TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI

VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

———————— * ————————


**Bài tập lớn: Hệ trợ giúp quyết định**

# Xây dựng hồ sơ công nghệ phần mềm trung gian Storm, Logstash, Fluentd

*Sinh viên thực hiện:*
**Nguyễn Quang Quý - 20153108**

*Giáo viên hướng dẫn:*
**PGS.TS.**
**Phạm Văn Hải**

**Hà Nội, Ngày 16 tháng 11 năm 2019**

# Mục lục

# Danh sách hình vẽ

# Danh sách bảng

# Danh mục từ viết tắt

| **Từ viết tắt** | **Mô tả** |
|---|---|
| VPN | Virtual Private Network - Mạng riêng ảo |
| OS | Operating System - Hệ điều hành |
| SSH | Secure Shell |
| ELK | Elasticsearch, Logstash, Kibana |
| TCP | Transmission Control Protocol |
| UDP | User Datagram Protocol |
| CPU | Central Processing Unit |
| RAM | Random Access Memory |
| HTTP | HyperText Transfer Protocol |
| JVM | Java Virtual Machine |

# Mở đầu

Một hệ thống thông tin muốn đạt được đến sự ổn định và có thể đáp ứng tốt nhất đối với yêu cầu của người dùng thì đội ngũ phát triển cũng như người điều hành phải có một cách thức để biết được các hoạt động của hệ thống, hành vi cũng như các phản hồi từ người dùng. Những thông tin đó được sử dụng vào nhiều mục đích khác nhau tùy vào đối tượng sử dụng là ai, người quản trị hay người xây dựng, vận hành, phát triển hệ thống, nhưng đều có mục đích chung đó là cải thiện để hệ thống tốt lên. Những thông tin đó có thể được thu thập từ các hoạt động bên trong hệ thống, các hoạt động của người dùng tác động nên hệ thống. Hệ thống có một thành phần hoạt động tách riêng và song song đó là hệ thống Log, có vai trò thu thập, xử lý và lưu trữ các hoạt động của các luồng dữ liệu trong hệ thống, luồng dữ liệu này được tạo ra từ yêu cầu phía người dùng hoặc từ các thành phần của chính hệ thống với nhau hoặc với các hệ thống khác.

Chuyên đề này nhằm giới thiệu về hệ thống Log và một số phần mềm thu thập dữ liệu trong hệ thống Log. Cấu trúc chuyên như sau:

- Chương 1: Tổng quan về quản lý Log
- Chương 2: Giới thiệu công cụ thu thập dữ liệu log: Apache Storm, Logstash, Fluentd

# Chương 1

# Tổng quan về quản lý Log

## 1 Log và phân loại log

Để quan sát và theo dõi hiện trạng của một hệ thống thông tin, người ta sử dụng log. Có thể hiểu đơn giản log là một hoạt động ghi lại các thông tin có ý nghĩa đối với người vận hành trong quá trình hệ thống hoạt động. Các thông tin này được ghi vào trong log file.

Log file là một bản ghi ghi lại các sự kiện xảy ra trong một hệ thống thông tin theo một chu kì được đặt trước. Thông tin trong log file cho ta biết sự kiện nào diễn ra trong hệ thống, thời điểm diễn ra và nguồn gốc tạo ra sự kiện. Ban đầu log file được tạo ra nhằm mục đích xử lý các sự cố trong hệ thống, nhưng về sau log file cung cấp nhiều chức năng hơn. Một số chức năng nổi bật của log đó là:

- **Xử lý sự cố:** Khi một lỗi hoặc sự cố được báo cáo, nơi đầu tiên cần phải xem xét để tìm ra nguyên nhân sự cố đó chính là log
- **Hiểu hành vi của hệ thống hoặc ứng dụng:** Khi một ứng dụng hoặc hệ thống đang chạy, nó sẽ giống như một hộp đen, để có thể xem xét và hiểu được điều gì diễn ra bên trong đó chúng ta phải dựa vào log
- **Phân tích và dự báo:** Với sự tiến bộ về học máy, khai phá dữ liệu và trí tuệ nhân tạo thì những xu hướng những năm gần đây trong phân tích đó là phân tích dự báo. Nó là một nhánh của phân tích nâng cao, được sử dụng để tiên đoán những sự kiện chưa biết có thể xảy ra trong tương lai. Phân tích dự đoán cũng cho phép các tổ chức trở nên chủ động và suy nghĩ tiến bộ, dự đoán kết quả và hành vi dựa trên kết quả thu được và các giả định

Log file được tạo nên từ các thực thể sự kiện. Các thực thể này có thể thuộc các loại log file khác nhau - tương ứng với các sự kiện được sinh ra từ các loại nguồn khác nhau. Tuy nhiên, về cơ bản cấu trúc của một thực thể sẽ bao gồm:

- **Nhãn thời gian(timestamp):** Thời gian xảy ra sự kiện
- **Cấp độ (level):** Mức độ nghiêm trọng của sự kiện. Các hệ thống Log khác nhau sẽ quy định các cấp độ khác nhau nhưng có thể có các cấp độ chung gồm:
  - **DEBUG:** ở cấp độ này, các thông tin debug sẽ được log.

  - **INFO:** ở cấp độ này, các thông tin về luồng làm việc của chương trình sẽ được log
  - **WARN:** cấp độ cho cho phép chúng ta log các thông tin cảnh báo của chương trình
  - **ERROR:** các lỗi khi chạy chương trình sẽ được log nếu chúng ta sử dụng cấp độ này
  - **FATAL:** cấp độ này sẽ log các lỗi nghiêm trọng xảy ra trong chương trình, có thể làm cho chương trình không sử dụng được nữa

- **Thông điệp (message)**: Dữ liệu được đóng gói trong sự kiện

Có nhiều loại thông tin khác nhau được ghi vào trong log file. Ta có thể phân loại như sau:

1. **Thông tin liên quan tới bảo mật:** Hầu hết các hệ thống đều sử dụng các phần mềm bảo mật mạng và máy chủ để xác định các loại hoạt động tấn công, gián điệp nhằm bảo vệ dữ liệu và hệ thống. Do đó, phần mềm bảo mật là nguồn chính tạo ra các log về bảo mật. Những loại phần mềm bảo mật mạng và máy chủ bao gồm:

   - *Antimalware Software:* Thường là các phần mềm diệt vi rút, ghi lại các thực thể malware đã phát hiện, kiểm duyệt hệ thống file và đánh giá mức độ an toàn của hệ thống, ...
   - *Remote Access Software (phần mềm truy cập từ xa):* Hoạt động truy cập từ xa được cấp phép và bảo mật thông qua mạng riêng ảo (VPN). Các log được tạo ra từ VPN thường là các thông tin về hoạt động đăng nhập thành công hay thất bại, ngày giờ và tần suất người sử dụng kết nối và rời đi, lượng dữ liệu gửi và nhận của mỗi người sử dụng trong mỗi phiên hoạt động, ...
   - *Proxies:* là các máy chủ trung gian giữa người dùng và máy chủ phục vụ người dùng đang yêu cầu.
   - *Authentication Servers:* Các thông tin được ghi lại liên quan tới hoạt động xác thực bao gồm: username, trạng thái xác thực (thành công hay thất bại), thời gian xác thực
   - *Routers:* Router có thể được cấu hình để cho chép hoặc từ các loại lưu lượng mạng được xác định theo một chính sách nào đó. Các log mà router ghi lại thường liên quan tới các đặc trưng cơ bản của loại hoạt động tương ứng.
   - *Firewalls:* Cũng giống như router, firewall cho phép hoặc từ chối các hoạt động dựa trên một chính sách nào đó. Tuy nhiên firewall sử dụng nhiều phương pháp tinh vi hơn để giám sát lưu lượng mạng. Firewall có thể theo dõi trạng thái của lưu lượng mạng và giám sát nội dung. Log được tạo ra từ firewall thường chi tiết hơn router.

