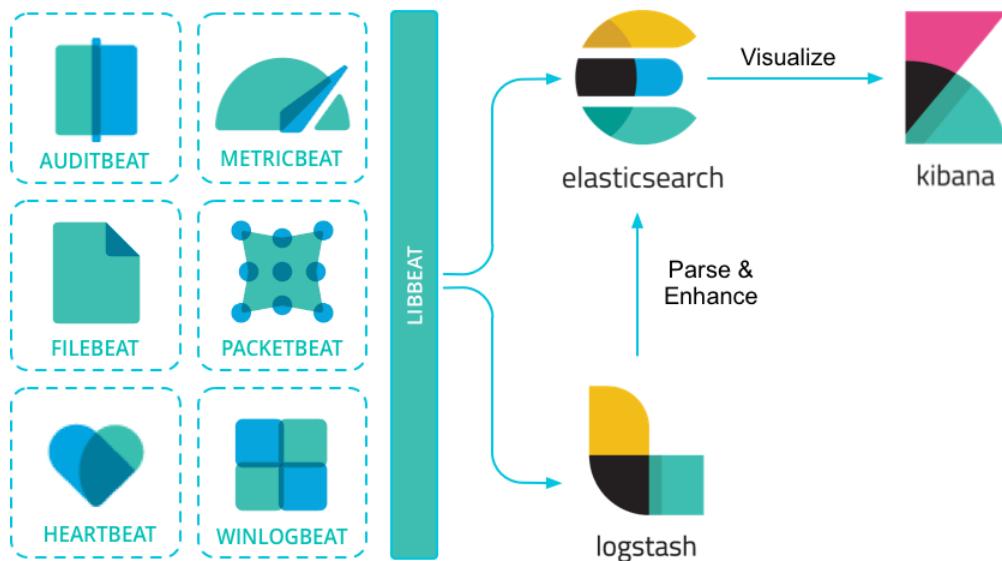
2.2.4 Beats

Khái niệm và đặc điểm

Beats là những đối tác nhẹ (lightweight agents) được cài đặt trực tiếp trên các máy chủ vận hành (edge hosts) để thu thập các loại dữ liệu khác nhau, sau đó chuyển tiếp chúng vào ELK Stack. Beats hoạt động như những "người giao hàng dữ liệu" (log shippers) được tối ưu hóa để đạt hiệu suất tối đa mà vẫn giữ một lợi thế về cài đặt nhỏ, tiết kiệm tài nguyên và không yêu cầu bất kỳ phụ thuộc nào.

Các loại Beats phổ biến

Mỗi Beats khác nhau thì sẽ thu thập được những loại dữ liệu khác nhau.



Hình 2.5: Các loại Beats

Filebeat: Dùng để thu thập log từ các file log của hệ thống, ứng dụng, dịch vụ, và chuyển đến Logstash hoặc Elasticsearch. Đây là lựa chọn lý tưởng để theo dõi các file log liên tục thay đổi.

Metricbeat: Thu thập các số liệu (metrics) từ hệ thống, bao gồm CPU, bộ nhớ, I/O, và nhiều chỉ số khác từ hệ thống và ứng dụng, giúp giám sát hiệu suất và tình trạng sức khỏe của hệ thống.

Packetbeat: Thu thập thông tin về lưu lượng mạng, giúp phân tích các giao dịch, hiệu suất mạng và phát hiện các vấn đề tiềm ẩn về kết nối và bảo mật.

Auditbeat: Thu thập dữ liệu về bảo mật và kiểm tra hệ thống (audit data), giúp giám sát các sự kiện an ninh, như các thay đổi quyền truy cập và hoạt động người dùng.

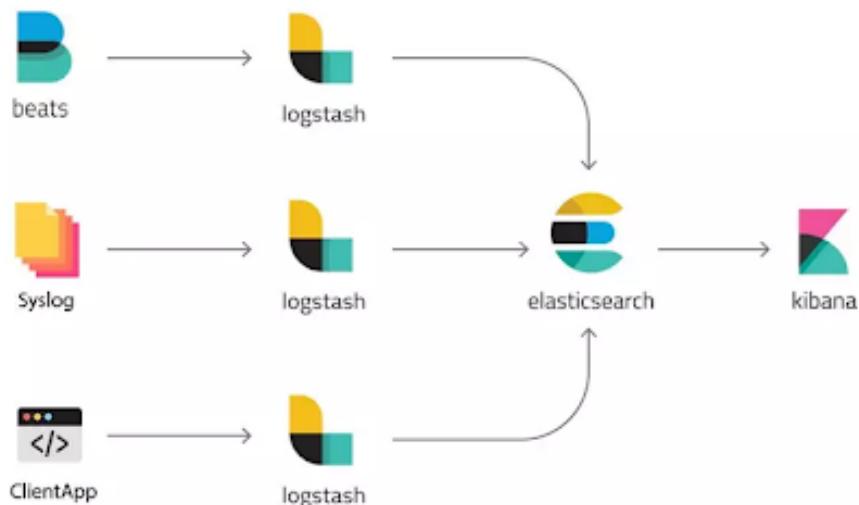
Heartbeat: Kiểm tra trạng thái của các dịch vụ và điểm cuối (endpoint) để phát hiện và cảnh báo khi một dịch vụ bị gián đoạn hoặc gặp vấn đề về hiệu suất.

Winlogbeat: Thu thập log từ hệ thống Windows Event Logs, giúp phân tích các sự kiện và lỗi của hệ điều hành Windows.

2.3 Vận hành hệ thống ELK Stack

2.3.1 Cách thức và hoạt động của các thành phần

Logstash tiếp nhận dữ liệu log được đưa đến thông qua nhiều hình thức khác nhau như Server gửi 1 request UDP chứa log đến URL của Logstash hay Beats đọc files logs sau đó gửi đến cho Logstash... Sau khi đọc logs gửi về, logstash sẽ thêm một số thông tin về IP, thời gian, parse từ dữ liệu log (server, level cảnh báo, nội dung,...) rồi ghi log xuống database là Elasticsearch. Sau khi log được lưu xuống Elasticsearch thì Kibana sẽ tiến hành đọc những logs này và hiển thị ra giao diện để người dùng có thể xem, đưa truy vấn và xử lý dữ liệu.



Hình 2.6: Luồng hoạt động của Docker Compose

2.3.2 Cài đặt và cấu hình ELK Stack

Trước khi cài đặt Elastic Stack lên app thì ta phải đảm bảo các cổng sau được mở:

Lưu ý là các thành phần tích hợp của Elastic cũng có cổng và dependencies riêng cho nên ta cũng cần kiểm chứng trước khi cài đặt.

Cài đặt và cấu hình Elasticsearch

Các thành phần của Elasticsearch không có sẵn trong các package mặc định của Ubuntu. Tuy nhiên, có thể cài đặt với APT bằng cách thêm danh sách nguồn gói của Elastic vào trình quản lý của Ubuntu.

Cổng	Thành phần
3002	Enterprise Search
5044	Elastic Agent → Logstash Beats → Logstash
5601	Kibana Elastic Agent → Fleet Fleet Server → Fleet
8220	Elastic Agent → Fleet Server APM Server
9200-9300	Elasticsearch REST API
9300-9400	Elasticsearch node transport and communication
9600-9700	Logstash REST API

Bảng 2.1: Các cổng và thành phần tương ứng trong hệ thống Elastic Stack

Trước tiên, ta sẽ nhập khóa GPG công khai của Elasticsearch và thêm nguồn gói Elastic để cài đặt Elasticsearch

```
curl -fsSL https://artifacts.elastic.co/GPG-KEY-elasticsearch | sudo apt key add -
```

Thêm list nguồn của Elastic vào thư mục source.list.d để APT tìm nguồn:

```
echo "deb https://artifacts.elastic.co/packages/7.x/apt stable main" | sudo tee -a /etc/apt/sources.list.d/elastic-7.x.list
```

Cập nhật danh sách gói nguồn để APT đọc nguồn mới:

```
sudo apt update
```

Sau đó ta mới cài được Elasticsearch:

```
sudo apt install elasticsearch
```

Như vậy là Elasticsearch đã được cài đặt và có thể được cấu hình qua file `elasticsearch.yml`. Các lựa chọn để cấu hình bao gồm cài đặt cụm (cluster), node, đường dẫn, bộ nhớ, ...

Elasticsearch mặc định dùng cổng 9200, để kiểm soát truy cập thì có thể cài đặt địa chỉ IP của host trong file cấu hình, ví dụ dùng localhost:

```
network.host: localhost
```

Để kích hoạt Elasticsearch và đặt cho nó khởi chạy mỗi khi server chạy thì ta dùng các lệnh:

```
sudo systemctl start elasticsearch
sudo systemctl enable elasticsearch
```

Để kiểm tra xem Elasticsearch có đang chạy hay không thì ta có thể gửi một request HTTP:

```
curl -X GET "localhost:9200"
```

Nếu Elasticsearch đang chạy thì ta sẽ nhận được một phản hồi chứa thông tin của node.

Cài đặt và cấu hình Kibana

Vì ta đã thêm gói nguồn của Elastic từ bước trên nên ta chỉ cần dùng APT để cài đặt các thành phần còn của Stack:

```
sudo apt install kibana
```

Kích hoạt và khởi chạy Kibana:

```
sudo systemctl enable kibana
sudo systemctl start kibana
```

Vì Kibana chỉ nghe trên localhost, nên nếu muốn cho phép truy cập từ bên ngoài thì chúng ta sẽ phải sử dụng reverse proxy.

Cài đặt và cấu hình Logstash

Ta cài đặt Logstash để xử lý dữ liệu:

```
sudo apt install logstash
```

Sau khi cài đặt thì có thể cấu hình Logstash qua các file trong thư mục /etc/logstash/conf.d/. Khi cấu hình các tệp thì lưu ý rằng Logstash luôn có 2 phần tử bắt buộc là input và output và một phần tử tùy chọn là filter. Các plugin input lấy dữ liệu từ nguồn, các plugin filter xử lý dữ liệu và các plugin output ghi dữ liệu vào đích.

Ví dụ, để cấu hình input Filebeat, ta tạo file cấu hình tên là 02-beats-input.conf trong thư mục file cấu hình:

```
sudo nano /etc/logstash/conf.d/02-beats-input.conf
```

Ta cài đặt để input Beats sẽ nghe cổng TCP 5044:

```
input {
  beats {
    port => 5044
  }
}
```

Ta cài đặt output để Logstash lưu dữ liệu của Beats vào Elasticsearch (chạy ở cổng 9200). Trước tiên tạo file cấu hình, ví dụ dùng Filebeat:

```
sudo nano /etc/logstash/conf.d/30-elasticsearch-output.conf
```

Và cài đặt file như sau:

```
output {
  if [@metadata][pipeline] {
    elasticsearch {
      hosts => ["localhost:9200"]
      manage_template => false
    }
  }
}
```

```

        index => "%{@metadata} [beat]-%{@metadata} [version]-%{+YYYY
                      .MM.dd}"
        pipeline => "%{@metadata} [pipeline]"
    }
} else {
    elasticsearch {
        hosts => ["localhost:9200"]
        manage_template => false
        index => "%{@metadata} [beat]-%{@metadata} [version]-%{+YYYY
                      .MM.dd}"
    }
}
}

```

Sau khi kiểm tra cấu hình thì ta kích hoạt và chạy Logstash:

```

sudo systemctl start logstash
sudo systemctl enable logstash

```

Cài đặt và cấu hình Filebeat

Cài đặt Filebeat:

```
sudo apt install filebeat
```

Filebeat được cấu hình bằng file `filebeat.yml`. Ta cài đặt để Filebeat gửi dữ liệu đến Logstash thay vì gửi đến Elasticsearch bằng cách đặt # trước dòng (comment):

```

# output.elasticsearch:
#   # Array of hosts to connect to.
#   hosts: ["localhost:9200:"]

```

Và cấu hình để Filebeat kết nối với Logstash:

```

output.logstash:
  # The Logstash hosts
  hosts: ["localhost:5044"]

```

Ta kích hoạt các module của Filebeat:

```
sudo filebeat modules enable system
```

Để xem các module của Filebeat, ta dùng lệnh:

```
sudo filebeat modules list
```

Sẽ thấy một danh sách có dạng như sau:

```

Enabled:
system

```

```

Disabled:
apache2
auditd
elasticsearch
icinga

```

```
iis  
kafka  
kibana  
logstash  
mongodb
```

Theo mặc định, thì filebeat đã được thiết lập để sử dụng các hướng dẫn mặc định cho việc thu thập nhật ký hệ thống và nhật ký ủy quyền vậy nên quản trị viên không cần cấu hình thêm.

Tiếp theo, ta thiết lập quy trình cho Filebeat:

```
sudo filebeat setup --pipelines --modules system
```

Sau đó ta chỉ cần kích hoạt và chạy Filebeat:

```
sudo systemctl start filebeat  
sudo systemctl enable filebeat
```

Như vậy là ta đã cài đặt và cấu hình Elastic Stack thành công. Giờ chỉ cần mở giao diện web của Kibana là có thể tương tác, xử lý dữ liệu trực quan.

2.3.3 Các vấn đề cần lưu ý khi vận hành hệ thống ELK Stack

Một số chú ý trong quá trình vận hành hệ thống ELK Stack có thể kể đến như:

Cấu hình phần cứng: ELK stack cần một hệ thống phần cứng đủ mạnh để xử lý khối lượng dữ liệu log lớn. Cụ thể, cần lưu ý đến các yếu tố sau:

- CPU: ELK stack sử dụng nhiều CPU để xử lý dữ liệu log. Do đó, cần sử dụng CPU có số lượng nhân và tốc độ xung nhịp cao.
- RAM: ELK stack cần nhiều RAM để lưu trữ dữ liệu log. Do đó, cần sử dụng RAM có dung lượng lớn.
- Ổ cứng: ELK stack sử dụng ổ cứng để lưu trữ dữ liệu log. Do đó, cần sử dụng ổ cứng có tốc độ đọc/ghi cao.

Cấu hình phần mềm: ELK stack có thể được cài đặt trên nhiều hệ điều hành khác nhau. Tuy nhiên, để đảm bảo hiệu suất và ổn định, nên cài đặt ELK stack trên hệ điều hành Linux.

Kết nối mạng: ELK stack cần có kết nối mạng ổn định để truyền tải dữ liệu log. Do đó, cần sử dụng mạng có băng thông cao và độ trễ thấp. Quản lý dữ liệu log: ELK stack có thể lưu trữ dữ liệu log trong thời gian dài. Tuy nhiên, cần có kế hoạch quản lý dữ liệu log hợp lý để tránh tình trạng ổ cứng bị đầy. Cụ thể, có thể sử dụng các phương pháp sau:

- Xóa các dữ liệu log cũ.
- Nén dữ liệu log.

- Sử dụng lưu trữ đám mây.

Bảo mật: ELK stack cần được bảo mật để ngăn chặn các truy cập trái phép vào dữ liệu log. Do đó, cần thiết lập các biện pháp bảo mật sau:

- Sử dụng xác thực hai yếu tố.
- Sử dụng tường lửa.
- Tài khoản quản trị có bảo mật mạnh.

Dưới đây là một số lưu ý cụ thể khi vận hành từng thành phần của ELK stack:

- **Elasticsearch:** Elasticsearch là một cơ sở dữ liệu phân tán có thể xử lý khối lượng dữ liệu lớn. Do đó, cần lưu ý đến các vấn đề về số lượng node Elasticsearch cần được xác định dựa trên khối lượng dữ liệu log cần xử lý và cấu hình node Elasticsearch cần đảm bảo đủ mạnh để xử lý khối lượng dữ liệu log. Và cuối cùng nên phân bổ dữ liệu log đều trên các node Elasticsearch để tránh tình trạng quá tải một node nào đó.
- **Logstash:** Logstash có thể được sử dụng để phân tích dữ liệu log. Cần xác định các quy tắc phân tích phù hợp để thu thập được các thông tin cần thiết từ dữ liệu log. Hơn nữa Logstash có thể xử lý dữ liệu log với kích thước lớn. Tuy nhiên, cần lưu ý đến dung lượng lưu trữ của hệ thống để tránh tình trạng ổ cứng bị đầy.
- **Kibana:** Cần thiết kế dashboard phù hợp với nhu cầu sử dụng.

Chương 3

Docker và Docker Compose

3.1 Docker

3.1.1 Giới thiệu chung

Khái niệm

Docker là một nền tảng mở để phát triển, vận chuyển, và chạy các ứng dụng. Nó giúp tách biệt ứng dụng khỏi cơ sở hạ tầng, cho phép phân phối phần mềm nhanh chóng và hiệu quả. Với Docker, việc quản lý cơ sở hạ tầng có thể thực hiện tương tự như cách bạn quản lý các ứng dụng của mình.

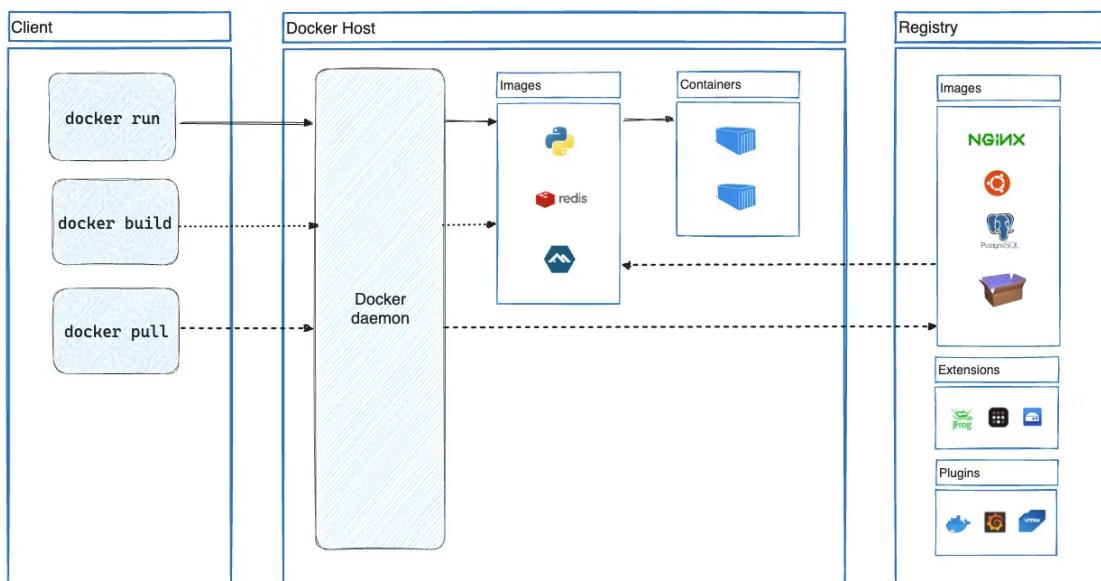
Cơ chế

Docker cung cấp khả năng đóng gói ứng dụng và các thành phần liên quan trong các container. Các container này cô lập ứng dụng và tài nguyên, đảm bảo an toàn và khả năng chạy đồng thời nhiều ứng dụng trên cùng một máy chủ.

Các thành phần của Docker

- Docker Engine: Công cụ chính dùng để tạo và quản lý container.
- Docker Hub: Kho lưu trữ công khai các Docker images.
- Docker Images: Khuôn mẫu chứa mã nguồn và các thư viện cần thiết để chạy ứng dụng.
- Docker Container: Phiên bản chạy của Docker Image.
- Docker Client: Giao diện dòng lệnh để tương tác với Docker Engine.

- Docker Daemon: Thành phần quản lý container, image và các đối tượng khác.
- Dockerfile: Tệp tin chứa các chỉ dẫn để tạo Docker Image.
- Docker Volumes: Phần dữ liệu được tạo ra trong quá trình container hoạt động.
- Docker Machine: Công cụ tạo Docker Engine trên máy chủ.
- Docker Compose: Công cụ cấu hình và chạy nhiều container từ file cấu hình.



Hình 3.1: Cơ chế và các thành phần của Docker.