2. **Thông tin về hoạt động của ứng dụng:** Một số ứng dụng tự tạo log file, trong khi số sử dụng khả năng log của hệ điều hành mà chúng được cài đặt. Các thông tin ứng dụng gồm:

   - *Yêu cầu từ máy khách và phản hồi của máy chủ:* Thông tin này dùng để theo dõi hiệu năng của ứng dụng, hành vi người dùng
   - *Thông tin tài khoản:* Các thông tin về trạng thái xác thực, các thay đổi đối với tài khoản, quyền truy cập của tài khoản, ...
   - *Thông tin về mức độ sử dụng:* Số lượng giao dịch diễn ra theo một chu kì xác định (ví dụ theo phút, giờ, ...) và kích thước của mỗi giao dịch (ví dụ kích thức e-mail, kích thước file được chuyển, ...)

# 2 Khái niệm về quản lý log

Quản lý log [1] được định nghĩa là một cách tiếp cận để có thể giải quyết được các vấn đề liên quan tới việc kích thước log trở lên lớn. Quản lý log bao gồm các hoạt động:

1. **Thu thập dữ liệu**
   Có nhiều chiến lược khác nhau để thực hiện thu thập, chúng thường sẵn có trong các thành phần cơ sở như hệ điều hành, hạ tầng cloud và mạng. Các ứng dụng có thể sử dụng các chiến lược sẵn có hoặc tự xây dựng dịch vụ thu thập. Các tác tử thu thập được cài đặt trên nhiều cấp độ: hệ điều hành (OS), platform và ứng dụng nhằm đáp ứng được các loại thông tin khác nhau. Mức độ chi tiết của log cũng là một yếu tố ảnh hưởng tới việc lựa chọn chiến lược thu thập.

2. **Nhập và tổng hợp dữ liệu**
   Nhập và tổng hợp là quá trình định dạng, nhập dữ liệu từ các nguồn ngoài như ứng dụng, máy chủ, ... Định dạng của log được quy định tùy theo mục đích sử dụng của thông tin cần log, thường bao gồm nhãn thời gian, phân loại thông tin, ví trí phát sinh, sự kiện mạng, ... Dịch vụ nhập cần đáp ứng được các yêu cầu:

   - Các dữ liệu được nhập linh hoạt: từ nhiều nguồn với các định dạng ban đầu khác nhau
   - Sắp xếp và đánh chỉ mục dữ liệu một cách hiệu quả nhằm cho phép tìm kiếm nhanh chóng
   - Dữ liệu luôn sẵn có cho các nhóm khác nhau để thực hiện phân tích và giám sát

3. **Tìm kiếm và phân tích**
   Có nhiều cách để thực hiên tìm kiếm trong một giải pháp quản lý log. Với nền tảng Unix, chúng ta có thể sử dụng các công cụ sẵn có liên quan tới chức năng của hệ thống file như grep, find, ... Bằng cách sử dụng biểu thức chính quy để tìm kiếm nhanh chóng file. Tuy nhiên khi xem xét về khía cạnh dễ sử dụng và trực quan thì các công cụ đó lộ rõ các giới hạn. Các giải pháp quản lý log hiện nay khắc phục các yếu điểm đó bằng cách cho phép tìm kiếm thông qua ngôn ngữ tự nhiên.

   Điểm quan trọng khi xem xét một giải pháp quản lý log là tốt đó là thời gian phản hồi cho truy vấn đối với kích thước của log lớn. Một yếu tố quan trọng khác đó là cung cấp cái nhìn thời gian thực đối với các luồng hoạt động của hệ thống và cho phép nhanh chóng xem xét hoạt động tại bất cứ thời điểm nào đã được ghi lại để tiến hành phân tích tìm ra nguyên nhân chính khi có sự cố.

4. **Trực quan hóa và báo cáo**
   Các dữ liệu trở nên dễ hiệu hơn bằng cách biểu diễn chúng qua các đồ thì và báo cáo. Trước khi xác định mức độ trực quan hóa và báo cáo, cần xác định đối tượng sử dụng báo cáo đó là ai. Người phát triển và người quản lý có thể sẽ có các yêu cầu báo cáo khác nhau so với các nhóm tiếp thị, bán hàng, quyền riêng tư, bảo mật và tuân thủ và có thể yêu cầu các giải pháp khác nhau.

# 3 Thách thức trong quản lý log

Hiện nay, mọi cơ sở hạ tầng truyền thông luôn ở trong tình trạng thay đổi liên tục với các ứng dụng, thiết bị mới. Từ đó mà xuất hiện thêm nhiều định dạng nguồn dữ liệu log khác nhau và tăng về mặt kích thước. Do đó có nhiều thách thức đặt ra đối với một giải pháp quản lý log hiệu quả. Một số thách thức có thể kể đến đó là:

- **Không có định dạng chung:** Mỗi hệ thống tạo log theo định dạng riêng, người quản trị hoặc người dùng cuối cần có chuyên môn để hiểu được nó. Do định dạng là khác nhau cho nên việc tìm kiếm log sẽ trở nên khó khăn. Ví dụ sau minh họa sự khác nhau về định dạng log của SQL server log, Elasticsearch exception/log và NGINX log:


Hình 1.1: Định dạng của log file trong SQL Server, Elasticsearch và NGINX

- **Log thường có tính chất phi tập trung:** Vì log được tạo ra từ nhiều nguồn khác nhau như hệ thống, ứng dụng, thiết bị, ... cho nên nó sẽ xuất hiện tại nhiều máy chủ. Với sự xuất hiện của điện toán đám mây và tính toán phân tán, việc tìm kiếm log sẽ trở nên thách thức hơn khi mà trong trường hợp này các công cụ như SSH va grep không đủ khả năng đáp ứng về vấn đề mở rộng. Do vậy cần phải có một giải pháp quản lý log tập trung nhằm hỗ trợ các nhà quản trị, các nhà phân tích trong việc tìm kiếm thông tin một cách dễ dàng hơn.

- **Không nhất quán về định dạng thời gian:** Khi log được tạo ra với nhãn thời gian, mỗi hệ thống ghi log theo định dạng thời gian của riêng hệ thống đó. Điều này gây khó khăn trong việc xác định chính xác thời gian xảy ra sự kiện (một số định dạng thì tiện lợi cho máy hơn cho con người). Ví dụ một số định dạng thời gian trong log:

```
Nov 14 22:20:10
[10/Oct/2000:13:55:36 -0700]
172720538
053005 05:45:21
1508832211657
```

- **Dữ liệu thu thập là phi cấu trúc:** Điều này gây nên khó khăn trong việc thực hiện phân tích trực tiếp dữ liệu thu được. Do vậy để có thể phân tích được, dữ liệu cần được chuyển đổi về một cấu trúc đã xác định để thực hiện phân tích dễ dàng hơn. Hầu hết công cụ phân tích hiện nay làm việc với dữ liệu có cấu trúc hoặc bán cấu trúc.

# Chương 2

# Giới thiệu công cụ thu thập dữ liệu log: Apache Storm, Logstash và Fluentd

## 1 Apache Storm

### 1.1 Tổng quan


Hình 2.1: Apache Storm

Apache Storm là hệ thống tính toán phân tán mã nguồn mở thời gian thực miễn phí. Nếu như Hadoop xử lý dữ liệu hàng loạt (batch processing) thì Apache Storm thực hiện xử lý dữ liệu luồng (unbounded streams of data) một cách đáng tin cậy [3].

Storm được phát triển bởi Nathan Marz và team BackType, sau này được Twitter mua lại vào năm 2011. Năm 2013, Twitter công khai Storm trên GitHub. Storm đã tham gia vào Apache Software Foundation vào năm 2013. Apache Storm được viết bằng Java và Clojure và đã trở thành chuẩn cho các hệ thống tính toán phân tán thời gian thực [4].