3.1.2 Một số tính năng nổi bật

Docker đã trở thành một công cụ quan trọng trong việc phát triển và triển khai ứng dụng, nhờ vào khả năng tối ưu hóa quy trình làm việc và giải quyết các vấn đề thường gặp trong quản lý môi trường hệ thống, với một số tính năng chính nổi bật sau:

- **Phân phối ứng dụng nhanh chóng và nhất quán:** Docker giải quyết vấn đề thiết lập môi trường phát triển, kiểm thử và sản xuất bằng cách đóng gói toàn bộ ứng dụng và các phụ thuộc vào trong một container duy nhất. Điều này giúp đảm bảo tính nhất quán giữa các môi trường khác nhau, từ môi trường phát triển cho đến môi trường sản xuất. Thay vì phải thiết lập lại cấu hình từng máy chủ và môi trường, Docker giúp lập trình viên dễ dàng triển khai ứng dụng mà không lo lắng về các vấn đề tương thích giữa các hệ thống. Các ứng dụng trong container luôn

hoạt động giống nhau bất kể chúng chạy trên máy tính cá nhân của nhà phát triển hay trên môi trường sản xuất, giúp giảm thiểu rủi ro lỗi phát sinh do sự khác biệt môi trường.

- **Triển khai và mở rộng linh hoạt:** Docker hỗ trợ việc triển khai ứng dụng trên nhiều môi trường khác nhau, bao gồm máy tính cá nhân, máy chủ vật lý, hoặc các dịch vụ đám mây như AWS, Azure và Google Cloud. Điều này tạo ra một mức độ linh hoạt cao trong việc triển khai và quản lý ứng dụng. Docker cho phép các container di chuyển dễ dàng giữa các môi trường mà không gặp phải các vấn đề về cấu hình hoặc sự khác biệt hệ thống. Các container có thể được triển khai trong một môi trường cục bộ để thử nghiệm, và sau đó dễ dàng chuyển sang môi trường đám mây hoặc máy chủ vật lý mà không cần thay đổi cấu hình hay mã nguồn. Hơn nữa, Docker cũng hỗ trợ việc mở rộng linh hoạt, cho phép các ứng dụng có thể dễ dàng mở rộng quy mô bằng cách thêm hoặc giảm số lượng container mà không làm gián đoạn dịch vụ.
- **Tăng hiệu suất tài nguyên:** Docker có thể tối ưu hóa việc sử dụng tài nguyên phần cứng nhờ vào cơ chế nhẹ và nhanh hơn so với các máy ảo truyền thống. Trong khi các máy ảo yêu cầu phần cứng ảo hóa riêng biệt cho mỗi ứng dụng, Docker chia sẻ tài nguyên của hệ điều hành chủ, giúp giảm thiểu overhead và tăng hiệu quả sử dụng tài nguyên. Điều này không chỉ giúp tiết kiệm chi phí phần cứng mà còn cải thiện hiệu suất của hệ thống tổng thể. Docker cho phép chạy hàng chục, thậm chí hàng trăm container trên cùng một máy chủ vật lý mà không làm giảm hiệu suất đáng kể. Nhờ vào tính nhẹ nhàng và khả năng chia sẻ tài nguyên, Docker giúp các ứng dụng có thể chạy nhanh chóng và hiệu quả mà không gây ảnh hưởng lớn đến các hệ thống khác.

Tóm lại, Docker giúp các tổ chức triển khai ứng dụng nhanh chóng, dễ dàng mở rộng, và tối ưu hóa tài nguyên hệ thống. Với Docker, các doanh nghiệp có thể đảm bảo tính nhất quán giữa các môi trường phát triển, kiểm thử và sản xuất, đồng thời linh hoạt trong việc di chuyển và mở rộng ứng dụng. Bên cạnh đó, khả năng tiết kiệm tài nguyên của Docker giúp giảm chi phí phần cứng và tăng hiệu suất cho các hệ thống.

3.1.3 Ứng dụng thực tế

- **Độc lập với môi trường hệ thống:** Đảm bảo ứng dụng chạy đúng trên nhiều môi trường mà không lo ngại về sự không tương thích.
- **Hỗ trợ phát triển và kiểm thử:** Đóng gói ứng dụng cùng môi trường phát triển để giảm rủi ro lỗi.
- **Tích hợp CI/CD:** Docker tích hợp tốt với các quy trình phát triển liên tục và triển khai liên tục.

- **Quản lý microservices:** Dễ dàng triển khai các dịch vụ theo kiến trúc microservices.
- **Đa nền tảng và mở rộng:** Hỗ trợ triển khai trên nhiều hệ điều hành và môi trường khác nhau.
- **Kỹ thuật đám mây và DevOps:** Đơn giản hóa việc triển khai và quản lý ứng dụng trong môi trường đám mây.

3.2 Docker Compose

3.2.1 Giới thiệu chung

Khái niệm

Docker Compose là một công cụ mạnh mẽ dùng để định nghĩa và quản lý các container Docker, giúp thiết lập, cấu hình, và chạy các dịch vụ phụ thuộc lẫn nhau một cách dễ dàng. Một tệp Docker Compose thông thường có định dạng .yml hoặc .yaml, trong đó chúng ta định nghĩa các dịch vụ cần thiết cho ứng dụng. Dưới đây là tổng quan về cấu trúc của một tệp Docker Compose.

Cấu trúc cơ bản của Docker Compose

```

version: '3.8' #Phiên bản của Docker Compose
services:       # Khởi động dịch vụ, chứa định nghĩa của các container
  service_name: # Tên dịch vụ, ví dụ: web, db, redis
    image: image_name:tag # Tên và tag của image Docker
    build:          # Hoặc chỉ định thư mục chứa Dockerfile
    context: ./directory # Đường dẫn thư mục của Dockerfile
    dockerfile: Dockerfile-alternate # Tên tệp Dockerfile (nếu khác mặc định)
  ports:          # Cấu hình ánh xạ cổng host và container
    - "host_port:container_port"
  environment:    # Các biến môi trường của container
    - ENV_VAR_NAME=value
  volumes:         # Cấu hình volume để lưu trữ dữ liệu giữa các container
    - host_path:container_path
  networks:        # Khai báo mạng mà container này sẽ kết nối
    - network_name
  depends_on:      # Chỉ định dịch vụ phụ thuộc, để đảm bảo thứ tự khởi động
    - another_service
  restart: "always" # Cấu hình chế độ khởi động lại

  volumes:          # Khai báo các volumes chia sẻ dữ liệu giữa các container hoặc với host
    volume_name:
      driver: local # Chỉ định driver cho volume

  networks:         # Khai báo các networks, để các container có thể kết nối
    network_name:
      driver: bridge # Loại network (bridge là mặc định)

```

Giải thích từng thành phần:

- version: Phiên bản Docker Compose, phiên bản 3.8 là phổ biến hiện nay.
- services: Khối chính, chứa danh sách các dịch vụ (container) cần chạy. Mỗi dịch vụ sẽ được khai báo bằng:
 - image: Chỉ định tên image cần dùng từ Docker Hub hoặc custom image.
 - build: Thay vì sử dụng một image có sẵn, có thể build từ Dockerfile.
 - ports: Chỉ định ánh xạ cổng giữa máy host và container.
 - environment: Đặt biến môi trường cho dịch vụ, ví dụ như mật khẩu database, API keys.
 - volumes: Ánh xạ thư mục hoặc volume để chia sẻ dữ liệu.
 - networks: Chỉ định các mạng mà container có thể kết nối.
 - depends_on: Định nghĩa sự phụ thuộc giữa các dịch vụ, đảm bảo khởi động đúng thứ tự.
 - restart: Tùy chọn khởi động lại container nếu gặp lỗi, giúp tăng tính ổn định.
- volumes: Khởi khai báo volume, giúp lưu trữ dữ liệu giữa các lần khởi động container, tránh mất dữ liệu khi container bị xóa.
- networks: Khởi khai báo mạng, để các container có thể kết nối với nhau trong môi trường Docker Compose.

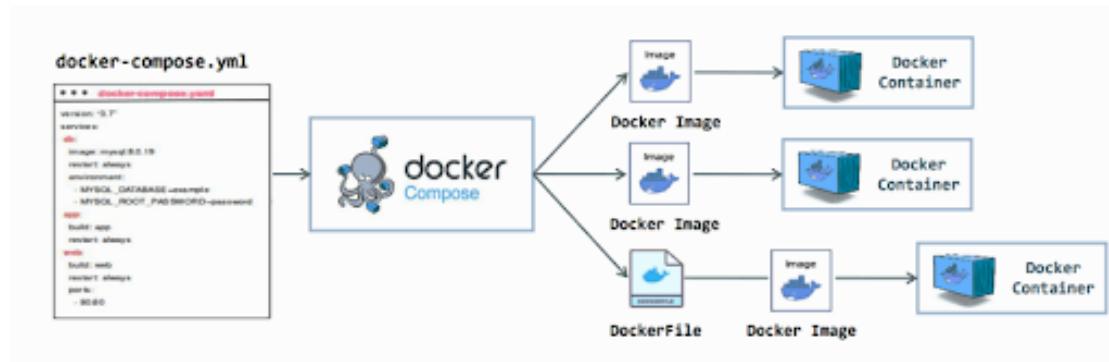
3.2.2 Quy trình hoạt động

Quy trình hoạt động

Quy trình hoạt động của Docker Compose bao gồm các bước sau:

- Tạo tệp docker-compose.yml: Đây là tệp cấu hình định nghĩa các dịch vụ, volume, network và các thiết lập khác cho từng container cần thiết cho ứng dụng.
- Khởi chạy Docker Compose: Sử dụng lệnh docker compose up để khởi động toàn bộ các dịch vụ được định nghĩa trong tệp docker-compose.yml. Docker Compose sẽ:
 - Đọc cấu hình từ tệp docker-compose.yml.
 - Kiểm tra xem các images đã được tải xuống hay chưa. Nếu chưa, nó sẽ tải các images cần thiết từ Docker Hub hoặc build từ Dockerfile chỉ định.
 - Khởi tạo các volume và network nếu chưa có.