Đặc điểm của Apache Storm:

- Có thể xử lý hơn một triệu tính toán trong một thời gian ngắn trên một node

- Tích hợp với hadoop để khai thác lượng thông tin cao hơn
- Dễ triển khai và có thể tương tác với bất kỳ ngôn ngữ lập trình nào

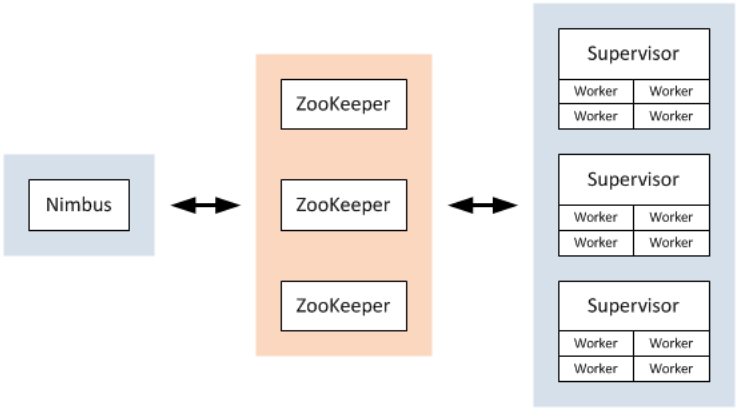
Lợi ích của Apache Storm:

- Storm là mã nguồn mở, mạnh mẽ và thân thiện với người dùng. Nó có thể được sử dụng trong các công ty nhỏ cũng như các tập đoàn lớn.
- Có khả năng chịu lỗi ( fault tolerant), linh hoạt , tin cậy ( reliable) và có khả năng hỗ trợ nhiều ngôn ngữ.
- Cho phép xử lý luồng thời gian thực.
- Storm có thể theo kịp hiệu suất ngay cả khi tăng tải bằng cách thêm tài nguyên tuyến tính. Có khả năng mở rộng cao (scalable)
- Storm thực hiện làm mới dữ liệu và phản hồi phân phối từ đầu đến cuối trong vài giây hoặc vài phút tùy thuộc vào sự cố, có độ trễ thấp (low latency)
- Storm cung cấp đảm bảo xử lý dữ liệu ngay cả khi kết nối trong 1 cluster chết hoặc mất tin.

Apache Storm là một dự án mã nguồn mở được thiết kế theo kiến trúc phân tán trong việc xử lý dữ liệu luồng. Có thể cấu hình Storm để sử dụng với mục đích thu thập dữ liệu, tuy nhiên đây không phải cách làm phổ biến được mọi người khuyên dùng.

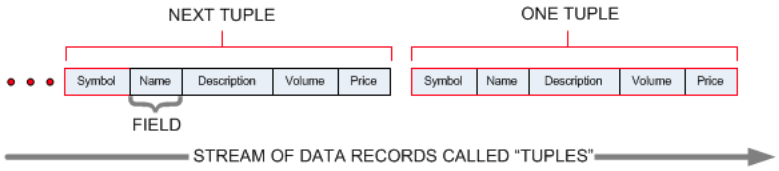
## 1.2 Các thành phần trong Apache Storm

Một hệ thống Storm bao gồm hai thành phần chính là: master node được gọi là Nimbus và worker node được gọi là Supervisor. Nimbus chịu trách nhiệm phân phối các mã dữ liệu trong toàn bộ cluster, phân tác vụ đến các máy chủ và giám sát các tác vụ thất bại. Supervisor thực hiện tiếp nhận các tác vụ mà nó được phân công từ Nimbus; Storm hoạt động dựa trên nền tảng Zookeeper với mục đích quản lý/ phối hợp giữa các node và quản lý trạng thái lưu trữ của dữ liệu.




Hình 2.2: Storm Cluster

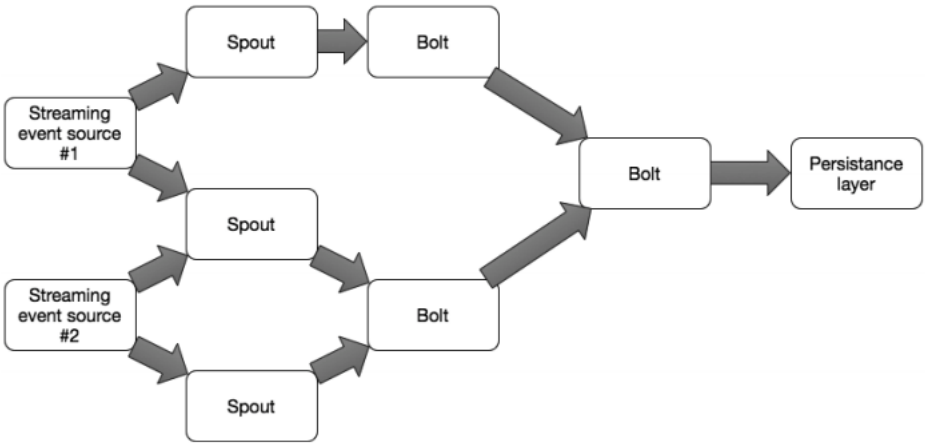
Thành phần cốt lõi của một hệ thống Storm được gọi là Stream. Cấu trúc của Stream bao gồm các tập hợp dữ liệu (tuples of data). Mỗi phần tử trong một tập hợp có thể là bất kỳ loại dữ liệu nào. Kết quả xử lý dữ liệu từ hệ thống Storm có thể xuất ra thành một hoặc nhiều luồng dữ liệu khác nhau; hoặc có thể chuyển đến hệ thống Kafka hay một hệ thống lưu trữ/ cơ sở dữ liệu. Storm cung cấp hai khái niệm cơ bản trong quá trình xử lý dữ liệu được gọi là Bolt và Spout. Bạn có thể triển khai bolt và spout để tạo ra một ứng dụng xử lý luồng dữ liệu theo mong muốn [3].




Hình 2.3: Thành phần Stream trong Apache Storm

Một spout có thể được xem là một nơi tiếp nhận các dữ liệu đầu vào trong mô hình kiến trúc Storm. Nó đóng vai trò thu thập các dữ liệu từ hệ thống Kafka hoặc từ Twitter API hoặc bất kỳ hệ thống nào cung cấp cơ chế xử lý dữ kiện theo luồng. ISpout là giao diện cốt lõi để thực hiện các spouts.

Một bolt có thể tiếp nhận dữ liệu từ một hoặc nhiều nguồn dữ liệu mà spout chuyển đến; nó làm việc hoàn toàn dựa trên kiến trúc mà bạn đã quy hoạch ban đầu. Dữ liệu đầu ra của một bolt có thể phục vụ cho một bolt khác để tiếp tục xử lý. Bolt có thể làm được mọi thứ từ việc chạy các hàm xử lý, thu thập các trường dữ liệu trong một tập hợp, phân tích các luồng dữ liệu, thực hiện xử lý streaming-join, tương tác với cơ sở dữ liệu và nhiều tính năng khác. IBolt là giao diện cốt lõi để triển khai các Bolts.




Hình 2.4: Spout và Bolt trong thành phần Stream

Một hệ thống như vậy sẽ hoạt động cho đến khi bạn kết thúc nó. Mỗi node trong hệ thống bạn có thể thiết lập xử lý song song và Storm sẽ tiến hành phân bổ tài nguyên theo yêu cầu dựa

trên việc tạo thêm các thread xử lý. Khi có một tác vụ thất bại, Storm sẽ tự động khởi không lại chúng.