- Tạo và khởi động các container: Docker Compose sẽ tạo và khởi động các container dựa trên các dịch vụ được định nghĩa. Nếu có dịch vụ phụ thuộc (depends_on), Docker Compose sẽ đảm bảo thứ tự khởi động phù hợp.
- Kết nối mạng giữa các container: Docker Compose sẽ tự động tạo và cấu hình mạng (network) cho phép các container liên lạc với nhau qua tên dịch vụ (DNS).
- Quản lý logs: Docker Compose tự động thu thập và hiển thị logs của từng container, giúp dễ dàng theo dõi hoạt động của từng dịch vụ trong quá trình vận hành.
- Giám sát và quản lý trạng thái: Trong quá trình chạy, Docker Compose có thể tự động khởi động lại các container nếu có sự cố, tùy thuộc vào thiết lập restart.
- Dừng các container: Sử dụng lệnh docker compose down để dừng và xóa toàn bộ container, network, và volume (nếu được cấu hình) của ứng dụng. Điều này giúp giải phóng tài nguyên hệ thống khi không cần sử dụng ứng dụng.
- Cập nhật ứng dụng (nếu cần): Khi cần cập nhật phiên bản dịch vụ hoặc thay đổi cấu hình, bạn có thể sửa đổi tệp docker-compose.yml, sau đó chạy docker compose up -d để áp dụng các thay đổi mà không cần dừng toàn bộ hệ thống.



Hình 3.2: Luồng hoạt động của Docker Compose

Các câu lệnh thông dụng trong Docker Compose

- `docker compose up`: Khởi động toàn bộ các dịch vụ được định nghĩa trong tệp `docker-compose.yml`.
- `docker compose down`: Dừng và xóa các container, network, và volume.
- `docker compose logs`: Xem logs của tất cả các container.
- `docker compose ps`: Xem trạng thái của các container trong Docker Compose.

3.2.3 Lợi ích của Docker Compose

Tại sao nên dùng Docker Compose?

Quản lý đơn giản: Dễ dàng quản lý các container với cấu hình tập trung. Hiệu quả trong phát triển: Tăng tốc quá trình phát triển bằng cách tổ chức và vận hành nhiều container cùng lúc.

Bảo mật mạng nội bộ giữa các container: Cung cấp một mạng riêng biệt, an toàn giữa các container trong ứng dụng.

Kiểm soát phiên bản: Hỗ trợ theo dõi và duy trì phiên bản cấu hình dịch vụ. Cách ly môi trường host: Đảm bảo rằng môi trường host không bị ảnh hưởng bởi các container.

3.2.4 Ứng dụng thực tế

Phát triển ứng dụng web đa dịch vụ

Ví dụ: Một ứng dụng thương mại điện tử gồm:

- Backend: Chạy bằng Python (Django).
- Frontend: Sử dụng React.
- Cơ sở dữ liệu: PostgreSQL.

Docker Compose file (docker-compose.yml):

```
version: "3.8"
services:
  web:
    build: ./web
    ports:
      - "8000:8000"
    depends_on:
      - db
  frontend:
    build: ./frontend
    ports:
      - "3000:3000"
  db:
    image: postgres
    environment:
      POSTGRES_USER: user
      POSTGRES_PASSWORD: password
      POSTGRES_DB: ecommerce
```

→ Cài đặt nhanh chóng mà không cần cấu hình thủ công từng thành phần.

Quản lý các microservices

Ví dụ: Một hệ thống đặt vé gồm các microservices:

- Service 1: Quản lý người dùng.
- Service 2: Quản lý đặt vé.
- Service 2: Quản lý đặt vé.

Docker Compose file (docker-compose.yml):

```
version: "3.8"
services:
  user-service:
    build: ./user-service
    ports:
      - "5001:5001"
  booking-service:
    build: ./booking-service
    ports:
      - "5002:5002"
  email-service:
    build: ./email-service
    ports:
      - "5003:5003"
```

→ Các dịch vụ hoạt động độc lập nhưng giao tiếp hiệu quả.

Môi trường phát triển đồng nhất

Ví dụ: Một đội phát triển ứng dụng AI sử dụng Docker Compose để đồng bộ môi trường TensorFlow.

- Service 1: Quản lý người dùng.
- Service 2: Quản lý đặt vé.
- Service 2: Quản lý đặt vé.

Docker Compose file (docker-compose.yml):

```
version: "3.8"
services:
  ai-app:
    image: tensorflow/tensorflow:latest
    volumes:
      - ./code:/workspace
    ports:
      - "8888:8888"
    command: ["jupyter", "notebook", "--ip=0.0.0.0", "--allow-root"]
```

→ Dù máy cá nhân khác nhau, môi trường làm việc giống hệt.

Chương 4

DEMO

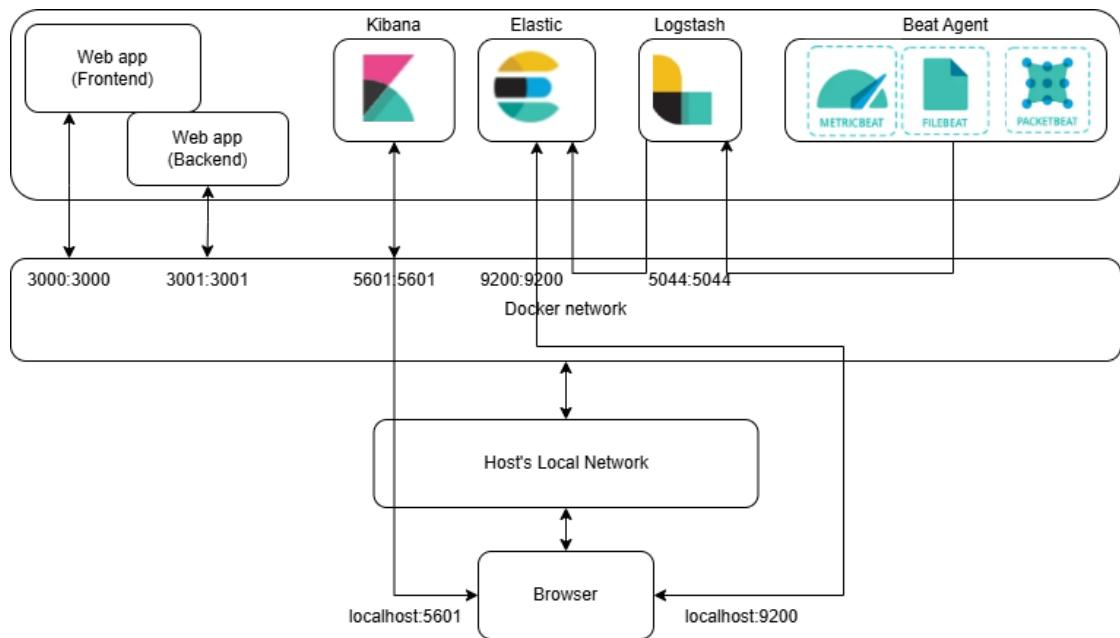
4.1 Tổng quan

Dự án giả (Demo) thiết lập hệ thống giám sát ELK (Elasticsearch, Logstash, Kibana) để thu thập, phân tích, và trực quan hóa log từ một ứng dụng web thương mại điện tử cơ bản, bao gồm các tính năng:

- Đăng nhập/Đăng ký người dùng
- Thêm sản phẩm vào giỏ hàng

Demo bao gồm các dịch vụ sau, được triển khai thành các Docker Container:

- Frontend: Giao diện người dùng, hiển thị trang đăng nhập, đăng ký và thêm sản phẩm vào giỏ hàng.
- Backend: API xử lý logic cho các thao tác như đăng nhập, đăng ký, và quản lý giỏ hàng.
- Elasticsearch: Lưu trữ và xử lý các log thu thập được.
- Logstash: Xử lý và định dạng log, chuyển dữ liệu đã chuẩn bị tới Elasticsearch.
- Kibana: Trình bày giao diện để phân tích và giám sát log trong Elasticsearch.
- Filebeat: Công cụ thu thập log, truyền log từ backend sang Logstash.
- Packetbeat: Công cụ thu thập thông tin liên quan đến mạng từ các thành phần trong hệ thống.
- Metricbeat: Công cụ thu thập các thông số, tình trạng các thành phần trong hệ thống.



Hình 4.1: Topo của dự án Demo

```

.
├── docker-compose.yml
├── frontend/
│   └── (mã nguồn ứng dụng Frontend)
├── express-backend/
│   └── (mã nguồn ứng dụng Backend)
├── elk/
│   ├── filebeat.yml
│   ├── logstash.conf
│   └── metricbeat.yml
├── nginx/
│   └── nginx.conf
└── logs/
    └── (thư mục chứa file log)
        └── nginx/
            └── (log Nginx)

```

Hình 4.2: Kiến trúc Folder của dự án Demo

4.2 Thực nghiệm

4.2.1 Mục tiêu thực nghiệm

- Theo dõi được tình trạng sức khoẻ các dịch vụ qua log
- Lọc được những thông tin cần thiết:
 - Packetbeat: agent.type, destination.ip, source.ip, network.byte, network.transport, network.type
 - MetricBeat: docker-composedepends-on, cpu, memory, health
- Biểu diễn được thông tin thành các biểu đồ
- Tạo các cảnh báo cho hệ thống

4.2.2 Cài đặt và cấu hình các thành phần

Thông qua Docker Compose, các thành phần ELK Stack (Elasticsearch, Kibana, Logstash, Beats modules) được cài đặt và triển khai bằng cách tải các image tương ứng từ Docker Hub còn các thành phần ứng dụng web (frontend, backend) được triển khai từ các image tạo từ Dockerfile tương ứng trên mã nguồn.

Cấu hình các dịch vụ trên Docker Compose

Frontend:

- Tên dịch vụ trên hệ thống: frontend
- Cổng expose: 3000

Backend:

- Tên dịch vụ trên hệ thống: backend
- Cổng expose: 3001

ElasticSearch

- Tên dịch vụ trên hệ thống: elastic
- Image: docker.elastic.co/elasticsearch/elasticsearch:8.12.2
- Chức năng: Lưu trữ và xử lý dữ liệu log.
- Cổng expose: **9200** (HTTP API).

- Giới hạn RAM: 3GB.

Kibana

- Tên dịch vụ trên hệ thống: kibana
- Image: docker.elastic.co/kibana/kibana:8.12.2
- Chức năng: Giao diện hiển thị dữ liệu và giám sát.
- Cổng expose: **5601**.
- Giới hạn RAM: 1GB.