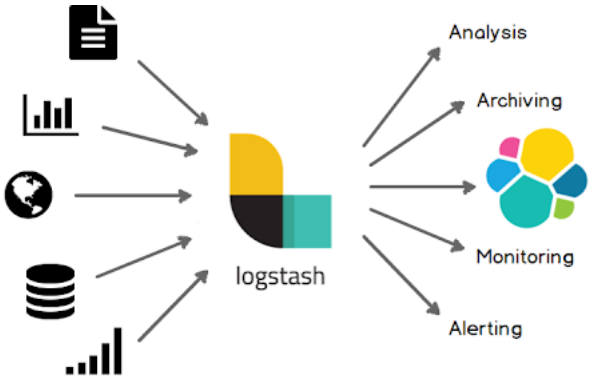
## 1.3 Nguyên lý hoạt động

Một cluster trong Storm thường có một nimbus và một hoặc nhiều supervisor. Một thành phần quan trọng khác là Apache ZooKeeper, sẽ được sử dụng để điều phối giữa nimbus và các supervisor.

1. Ban đầu, nimbus chờ đợi "Storm topology" được gửi đến.
2. Sau đó, nó sẽ xử lý topology và thu thập tất cả nhiệm vụ sẽ thực hiện và thứ tự mà tác vụ được thực hiện.
3. Sau đó, Nimbus sẽ phân phối đều các nhiệm vụ cho các supervisor có sẵn.
4. Tại một khoảng thời gian cụ thể, tất cả supervisor phải gửi heartbeat cho nimbus để thông báo là nó còn hoạt động. Nếu supervisor không hoạt động (không gửi heartbeat cho Nimbus) thì Nimbus sẽ phân công nhiệm vụ cho supervisor khác. Khi hoàn thành công việc, supervisor sẽ đợi nhiệm vụ mới đến.
5. Nếu nimbus dừng hoạt động đột ngột thì các supervisor vẫn làm việc bình thường với các nhiệm vụ đã được giao mà không gặp phải vấn đề gì. Trong khi chờ đợi, Nimbus bị hỏng sẽ được khởi động lại tự động bởi các công cụ giám sát dịch vụ (monitoring tools).
6. Các nimbus khởi động lại sẽ tiếp tục công việc tại thời điểm nó dừng lại. Tương tự, supervisor bị hỏng cũng có thể được khởi động lại tự động. Vì cả nimbus và supervisor có thể được tự động khởi động lại và tiếp tục làm việc như trước nên Storm được đảm bảo xử lý tất cả nhiệm vụ ít nhất một lần.
7. Khi "Storm topology" được xử lý xong, Nimbus sẽ tiếp tục đợi các topology khác đến, các Supervisor tiếp tục đợi công việc được phân công từ Nimbus.

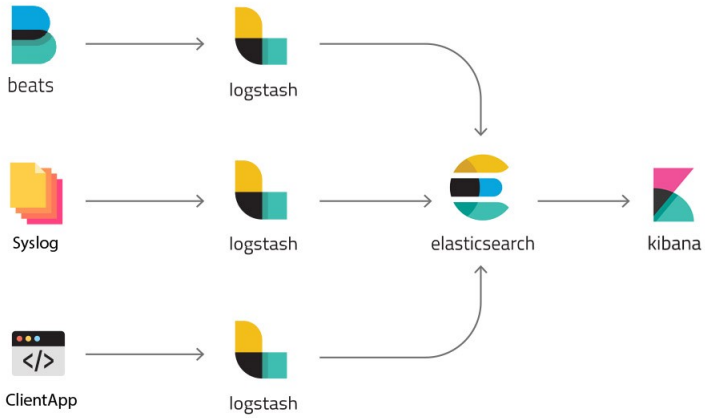
# 2 Logstash

## 2.1 Tổng quan




Hình 2.5: Tổng quan về logstash

Logstash là một công cụ mã nguồn mở thu thập dữ liệu được phát triển bởi Elastic - một công ty chuyên về công cụ tìm kiếm và các giải pháp phân tích dữ liệu. Logstash cho phép chúng ta dễ dàng xây dựng một quá trình pipeline nhằm thu thập dữ liệu từ các đang dạng các nguồn đầu vào và tiến hành phân tách, cải thiện, hợp nhất và lưu trữ nó trên nhiều đích khác nhau [5].




Hình 2.6: ELK Stack

Logstash là một phần trong giải pháp quản lý log có tên ELK Stack, với chức năng tiếp nhận và chuẩn hóa dữ liệu thôi theo yêu cầu. Các thành phần chính trong Logstash bao gồm:

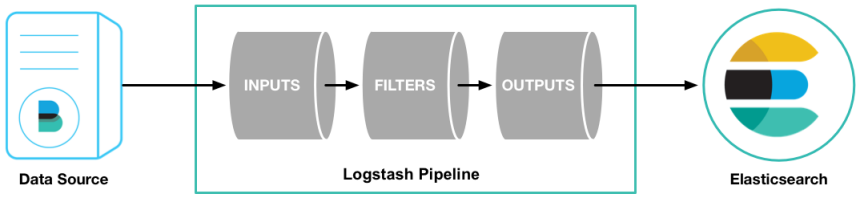
- **INPUT:** Tiếp nhận/thu thập dữ liệu sự kiện log ở dạng thô từ các nguồn khác nhau như file, redis, rabbitmq, beats, syslog, ...
- **FILTER:** Sau khi tiếp nhận dữ liệu sẽ tiến hành thao tác dữ liệu sự kiện log (như thêm, xoá, thay thế,.. nội dung log) theo cấu hình của quản trị viên để xây dựng lại cấu trúc dữ liệu log event theo mong muốn.

- **OUTPUT:** Sau cùng sẽ thực hiện chuyển tiếp dữ liệu sự kiện log về các dịch vụ khác như Elasticsearch tiếp nhận lưu trữ log hoặc hiển thị log, ...

Logstash được viết bằng ngôn ngữ lập trình JRuby - một phiên bản của Ruby chạy trên máy ảo JVM, do vậy mà Logstash có thể chạy trên hầu hết các nền tảng hệ điều hành khác nhau. Các loại dữ liệu mà Logstash thu thập rất đa dạng, ví dụ như: Logs, Packets, Events, Transactions, Timestamp Data, .... và từ hầu kết các nguồn có thể có như mạng xã hội, sàn thương mại điện thử, trò chơi, IOT, . . .

Hiện tại, Logstash đang được sử dụng bởi hơn 765 công ty trên toàn thế giới, với những cái tên nổi bật như Airbnb, Conceptboard, Harvest, ...

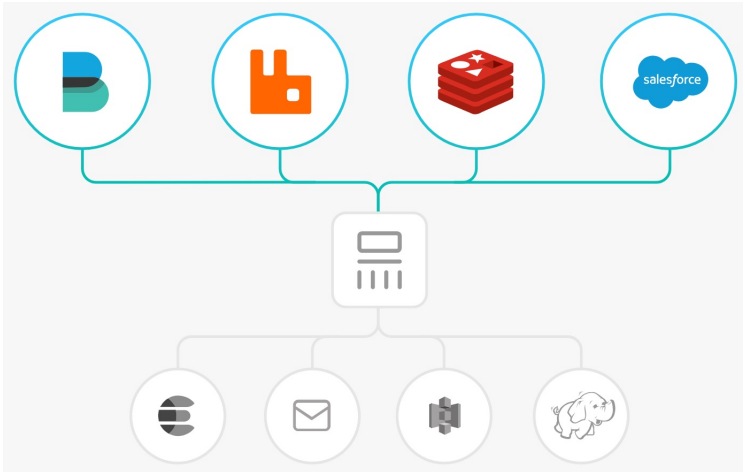
## 2.2 Nguyên lý hoạt động




Hình 2.7: Cơ chế pipeline trong Logstash

Logstash hoạt động theo cơ chế pipeline gồm 3 giai đoạn: Đầu vào (Input), Bộ lọc (Filter), Đầu ra (Output). Đầu vào của quá trình pipeline là dữ liệu thô từ các nguồn khác nhau được đóng gói trong các sự kiện gửi đến pipeline. Đầu ra của quá trình pipeline sẽ là định dạng dữ liệu theo mong muốn của người sử dụng hoặc định dạng phù hợp với hệ thống tiếp nhận sau đó. Đầu vào và đầu ra có hỗ trợ codec cho phép chúng ta mã hóa và giải mã dữ liệu mà không cần thiết phải sử dụng thêm bộ lọc riêng nào khác [5].

1. **Đầu vào (Input)**

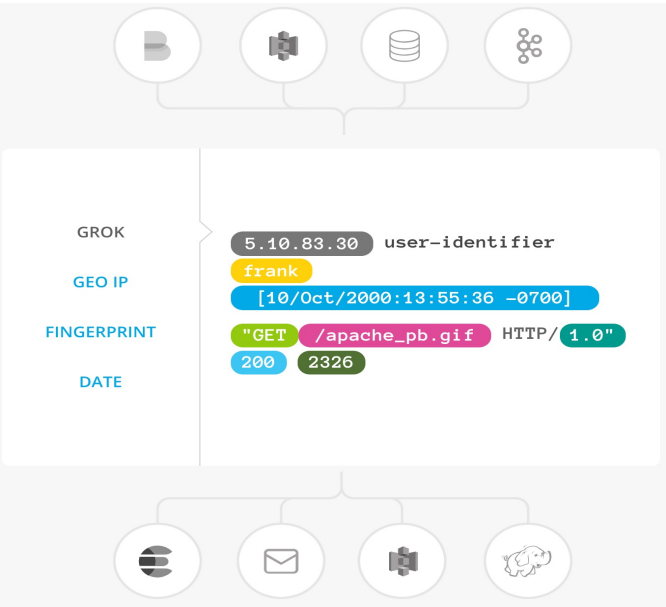



Hình 2.8: Đầu vào của Logstash

Trong giai đoạn này dữ liệu được thu thập từ các nguồn khác nhau để phục vụ cho việc xử lý về sau. Logstash hỗ trợ nhiều plugin cho phép thu thập dữ liệu từ nhiều nền tảng khác nhau. Một số plugin thông dụng đó là:

- *File :* đọc từ một tệp trên hệ thống, giống như lệnh UNIX tail -0F.
- *Syslog :* nghe trên cổng 514 nổi tiếng cho các thông báo nhật ký hệ thống và phân tích cú pháp theo định dạng RFC3164.
- *Redis :* đọc từ máy chủ redis, sử dụng cả kênh redis và danh sách redis. Redis thường được sử dụng như một "broker" trong một mô hình Logstash tập trung, có hàng đợi các sự kiện Logstash từ các "shippers" từ xa.
- *Beats :* xử lý các sự kiện do beats gửi

2. **Bộ lọc (Filter)**



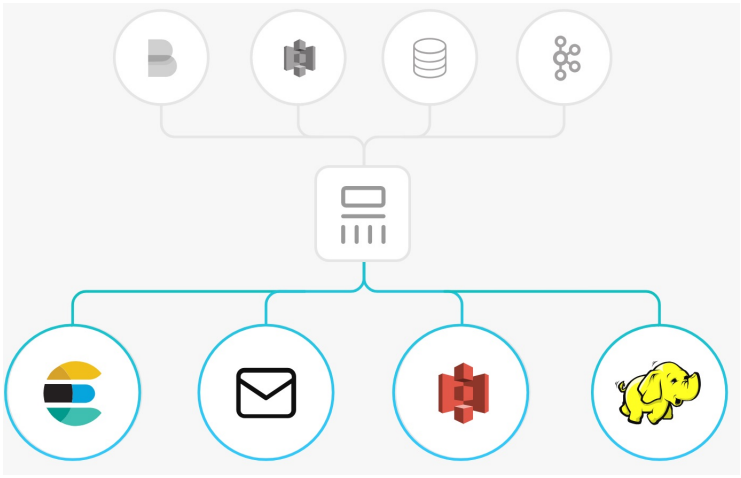

Hình 2.9: Bộ lọc của Logstash

Đây là giai đoạn thực sự diễn ra quá trình xử lý các sự kiện. Bộ lọc có thể được cấu hình để cho phép thực hiện lọc theo các tiêu chí xác định. Một số bộ lọc hữu ích:

- *Grok :* phân tích cú pháp và cấu trúc văn bản tùy ý - chỉnh sửa định dạng log từ client gửi về. Grok hiện là cách tốt nhất trong Logstash để phân tích cú pháp dữ liệu nhật ký không được cấu trúc thành một thứ có cấu trúc và có thể truy vấn được. Với 120 mẫu được tích hợp sẵn trong Logstash, nhiều khả năng chúng ta sẽ tìm thấy một mẫu đáp ứng nhu cầu của mình.
- *Mutate :* thực hiện các phép biến đổi chung trên các trường sự kiện. Bạn có thể đổi tên, xóa, thay thế và sửa đổi các trường trong sự kiện của mình.
- *Drop :* xóa hoàn toàn sự kiện, ví dụ: debug events.
- *Clone :* tạo bản sao của sự kiện, có thể thêm hoặc xóa các trường.

- *Geoip :* thêm thông tin về vị trí địa lý của địa chỉ IP (thường dùng để hiển thị trong Kibana)

3. **Đầu ra(Output)**



Hình 2.10: Đầu ra của Logstash

Một sự kiện có thể đi qua nhiều đầu ra, nhưng một khi tất cả xử lý đầu ra đã hoàn tất, sự kiện đã hoàn tất việc thực thi của nó. Một số đầu ra thường được sử dụng bao gồm :

- *Elasticsearch :* gửi dữ liệu sự kiện tới Elasticsearch. Nếu chúng ta đang có kế hoạch để lưu dữ liệu trong một định dạng hiệu quả, thuận tiện, và dễ dàng truy vấn ... Elasticsearch là con đường để đi.
- *File :* ghi dữ liệu sự kiện vào file trên bộ nhớ.
- *Graphite :* gửi dữ liệu sự kiện tới graphite, một công cụ nguồn mở phổ biến để lưu trữ và vẽ đồ thị số liệu.
- *Statsd :* gửi dữ liệu sự kiện đến statsd.

## 2.3 Ưu và nhược điểm

### Ưu điểm

- Cung cấp biểu thức chính quy cho phép xác định và phân tích các trường của bất kì hầu hết các loại sự kiện đầu vào
- Hỗ trợ đa dạng nguồn dữ liệu
- Cung cấp nhiều plugin cho phép phân tích và chuyển đổi dữ liệu được log thành bất kì định dạng theo mong muốn của người sử dụng
- Hỗ trợ nhiều cơ sở dữ liệu, giao thức mạng
- Logstash được thiết kế theo kiến trúc tập trung, giúp dễ dàng xử lý và thu thập dữ liệu từ nhiều nguồn khác nhau

### Nhược điểm

- Logstash sử dụng giao thức http, ảnh hưởng tiêu cực tới việc xử lý dữ liệu log
- Để sử dụng Logstash, đôi khi cần một sự hiểu biết đủ sâu và đòi hỏi khả năng phân tích dữ liệu đầu vào
- Các plugin trong bộ lọc không giống nhau, do đó cần xác định đúng thứ tự của mẫu để tránh nhầm lẫn khi phân tích

## 2.4 Cấu hình sử dụng cơ bản

Logstash sử dụng một file để cấu hình cho quá trình pipeline. Trong file cấu hình, chúng ta sẽ xác định plugin nào được sử dụng và cài đặt cho từng plugin. Chúng ta có thể tham chiếu tới các trường của sự kiện đầu vào và sử dụng các điều kiện để xử lý các sự kiện khi chúng phù hợp với các tiêu chí xác định nào đó. Để sử dụng file cấu hình, khi chạy Logstash, ta thêm lựa chọn "-f" kèm theo tên file cấu hình mong muốn sử dụng.
Cấu trúc của một file cấu hình gồm 3 phần tương ứng với mỗi giai đoạn của quá trình xử lý pipeline:

```
input {
    . . .
}
filter {
    . . .
}
output {
    . . .
}
```

Mỗi phần sẽ gồm các tùy chọn cho một hoặc nhiều plugin. Nếu chúng ta sử dụng nhiều bộ lọc, thì cần phải chú ý tới thứ tự xuất hiện của plugin tương ứng trong file cấu hình.