Logstash

- Tên dịch vụ trên hệ thống: logstash
- Image: docker.elastic.co/logstash/logstash:8.12.2
- Chức năng: Xử lý dữ liệu ghi log trước khi đưa vào ElasticSearch.
- Giới hạn RAM: 1GB.

Filebeat

- Tên dịch vụ trên hệ thống: filebeat
- Image: docker.elastic.co/beats/filebeat:8.12.2
- Chức năng: Thu thập và chuyển tiếp log từ backend server.

Packetbeat

- Tên dịch vụ trên hệ thống: packetbeat
- Image: docker.elastic.co/beats/packetbeat:8.12.2
- Chức năng: Thu thập và chuyển tiếp log từ backend server.

Metricbeat

- Tên dịch vụ trên hệ thống: metricbeat
- Image: docker.elastic.co/beats/metricbeat:8.12.2
- Chức năng: Thu thập dữ liệu hệ thống

Dockerfile của dịch vụ frontend

```
FROM node:20.16.0-alpine
WORKDIR /usr/src/app
COPY package*.json .
RUN npm install
COPY . .
RUN npm run build
EXPOSE 3000
CMD ["npm", "start"]
```

Dockerfile của dịch vụ backend

```
FROM node:20.16.0-alpine
WORKDIR /usr/src/app
COPY package*.json .
RUN npm install
RUN npm install -g nodemon
RUN npm install -g ts-node
RUN chmod +x ./node_modules/.bin/ts-node
COPY . .
EXPOSE 3001
CMD ["npm", "start"]
```

Cấu hình Logstash

```
input {
  beats {
    port => 5044
  }
  file {
    path => "/var/log/server/*.log"
    mode => "tail"
  }
}

filter {
  json {
```

```

        source => "message"
        target => "parseJson"
    }
    grok {
        match => {
            "message" => "%{IPV6:ipv6}:%{IPV4:ipv4} -- \[%{DATA:
                parsed_timestamp}\] \\ \"%{WORD:verb} %{URIPATHPARAM:request}
                HTTP/%{NUMBER:httpversion}\\\" \"%{NUMBER:response}\""
        }
    }
    date {
        match => [ "parsed_timestamp" , "dd/MMM/yyyy:HH:mm:ss Z" ]
    }
}

output {
    elasticsearch {
        hosts => "${ELASTIC_HOSTS}"
        user => "${ELASTIC_USER}"
        password => "${ELASTIC_PASSWORD}"
        index => "server-log-%{+YYYY.MM.dd}"
        retry_on_conflict => 3
        cacert=> "certs/ca/ca.crt"
    }
}

```

Cấu hình Filebeat

```

filebeat.inputs:
- type: filestream
  id: my-log-input
  paths:
    - /var/log/server/*.log
  fields:
    service: server

- type: filestream
  id: filebeat-filestream
  paths:
    - ingest_data/*.log

filebeat.autodiscover:
  providers:
    - type: docker
      hints.enabled: true

processors:
- add_docker_metadata: ~

setup.kibana:
  host: ${KIBANA_HOSTS}

```

```

username: ${ELASTIC_USER}
password: ${ELASTIC_PASSWORD}

output.elasticsearch:
  hosts: ${ELASTIC_HOSTS}
  username: ${ELASTIC_USER}
  password: ${ELASTIC_PASSWORD}
  ssl:
    enabled: true
    certificateAuthorities: ${CA_CERT}

Cấu hình Packetbeat

packetbeat.interfaces.device: any

packetbeat.flows:
  timeout: 30s
  period: 10s

packetbeat.protocols:
  - type: http
    ports: [80, 8080, 3000, 3001] # HTTP ports
    fields:
      service_id: nginx

  - type: tls
    ports: [443] # HTTPS port

  - type: dns
    ports: [53] # DNS port
    includeAuthorities: true
    includeAdditionals: true

  - type: ssh
    ports: [22] # SSH port

  - type: ftp
    ports: [21] # FTP port

  - type: dhcipv4
    ports: [67, 68] # DHCP ports

setup.kibana:
  host: "${KIBANA_HOSTS}"
  xpack.enabled: true
  ssl:
    enabled: true
    certificateAuthorities: ${CA_CERT}

output.elasticsearch:
  hosts: ['${ELASTIC_HOSTS}']
  username: "${ELASTIC_USER}"

```

```

password: "${ELASTIC_PASSWORD}"
ssl:
  enabled: true
  certificateAuthorities: ${CA_CERT}

processors:
  - add_host_metadata: ~
  - add_docker_metadata: ~

setup.dashboards.enabled: true

packetbeat.interfaces.device: any

packetbeat.flows:
  timeout: 30s
  period: 10s

packetbeat.protocols:
  - type: http
    ports: [80, 8080, 3000, 3001] # HTTP ports
    fields:
      service_id: nginx

  - type: tls
    ports: [443] # HTTPS port

  - type: dns
    ports: [53] # DNS port
    includeAuthorities: true
    includeAdditionals: true

  - type: ssh
    ports: [22] # SSH port

  - type: ftp
    ports: [21] # FTP port

  - type: dhcipv4
    ports: [67, 68] # DHCP ports

setup.kibana:
  host: "${KIBANA_HOSTS}"
  xpack.enabled: true
  ssl:
    enabled: true
    certificateAuthorities: ${CA_CERT}

output.elasticsearch:
  hosts: ['${ELASTIC_HOSTS}']
  username: "${ELASTIC_USER}"
  password: "${ELASTIC_PASSWORD}"
  ssl:
    enabled: true

```

```

certificateAuthorities: ${CA_CERT}

processors:
- add_host_metadata: ~
- add_docker_metadata: ~

setup.dashboards.enabled: true

```

Cấu hình Metricbeat

```

metricbeat.config.modules:
  path: ${path.config}/modules.d/*.yml
  reload.enabled: false

metricbeat.modules:
  # Elasticsearch module
  - module: elasticsearch
    xpack.enabled: true
    period: 10s
    hosts: "${ELASTIC_HOSTS}"
    username: "${ELASTIC_USER}"
    password: "${ELASTIC_PASSWORD}"
    ssl:
      enabled: true
      certificateAuthorities: ${CA_CERT}

  # Logstash module
  - module: logstash
    xpack.enabled: true
    period: 10s
    hosts: "${LOGSTASH_HOSTS}"

  # Kibana module
  - module: kibana
    metricsets:
      - stats
    period: 10s
    hosts: ${KIBANA_HOSTS}
    username: ${ELASTIC_USER}
    password: ${ELASTIC_PASSWORD}
    xpack.enabled: true
    ssl:
      enabled: true
      certificateAuthorities: ${CA_CERT}

  # Docker module
  - module: docker
    metricsets:
      - "container"
      - "cpu"
      - "diskio"
      - "healthcheck"

```

```
- "info"
- "memory"
- "network"
hosts: ["unix:///var/run/docker.sock"]
period: 10s
enable: true

processors:
- add_host_metadata: ~
- add_docker_metadata: ~

output.elasticsearch:
hosts: ["${ELASTIC_HOSTS}"]
username: "${ELASTIC_USER}"
password: "${ELASTIC_PASSWORD}"
ssl:
enabled: true
certificateAuthorities: ${CA_CERT}
```

4.2.3 Khởi chạy Demo

Bước 1: Đảm bảo Docker và Docker Compose đã được cài đặt.

Kiểm tra xem đã cài đặt Docker chưa: docker version

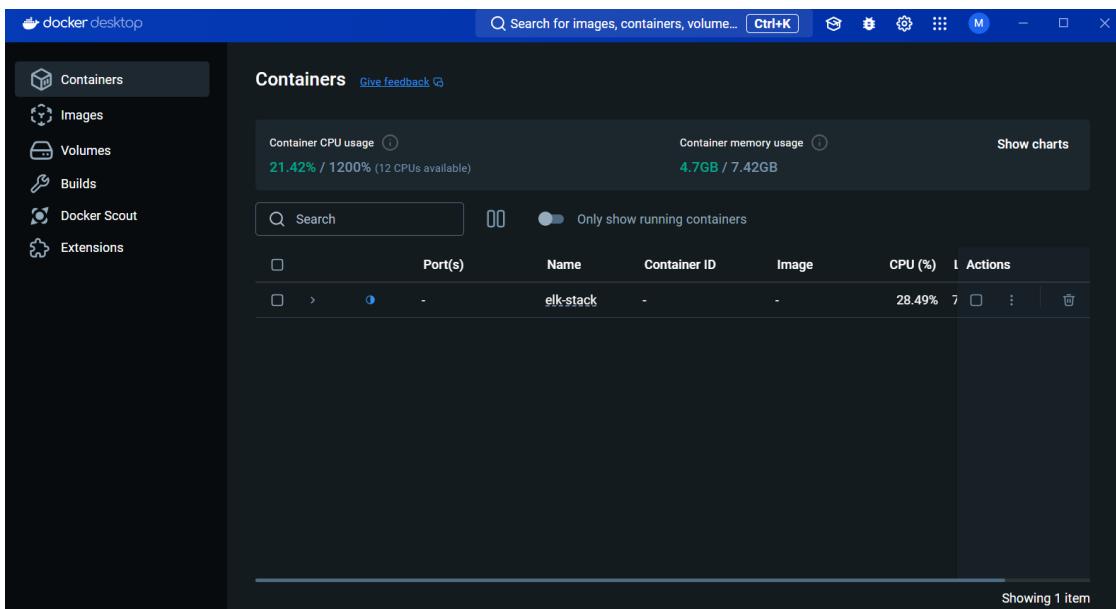
```
PS D:\Works\Projects\sysad-log-monitoring> docker version
Client:
  Version:          27.0.3
  API version:      1.46
  Go version:       go1.21.11
  Git commit:       7d4bcd8
  Built:            Sat Jun 29 00:03:32 2024
  OS/Arch:          windows/amd64
  Context:          desktop-linux

Server: Docker Desktop 4.32.0 (157355)
  Engine:
    Version:          27.0.3
    API version:      1.46 (minimum version 1.24)
    Go version:       go1.21.11
    Git commit:       662f78c
    Built:            Sat Jun 29 00:02:50 2024
    OS/Arch:          linux/amd64
    Experimental:    false
  containerd:
    Version:          1.7.18
    GitCommit:        ae71819c4f5e67bb4d5ae76a6b735f29cc25774e
  runc:
    Version:          1.7.18
```

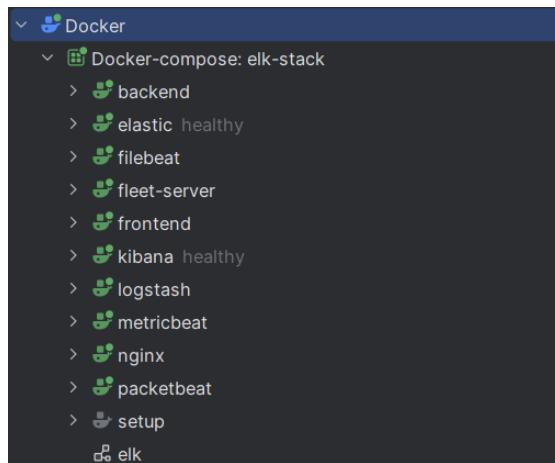
Kiểm tra xem đã cài đặt Docker Compose chưa: docker compose version

```
PS D:\Works\Projects\sysad-log-monitoring> docker compose version
Docker Compose version v2.28.1-desktop.1
```

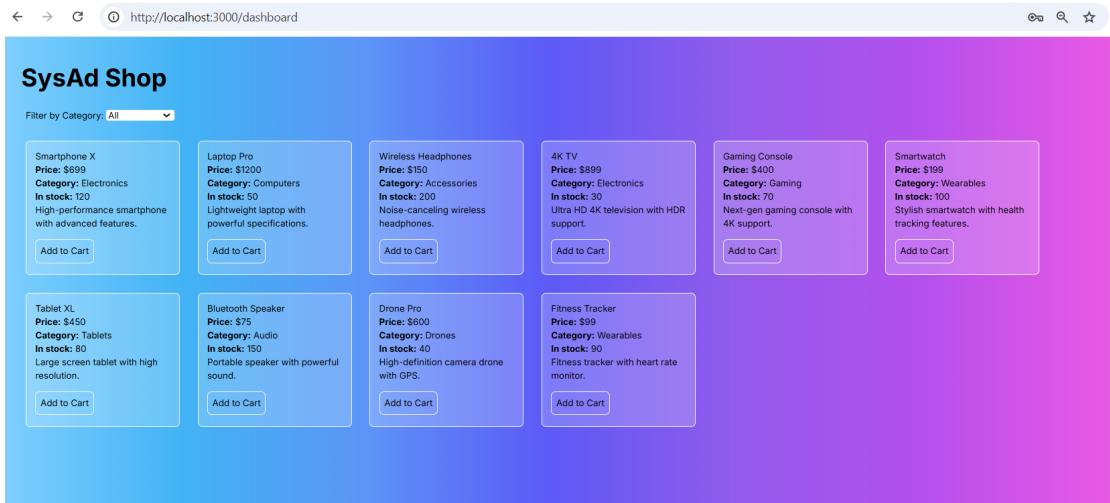
Giao diện Docker Desktop đang hoạt động



Bước 2: Chạy lệnh docker-compose up -d để khởi chạy các container, đợi cho tất cả các dịch vụ chạy thành công.



Bước 3: Truy cập web demo tại <http://localhost:3000>, đăng nhập, và thao tác trên web để tạo log.



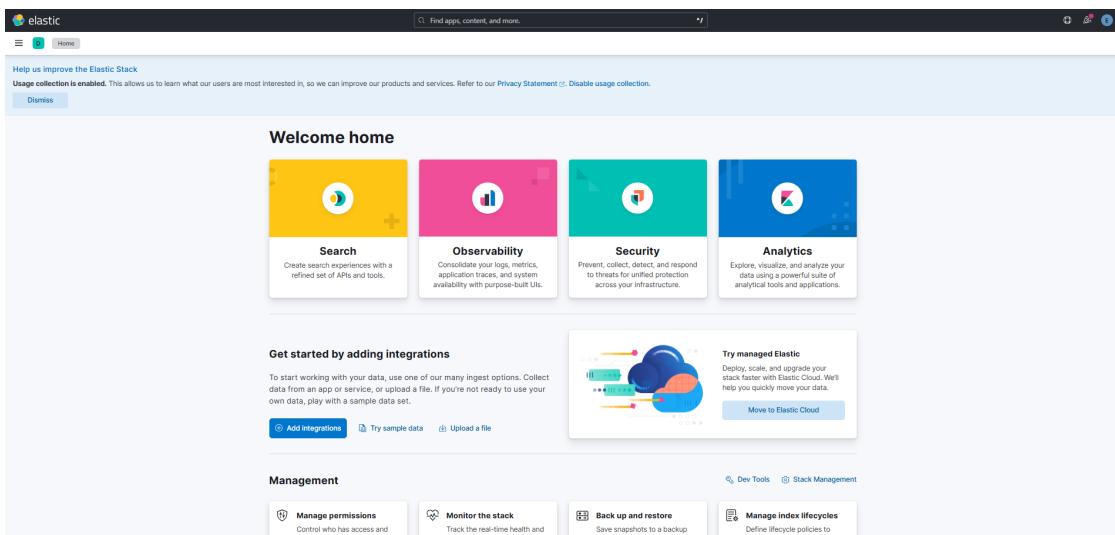
Hình 4.3: Giao diện ứng dụng web

Log được sinh ra

```
{"level":"[u001b[32m[info][u001b[39m","message":"fffff:172.18.0.1 - [15/Dec/2024:11:58:01 +0000] \"POST /api/Login HTTP/1.1\" 200 143 \"http://localhost:3000/\" \u2713 36 ~\n","level":"[u001b[32m[info][u001b[39m","message":"fffff:172.18.0.1 - [15/Dec/2024:11:58:02 +0000] \"GET /api/product/list HTTP/1.1\" 200 2585 \"http://localhost:3000/\" \"Mo\n","level":"[u001b[32m[info][u001b[39m","message":"fffff:172.18.0.1 - [15/Dec/2024:12:00:58 +0000] \"GET /api/product/list HTTP/1.1\" 304 - \"http://localhost:3000/\" \"MoZil\n","level":"[u001b[32m[info][u001b[39m","message":"fffff:172.18.0.1 - [15/Dec/2024:12:01:00 +0000] \"GET /api/product/list HTTP/1.1\" 304 - \"http://localhost:3000/\" \"MoZil\n","level":"[u001b[32m[info][u001b[39m","message":"fffff:172.18.0.1 - [15/Dec/2024:12:01:01 +0000] \"GET /api/cartbyid_id/673242305f52fe432615e9ee HTTP/1.1\" 200 406 \"http://Lo\n","level":"[u001b[32m[info][u001b[39m","message":"fffff:172.18.0.1 - [15/Dec/2024:12:01:05 +0000] \"DELETE /api/cart_delete_id/673242305f52fe432615e9ee HTTP/1.1\" 200 39 \"http://Lo\n","level":"[u001b[31m[error][u001b[39m","message":"fffff:172.18.0.1 - [15/Dec/2024:12:01:06 +0000] \"GET /api/cartbyid_id/673242305f52fe432615e9ee HTTP/1.1\" 404 28 \"http://Lo\n","level":"[u001b[31m[error][u001b[39m","message":"fffff:172.18.0.1 - [15/Dec/2024:12:01:06 +0000] \"GET /api/cartbyid_id/673242305f52fe432615e9ee HTTP/1.1\" 404 28 \"http://Lo\n","level":"[u001b[32m[info][u001b[39m","message":"fffff:172.18.0.1 - [15/Dec/2024:12:01:08 +0000] \"POST /api/cart/add HTTP/1.1\" 200 474 \"http://localhost:3000/\" \"MoZil\n","level":"[u001b[32m[info][u001b[39m","message":"fffff:172.18.0.1 - [15/Dec/2024:12:01:10 +0000] \"POST /api/cart/add HTTP/1.1\" 200 556 \"http://localhost:3000/\" \"MoZil\n","level":"[u001b[32m[info][u001b[39m","message":"fffff:172.18.0.1 - [15/Dec/2024:12:01:12 +0000] \"GET /api/product/list HTTP/1.1\" 304 - \"http://localhost:3000/\" \"MoZil\n"
```

Hình 4.4: Log được tạo sau các thao tác

Bước 4: Truy cập Kibana tại <https://localhost:5601> để xem log và thiết lập giám sát ứng dụng.