Cách thức cấu hình cho một plugin như sau: tên plugin theo sau là một khối các cài đặt cho plugin đó. Ví dụ, cấu hình phần đầu vào với hai file nguồn:

```
input {
  file {
    path => "/var/log/messages"
    type => "syslog"
  }

  file {
    path => "/var/log/apache/access.log"
    type => "apache"
  }
}
```

Trong ví dụ này, có cài đặt được thiết lập cho mỗi file nguồn đó là path và type.

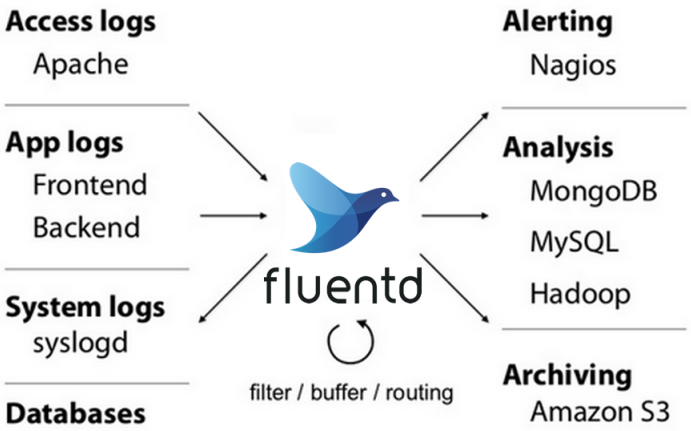
Một plugin có thể yêu cầu giá trị cho một cài đặt thuộc một loại xác đinh như boolean, list hay hash. Các kiểu giá trị được hỗ trợ trong cài đặt plugin được cho trong bảng sau:

| Kiểu dữ liệu | Ví dụ |
|---|---|
| Array | `users => [ {id => 1, name => bob}, {id => 2, name => jane} ]` |
| List | • `path => [ "/var/log/messages", "/var/log/*.log" ]`<br>• `uris => [ "http://elastic.co", "http://example.net" ]` |
| Boolean | `ssl_enable => true` |
| Bytes | • `my_bytes => "1113"  \# 1113 bytes`<br>• `my_bytes => "10MiB"  \# 10485760 bytes`<br>• `my_bytes => "100kib" \# 102400 bytes`<br>• `my_bytes => "180 mb" \# 180000000 bytes` |
| Codec | `codec => "json"` |
| Hash | `match => { "field1" => "value1" "field2" => "value2" }` |
| Number | `port => 33` |
| Password | `my_password => "password"` |
| URI | `my_uri => "http://foo:bar@example.net"` |
| Path | `my_path => "/tmp/logstash"` |

Bảng 2.1: Kiểu dữ liệu trong cấu hình Logstash

# 3 Fluentd

## 3.1 Tổng quan

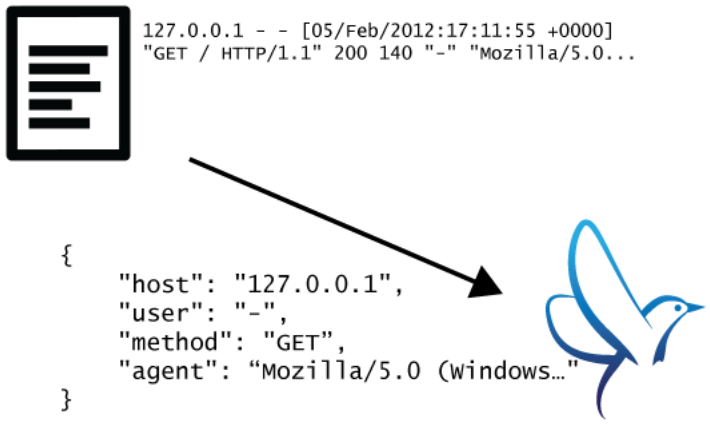



Hình 2.11: Tổng quan về Fluentd

Fluentd là một trình thu thập dữ liệu mã nguồn mở để thống nhất cơ sở lưu trữ log. Nó kết hợp các giải pháp đối với việc tính toán, ứng dụng, và dữ liệu với nhau để việc thu thập và lưu trữ log trở nên đơn giản và dễ mở rộng. Fluentd được phát triển bởi Treasure Data - công ty cung cấp các dịch vụ quản lý dữ liệu [6].

Fluentd có 4 tính năng chính khiến nó có khả năng xây dựng cơ chế pipeline mượt mà và đáng tin cậy dùng cho việc quản lí log:

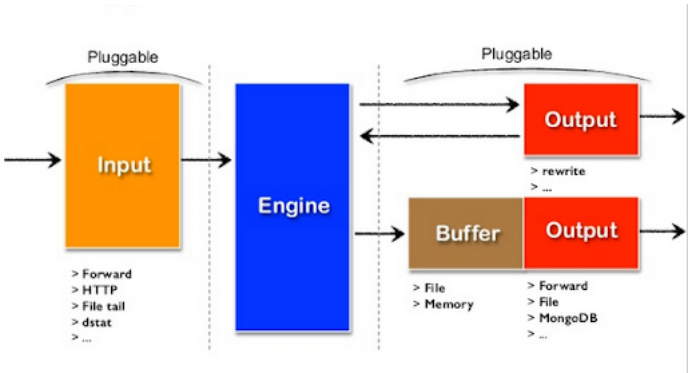
- *Log được kết hợp với JSON*




Hình 2.12: Chuyển đổi log theo cấu trúc JSON

Fluentd cố gắng để cấu trúc dữ liệu như JSON càng nhiều càng tốt. Điều này cho phép Fluentd để thống nhất tất cả các khía cạnh của xử lý log: thu thập, lọc, đệm, và xuất bản ghi log trên nhiều nguồn và điểm đến khác nhau. Việc xử lý dữ liệu downstream là dễ dàng hơn nhiều với JSON, vì nó có đủ cấu trúc để có thể truy cập mà không làm cứng hóa sơ đồ dữ liệu.

- *Hỗ trợ plugin*




Hình 2.13: Hệ thống Plugin trong Fluentd

Fluentd có một hệ thống plugin linh hoạt cho phép cộng đồng để mở rộng chức năng của nó. Hơn 300 plugin cộng đồng đóng góp các plugin kết nối hàng chục nguồn dữ liệu đến hàng chục đầu ra dữ liệu và cho phép thao tác với dữ liệu khi cần thiết. Bằng cách sử dụng plug-in, bạn có thể tận dụng tốt hơn các bản ghi của mình ngay lập tức.

- *Tối thiểu tài nguyên yêu cầu*


Hình 2.14: Ngôn ngữ dùng để xây dựng Fluentd

Một trình thu thập dữ liệu cần có dung lượng nhẹ để nó chạy thoải mái trên mọi máy. Fluentd được viết bằng C và Ruby, nên đòi hỏi tối thiểu tài nguyên của hệ thống. Bản tiêu chuẩn của nó chiếm 30-40MB bộ nhớ và có thể xử lý 18.000 event/giây/lõi.