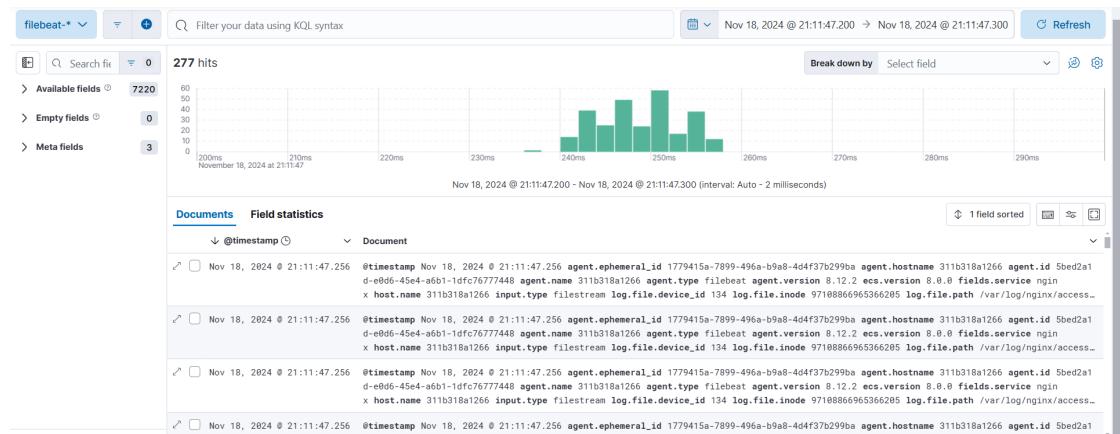


Hình 4.5: Giao diện Kibana

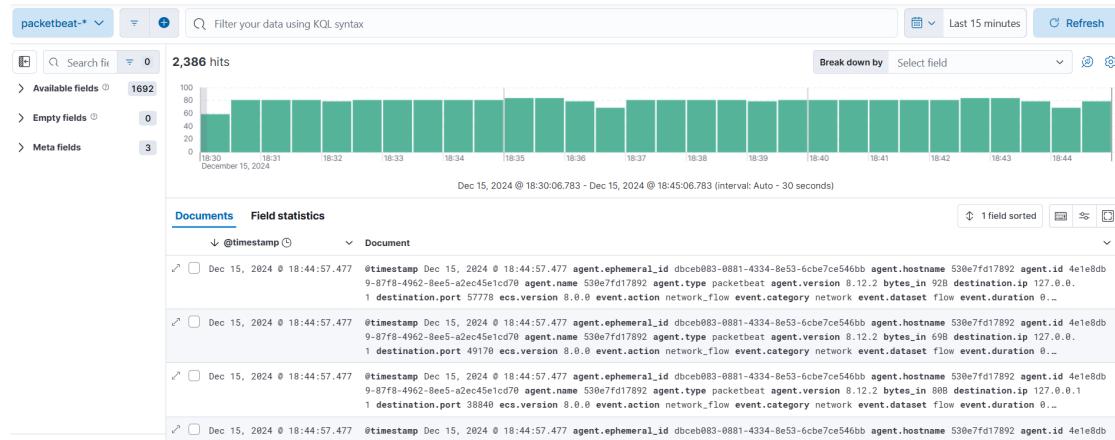
4.2.4 Kết quả thực nghiệm

Giám sát và truy vấn log

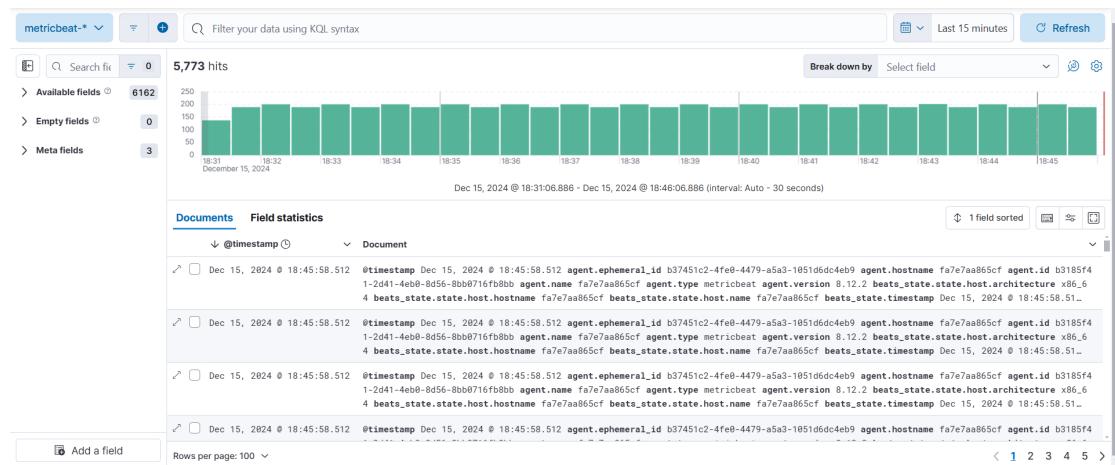
Để quản lý các log thu được, chọn biểu tượng 3 thanh nằm ngang bên trái, lựa chọn Analytics > Discover, khi đó giao diện trực quan log dưới dạng đồ thị được hiện ra, tùy chọn các data view để xem log:



Hình 4.6: Log mà filebeat đã lấy được



Hình 4.7: Log mà packetbeat đã lấy được



Hình 4.8: Log mà metricbeat đã lấy được

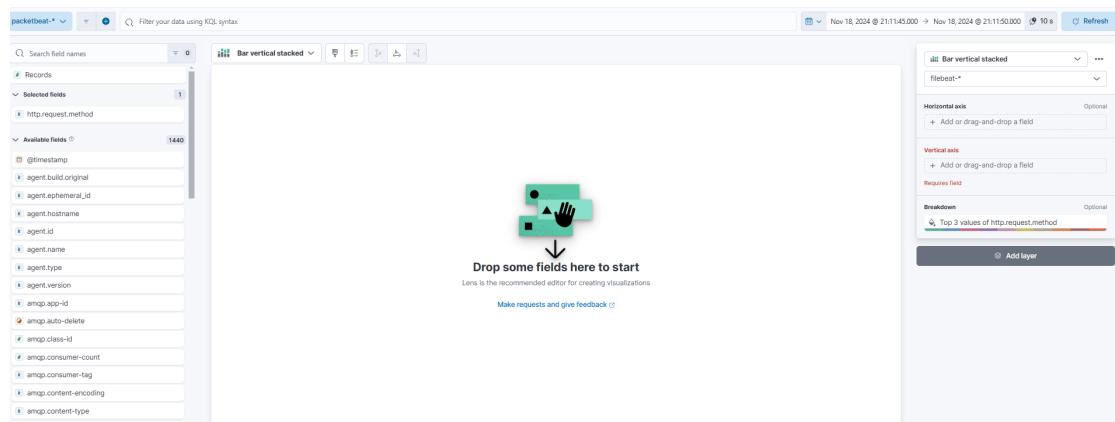
Các thao tác tùy chỉnh:

- Chỉnh sửa khoảng thời gian thu thập log.
- Kết hợp Add filter và thanh Search để lọc ra các log có trường mong muốn.
- Sử dụng phần Available fields để thêm vào giao diện các cột tương ứng với trường dữ liệu đã chọn.

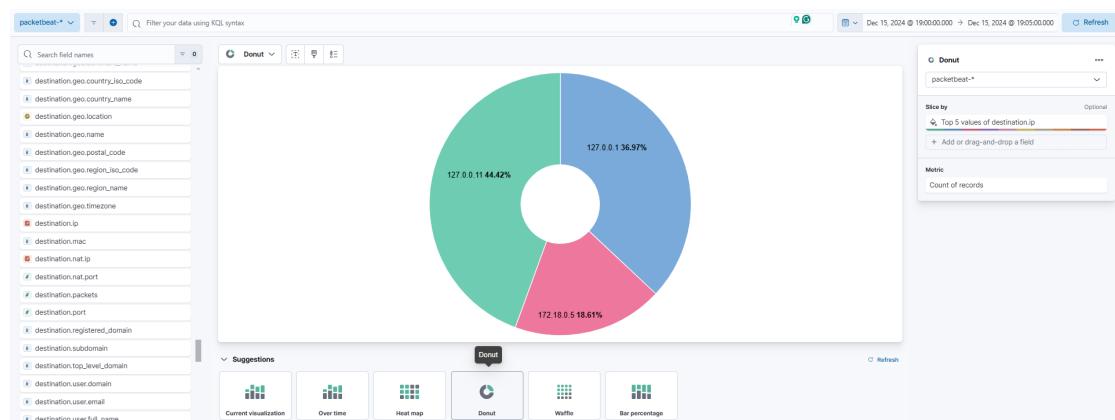
Trực quan hóa log dưới dạng biểu đồ

Để dữ liệu hiển thị được dưới dạng biểu đồ, chọn Analytics > Dashboard > New Dashboard > Create visualization. Ở đây, có thể tìm và chọn một trường dữ liệu, sau đó kéo vào

phần bảng chính giữa. Có thể chỉnh sửa khoảng thời gian lấy log hoặc kết hợp Filter và Search để lọc ra dữ liệu phù hợp.



Hình 4.9: Giao diện tạo biểu đồ mới



Hình 4.10: Biểu đồ biểu diễn các Destination IP

Theo dõi các thông số của các container bằng Metricbeat

Cài đặt Metricbeat trên các container để thu thập dữ liệu giám sát. Cấu hình Metricbeat để gửi dữ liệu tới Elasticsearch.

Để theo dõi các thông số, tình trạng của các container trong Cluster, chọn Management > Stack Monitoring.

The screenshot displays the Stack Monitoring interface for a cluster named 'docker-cluster'. It is organized into three main sections: Elasticsearch, Kibana, and Logstash.

- Elasticsearch:**
 - Overview:** Shows Health (Missing replica shards), Version (8.12.2), Uptime (51 minutes), and License (Basic).
 - Nodes: 1**: Disk Available 91.61% (922.3 GB / 1,006.9 GB), JVM Heap 35.75% (524.9 MB / 1.4 GB).
 - Indices: 45**: Documents 143,077, Disk Usage 174.8 MB, Primary Shards 45, Replica Shards 0.
 - Logs:** A message states "No logs for the selected time. Use the time filter to adjust your timeframe."
- Kibana:**
 - Overview:** Requests 8, Max. Response Time 284 ms, Rule Success Ratio 0.00%, Queued Rules 0.
 - Instances: 1**: Connections 6, Memory Usage 103.72% (543.5 MB / 524.0 MB).
- Logstash:**
 - Overview:** Events Received 0, Events Emitted 0.
 - Nodes: 1**: Uptime 49 minutes, JVM Heap 21.26% (217.7 MB / 1,024.0 MB).
 - Pipelines: 1**: With Memory Queues 1, With Persistent Queues 0.

Hình 4.11: Giao diện Stack Monitoring

Tại đây có thể xem được các thông số từ các thành phần Elasticsearch, Kibana, Logstash trong Cluster như: Tình trạng, số phiên bản, thời gian hoạt động, dung lượng lưu trữ,...

Thiết lập các quy tắc cảnh báo

Để sử dụng chức năng cảnh báo, cần cài đặt xpack security cho ELK Stack
Trong Kibana, vào mục Observability > Alerts.

The screenshot shows the Kibana Alerts interface. At the top, there is a search bar with placeholder text "Search alerts (e.g. kibana.alert.evaluation.threshold > 75)" and a date range selector "Last 15 minutes". Below the search bar are four buttons: "Show all" (highlighted in blue), "Active", "Recovered", and "Untracked".

Below these controls, there is a summary of alert statistics:

Rule count	Disabled	Snoozed	Errors
5	0	0	0

Next to the statistics is a "Manage Rules" button. The main area below the summary is currently empty, indicating no active alerts.