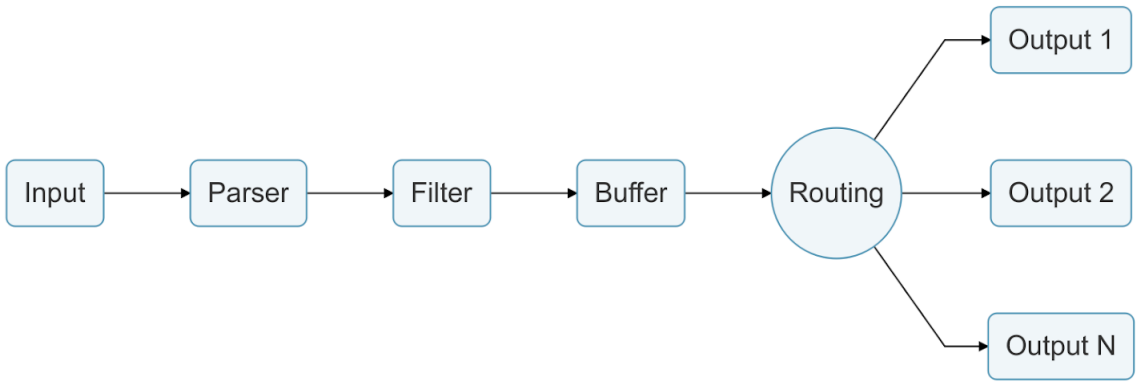
- *Cơ chế đảm bảo dữ liệu có độ tin cậy cao*

Fluentd đảm bảo khả năng xảy ra mất mát dữ liệu ở mức thấp bằng cách hỗ trợ bộ đệm cài đặt trên bộ nhớ trong và bộ nhớ ngoài. Nó cũng hỗ trợ chuyển đổi dự phòng mạnh mẽ và có thể được thiết lập cho tính sẵn sàng cao.

Hiện tại, có nhiều công ty công nghệ lớn đang sử dụng Fluentd như một phần trong stack công nghệ của họ, có thể kể đến như: Microsoft, Amazon, Line, ...

## 3.2 Nguyên lý hoạt động

Cơ chế hoạt động của Fluentd được mô tả như hình:




Hình 2.15: Sơ đồ khối cách thức hoạt động của Fluentd

1. **Input:** Dữ liệu log được gửi tới Input, tại đây nó sẽ được gắn thẻ ứng với nguồn cung cấp dữ liệu. Input hoạt động thông qua các plugin, mỗi nguồn dữ liệu sẽ có thể được hỗ trợ input plugin tương ứng. Một số input plugin thông dụng:
   - *collectd (Collectd):* lắng nghe các gói tin UDP
   - *cpu (CPU Usage):* tính toán lượng CPU được sử dụng của hệ thống
   - *disk (Disk Usage):* tính toán I/O
   - *mem (Memory Usage):* tính toán lượng bộ nhớ được sử dụng của hệ thống
   - *health (Health):* kiểm tra tình trạng của các dịch vụ TCP
   - *proc (Process):* kiểm tra tình trạng của tiến trình
   - *serial (Serial Interface):* đọc dữ liệu từ giao diện ghép nối
   - *stdin (Standard Input):* đọc dữ liệu từ dòng nhập chuẩn
   - *thernal:* tính toán nhiệt độ hệ thống
2. **Parser:** Sau khi dữ liệu được thu thập dưới dạng các sự kiện được gắn tag tại Input, nó sẽ được chuyển đổi về dạng có cấu trúc dựa theo các thẻ tương ứng. Bộ phân tích cú pháp có thể cấu hình được và cho phép xử lý các loại dữ liệu log dựa trên 2 định dạng là JSON và Regex (biểu thức chính quy).

   Ví dụ chuyển từ định dạng log của Apache (HTTP Server)

   ```
   192.168.2.20 -- [28/Jul/2006:10:27:10 -0300]
   "GET /cgi-bin/try/ HTTP/1.0" 200 3395
   ```

   dữ liệu này là một chuỗi đơn thuần, sau khi thực hiện "parser" nó sẽ chuyển thành:

```
{
    "host": "192.168.2.20",
    "user": "-",
    "method": "GET",
    "path": "/cgi-bin/try/",
    "code": "200",
    "size": "3395",
    "referer": "",
    "agent": ""
}
```

3. **Filter:** Cho phép thay đổi dữ liệu trước khi chuyển tới đích. Một số filter plugin được thông dụng:
   - *grep:* chấp nhận hoặc loại bỏ một số bản ghi dựa theo mẫu
   - *kubernestes:* cải thiện log với kurbernestes metadata
   - *lua:* lọc bản ghi sử dụng ngôn ngữ Lua
   - *parser:* phân tích ngữ pháp của bản ghi
   - *record_ modifier:* chỉnh sửa bản ghi
   - *stdout:* hiển thị bản ghi ra đầu ra chuẩn
4. **Buffering:** Khi dữ liệu hoặc log đã sẵn sàng để điều điều hướng tới đích, theo mặc định chúng sẽ được lưu trữ tạm thời tại bộ nhớ hoặc cũng có thể tại hệ thống file để tránh trường hợp mất dữ liệu khi hệ thống gặp sự cố.
5. **Routing:** Điều hướng dữ liệu dựa trên bộ lọc và các thẻ được gắn trên dữ liệu tới một hoặc nhiều đích. Có 2 thành phần quan trọng trong Routing đó là:
   - Tag: được tạo ra sau khi dữ liệu đi qua pha Input
   - Match: dùng để định nghĩa đích đến của dữ liệu
6. **Output:** Cho phép chúng ta định nghĩa đích đến của dữ liệu.Thông thường các đích đến là các dịch vụ điều khiển từ xa, hệ thống file địa phương hoặc các giao diện chuẩn khác. Output được thực hiện thông qua các plugin. Một số output plugin thông dụng:
   - *azure:* nhập bản ghi vào Azure Log Analytics
   - *bigquery:* nhập bản ghi vào Google BigQuery
   - *counter:* đếm số lượng bản ghi
   - *es:* chuyển bản ghi tới Elasticsearch server

## 3.3 Ưu và nhược điểm

## Ưu điểm

- Hỗ trợ đa dạng nguồn và đích: Hầu hết các nền tảng dịch vụ hiện nay như ELK, AWS, ...đều được Fluentd hỗ trợ các plugin tương ứng. Do đó Fluentd có thể theo gần như mọi loại log tính tới thời điểm hiện tại

- Hỗ trợ một số khả năng thao tác luồng bao gồm phân tích nhật ký, chuyển đổi và xử lý dữ liệu.
- Không tốn quá nhiều tài nguyên bộ nhớ: Fluentd yêu cầu chỉ khoảng 40MB RAM. Thậm chí có một phiên bản gọn nhẹ hơn gọi là Fluentd Bit chỉ chiếm khoảng 450KB RAM.

### Nhược điểm

- Vấn đề hiệu năng: Fluentd được viết bằng C, trong khi các plugin được viết bằng Ruby. Điều này tạo nên sự tiện lợi, nhưng sẽ ảnh hưởng tới vấn đề về tốc độ. Đối với phần cứng chuẩn, mỗi thực thể Fluentd chỉ có thể xử lý được khoảng 18000 sự kiện/ giây. Chúng ta có thể tăng tốc độ này lên bằng cách sử dụng nhiều thực thể Fluentd hơn (bằng cách cho phép multi-process worker), nhưng có thể có một số plugin không hỗ trợ tùy chọn này, và do đó có thể gây nên vấn đề.
- Fluentd đóng vai trò như một lớp trung gian giữa nguồn log và đích log. Điều này có thể làm chậm quá trình pipeline khi thực hiện quản lý log khi tốc độ sinh log lớn hơn tốc độ xử lý của Fluentd. Fluentd có hỗ trợ khả năng buffering, điều này có thể gây nên rủi ro đó là khi buffer bị đầy, sự kiện mới có thể sẽ bị bỏ qua.