Chọn Manage Rules. Tạo rule mới bằng cách ấn nút Create rules

The screenshot shows the Elastic Observability Rules page. On the left sidebar, under the 'Logs' section, 'Logs' is selected. In the main content area, the 'Rules' tab is active. A summary at the top indicates 5 succeeded, 0 failed, and 0 warning rules. Below this, a table lists five rules:

Name	Last run	Notify	Last response	State
CPU check	Dec 15, 2024 21:57:31pm 29 seconds ago	Succeeded	Enabled	...
Load Average	Dec 15, 2024 21:57:31pm 29 seconds ago	Succeeded	Enabled	...
Memory Usage Check	Dec 15, 2024 21:57:37pm 23 seconds ago	Succeeded	Enabled	...
Synthetics internal TLS rule	Dec 15, 2024 21:57:31pm 29 seconds ago	Succeeded	Enabled	...
Synthetics status internal rule	Dec 15, 2024 21:57:37pm 23 seconds ago	Succeeded	Enabled	...

At the bottom of the page, there are buttons for 'Rows per page' (set to 10) and navigation arrows.

Nhập tên và chọn rule type phù hợp, trong Demo là Metric Threshold (Theo dõi tài nguyên hệ thống như CPU, RAM.)

Create rule

Name
CPU Usage check

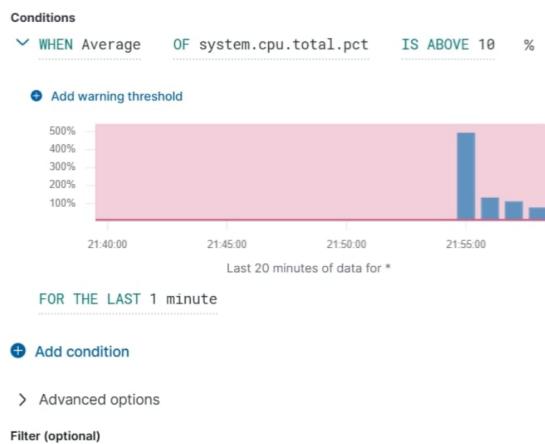
Tags Optional

Select rule type

Metric threshold
Alert when the metrics aggregation exceeds the threshold.

Thiết lập điều kiện để Giám sát mức sử dụng CPU.
Condition: "CPU Usage > 10%" trong 1 phút qua. Sau khi nhập xong thì có bảng điều kiện

Edit rule



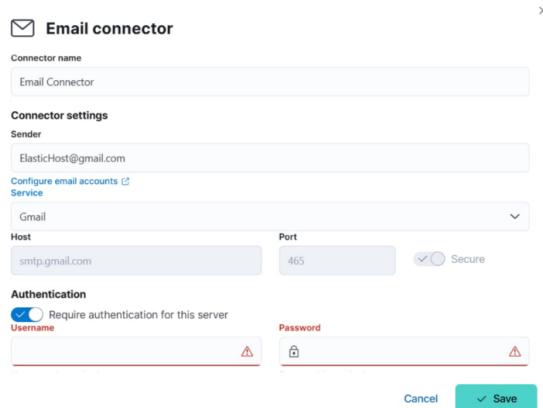
Đến phần action, ở đây có rất nhiều action và ta sẽ sử dụng chức năng gửi email khi điều kiện được trigger. Nhấn nút Email

The screenshot shows the 'Edit rule' configuration page with the 'Actions' section selected. It shows a list of actions: 'Email Connector' (selected) and 'Run when alert'. Below this, there is a 'Select a connector type' section with various connectors listed:

- D3 Security
- Email
- IBM Resilient
- Index
- Jira
- Microsoft Teams
- Opsgenie
- PagerDuty
- Server log
- ServiceNow ITOM

Nếu chưa có connector, ta sẽ tạo 1 email connector.

Sender là địa chỉ mà host tự xung, username là tài khoản gmail mà email connector dùng để gửi, mật khẩu là mật khẩu được tạo ra bởi chức năng "Application Password" - Mật khẩu cho ứng dụng



Set action email có dạng như trên ảnh dưới với thuộc tính message:

```
{{context.reason}}
```

Details:

- CPU Usage: {{state.value}}%
- Threshold: 10%
- Time: {{state.lastTriggeredTime}}

{{rule.name}} is active with the following conditions:

- Affected: {{context.group}}
- Metric: {{context.metric}}
- Observed value: {{context.value}}
- Threshold: {{context.threshold}}

[View alert details]({{context.alertDetailsUrl}})

Edit rule

Email connector

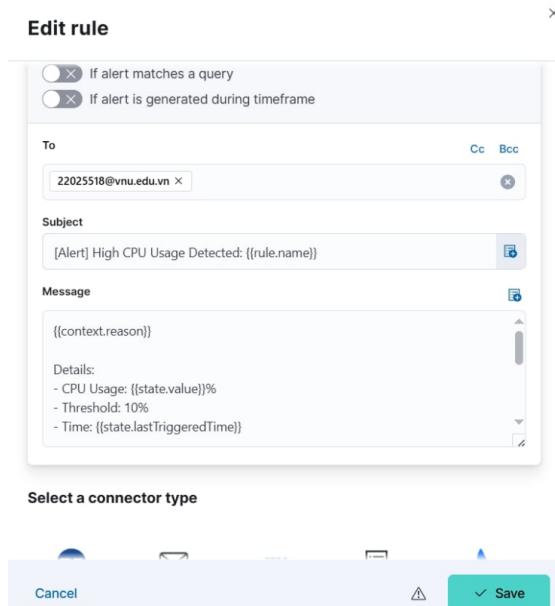
Action frequency: For each alert On check intervals

Run when: Alert

If alert matches a query
If alert is generated during timeframe

To: 22025518@vnu.edu.vn

Subject: [Alert] High CPU Usage Detected: {{rule.name}}



Cài đặt xong cảnh báo thì mỗi 1 phút hệ thống sẽ kiểm tra xem CPU Usage > 10% không, nếu có thì gửi mail thông báo như hình dưới

[Alert] High CPU Usage Detected: CPU check Bên ngoài Hộp thư đến

 phamxuanduong1310@gmail.com
system.cpu.total.pct is 35.9% in the last 1 min. Alert when > 10%. Details: - CPU Usage: % - Threshold: 10% - 1

 phamxuanduong1310@gmail.com đến tôi 17:48 Cf
system.cpu.total.pct is 70.1% in the last 1 min. Alert when > 10%.
Details:

- CPU Usage: %
- Threshold: 10%
- Time:

CPU check is active with the following conditions:

- Affected: *
- Metric: ("condition0" "system.cpu.total.pct")
- Observed value: ("condition0" "70.1%")
- Threshold: ("condition0" "[10%]")

...

Tổng kết

Việc kết hợp ELK Stack với Docker mang lại một hệ sinh thái mạnh mẽ và linh hoạt trong quản trị hệ thống, đặc biệt là đối với các ứng dụng container hóa. ELK Stack giúp đơn giản hóa việc thu thập và quản lý log từ nhiều nguồn khác nhau, tạo ra một hệ thống giám sát tập trung có khả năng phát hiện nhanh chóng các bất thường, tối ưu hóa hiệu suất và cải thiện tính sẵn sàng của dịch vụ. Từng thành phần trong ELK Stack đều đóng vai trò quan trọng: Elasticsearch lưu trữ và tìm kiếm dữ liệu log một cách nhanh chóng; Logstash xử lý và chuẩn hóa dữ liệu; Kibana cung cấp giao diện trực quan để phân tích và theo dõi; và Beats giúp thu thập dữ liệu từ các container Docker cũng như các thành phần khác trong hệ thống.

Ngoài ra, Docker góp phần tối ưu hóa việc triển khai và mở rộng hệ thống, nhờ khả năng cài đặt ứng dụng và tài nguyên, đảm bảo rằng các dịch vụ hoạt động đồng nhất trong mọi môi trường. Việc tích hợp ELK Stack với Docker không chỉ giúp tăng cường khả năng giám sát mà còn đơn giản hóa việc khắc phục sự cố, tối ưu hóa hoạt động hệ thống và đảm bảo an ninh mạng.

Mô hình được trình bày trong báo cáo này chứng minh rằng việc áp dụng ELK Stack kết hợp với Docker không chỉ mang lại hiệu quả vận hành cao mà còn giúp các tổ chức nâng cao khả năng dự đoán và ứng phó với các rủi ro công nghệ. Đây là một giải pháp linh hoạt, phù hợp với các hệ thống công nghệ thông tin hiện đại, đặc biệt trong bối cảnh nhu cầu phân tích và giám sát dữ liệu ngày càng tăng. Với việc triển khai đúng cách, ELK Stack và Docker sẽ trở thành công cụ quan trọng giúp các doanh nghiệp và tổ chức tối ưu hóa nguồn lực, tăng cường bảo mật và duy trì sự ổn định trong vận hành hệ thống.

Tài liệu tham khảo

1. *Deploy ELK Stack với Docker*
<https://viblo.asia/p/deploy-elk-stack-voi-docker-Ny0VG7a5VPA>
2. *Elastic Stack Documentation*
<https://www.elastic.co/docs>
3. *The Complete Guide to the ELK Stack*
<https://logz.io/learn/complete-guide-elk-stack/>
4. *Install Docker Desktop on Windows*
<https://docs.docker.com/desktop/setup/install/windows-install/>
5. *Docker Compose Documentation*
<https://docs.docker.com/compose/>

Bảng đóng góp của các thành viên

Tên thành viên	Mã sinh viên	Công việc thực hiện	Đóng góp
Nguyễn Thị Ngọc Mai	22025510	Trưởng nhóm, viết mã phần Demo, thuyết trình, thiết kế và soạn nội dung báo cáo	16,67%
Đinh Hồng Khanh	22025516	Soạn nội dung thuyết trình, thiết kế slide thuyết trình, soạn nội dung báo cáo	16,67%
Phạm Xuân Dương	22025518	Viết mã phần Demo, thuyết trình, soạn nội dung báo cáo	16,67%
Trần Khánh Duy	22025520	Viết mã phần Demo, soạn nội dung thuyết trình, soạn nội dung báo cáo	16,67%
Trương Quang Minh	22025503	Viết mã phần Demo, thiết kế slide thuyết trình, soạn nội dung báo cáo	16,67%
Hoàng Minh Nghĩa	22028054	Viết mã phần Demo, soạn nội dung báo cáo, soạn nội dung thuyết trình	16,67%