## 3.4 Cấu hình sử dụng cơ bản

Chúng ta sử dụng Fluentd thông qua việc thay đổi file cấu hình hoạt động của nó. Trong file cấu hình bao gồm các chỉ dẫn để quy định cách thức các sự kiện được xử lý bên trong Fluentd. Các chỉ dẫn trong file cầu hình bao gồm:

- **source:** chỉ dẫn xác định input. Cấu hình input cho Fluentd bằng cách cấu hình các input plugin mong muốn dưới chỉ dẫn <**source**>, sử dụng tham số `@type` để chỉ định input plugin được sử dụng. Các input plugin chuẩn của Fluentd đó là:
  - *http* chuyển Fluentd thành một HTTP endpoint để nhận yêu thông điệp HTTP gửi đến
  - *forward* chuyển Fluentd thành một TCP endpoint để nhận các gói tin TCP

  Ví dụ: cấu hình sử dụng input plugin: http và forward

```
# Receive events from 24224/tcp
# This is used by log forwarding and the fluent-cat command
<source>
    @type forward
    port 24224
</source>
# http://this.host:9880/myapp.access?json={"event":"data"}
<source>
    @type http
    port 9880
</source>
```

- **match:** chỉ dẫn xác định output. Cấu hình input cho Fluentd bằng cách cấu hình các output plugin mong muốn dưới chỉ dẫn <**match**>. Mỗi chỉ dẫn **match** cần kèm theo một mẫu và tham số `@type` để chỉ định output plugin được sử dụng. Các output plugin chuẩn của Fluentd đó là file và forward.

  Ví dụ: Cấu hình cho phép các dữ liệu đầu vào có nhãn là `myapp.access` đi qua được bộ lọc và sau đó lưu trữ vào vị trí `"/var/log/fluentd/access"`

```
 <match myapp.access>
    @type file
    path /var/log/fluent/access
</match>
```

- **filter:** chỉ dẫn xác định cách thức xử lý đường ống sự kiện. Cú pháp của filter tương tự như chỉ dẫn **match** chỉ khác là chúng có thể lồng nhau để thực hiện trong xử lý pipeline. Khi sử dụng filter, luồng di chuyển của các sự kiện sẽ như sau:

```
Input -> filter 1 -> ... -> filter N -> Output
```

  Ví dụ: Sự kiện nhận được mang theo dữ liệu là `{"event":"data"}`, sẽ đi qua bộ lọc `record_tranformer` đầu tiên, tại đây nó được bổ sung thêm trường `host_param` và cuối cùng đi qua bộ lọc với dữ liệu mang theo là `{"event":"data","host_param":"webserver1"}` được ghi vào file với đường dẫn `"/var/log/fluentd/access"`

```
# http://this.host:9880/myapp.access?json={"event":"data"}
<source>
    @type http
    port 9880
</source>
<filter myapp.access>
    @type record_transformer
    <record>
        host_param "#{Socket.gethostname}"
    </record>
</filter>
<match myapp.access>
    @type file
    path /var/log/fluent/access
</match>
```

- **system:** chỉ dẫn thiết lập cấu hình toàn cục trong Fluentd. Các tùy chọn dưới chỉ dẫn này gồm có:
  - log_level

  - suppress_repeated_stacktrace
  - emit_error_log_interval
  - suppress_config_dump
  - without_source
  - process_name (chỉ áp dụng bên trong chỉ dẫn system)

- **label:** chỉ dẫn nhóm output và filter để phục vụ điều hướng nội bộ. Ví dụ:

```
<source>
    @type forward
</source>
<source>
    @type tail
    @label @SYSTEM
</source>
<filter access.**>
    @type record_transformer
    <record>
    # ...
    </record>
</filter>
<match **>
    @type elasticsearch
    # ...
</match>
<label @SYSTEM>
    <filter var.log.middleware.**>
        @type grep
        # ...
    </filter>
    <match **>
        @type s3
        # ...
    </match>
</label>
```

  Trong cấu hình này, các sự kiện forward được điều hướng tới bộ lọc `record_transformer` và đầu ra là elasticsearch còn các sự kiện `in_tail` được điều hướng tới bộ lọc `grep` và đầu ra là `s3` bên trong nhãn `@SYSTEM`.

- **@include:** chỉ dẫn để cho phép phân chia file cấu hình giúp dễ dàng tái sử dụng.

Các kiểu dữ liệu hỗ trợ trong Fluentd:

| **Kiểu dữ liệu** | **Mô tả** |
|---|---|
| String | trường dữ liệu được phân tích như một chuỗi. Mỗi plugin 2 sẽ có những cách xử lý chuỗi riêng |
| Integer | trường dữ liệu được phân tích như một số nguyên |
| Float | trường dữ liệu được phân tích như một số thực |
| Size | trường dữ liệu được phân tích với ý nghĩa số lượng các byte. Cú pháp:<br>• `<INTEGER>k`: kích thước tính theo đơn vị kilobtyes<br>• `<INTEGER>m`: kích thước tính theo đơn vị megabytes<br>• `<INTEGER>g`: kích thước tính theo đơn vị gigabytes<br>• `<INTEGER>t`: kích thước tính theo đơn vị terabytes<br>• `<INTEGER>`: kích thước tính theo đơn vị bytes |
| Time | trường dữ liệu được phân tích với ý nghĩa một khoảng thời gian. Cú pháp:<br>• `<INTEGER>` hoặc `<INTEGER>s`: thời gian tính theo đơn vị giây<br>• `<INTEGER>m`: thời gian tính theo đơn vị phút<br>• `<INTEGER>h`: thời gian tính theo đơn vị giờ<br>• `<INTEGER>d`: thời gian tính theo đơn vị ngày |
| Array | trường dữ liệu được phân tích như một mảng JSON, có hỗ trợ cú pháp ngắn gọn. (ví dụ `a,b` tương đương với `["a", "b"]`) |
| Hash | trường dữ liệu được phân tích như một đối tượng JSON, có hỗ trợ cú pháp ngắn gọn (ví dụ `key1:value1,key2:value2` tương đương với `{"key1":"value1", "key2":"value2"}`) |

Bảng 2.2: Kiểu dữ liệu trong cấu hình Fluentd

# 4 So sánh Logstash và Fluentd

Apache Storm với khả năng xử lý dữ liệu luồng phân tán một cách mạnh mẽ. Tuy nhiên, sử dụng Apache Storm để thực hiện thu thập dữ liệu Log không phải mục đích để tạo ra Apache Storm. Trong khi đó Logstash và Fluentd là hai phần mềm được sử dụng với mục đích chính đó là thu thập dữ liệu Log. Do vậy, trong phần so sánh này sẽ tập trung vào so sánh Logstash và Fluentd để thấy rõ điểm mạnh của từng công cụ.

Fluentd và Logstash đều có hỗ trợ mạnh mẽ khả năng thu thập vào thao tác với nhiều nguồn log khác nhau. Để so sánh hai phần mềm này, ta xem xét qua các khía cạnh sau:

1. **Nền tảng hỗ trợ**

Logstash được phát triển bằng ngôn ngữ JRuby, do đó nó có thể chạy được trên nhiều nền tảng có hỗ trợ JVM. Trong khi đó, ở những phiên bản ban đầu, Fluentd không hỗ trợ nền tảng Window với lý do các thư viện để xây dựng Fluentd chỉ có thể hoạt động trên nền tảng UNIX. Tuy nhiên, những phiên bản gần đây Fluentd đã có thể hoạt động được trên nền tảng Window.

2. **Cách thức điều hướng sự kiện**

Logstash điều hướng dữ liệu vào một dòng đơn nhất và sau đó sử dụng các câu lệnh có dạng `if-then` để gửi chúng tới đúng đích. Ví dụ sau minh họa cho việc gửi các sự kiện báo lỗi trong bản thương mại tới PagerDuty trong Logstash:

```
output {
    if [loglevel] == "ERROR" and [deployment] == "production" {
        pagerduty {
        ...
        }
    }
}
```

Fluentd có một cách tiếp cận khác, sử dụng các thẻ để thực hiện điều hướng. Mỗi sự kiện được tiếp nhận bởi Fluentd sẽ được gắn nhãn để Fluentd có thể biết đích đến của sự kiện đó. Ví dụ sau minh họa cho việc gửi các sự kiện báo lỗi trong bản thương mại tới PagerDuty trong Fluentd:

```
<match production.error>
    type pagerduty
    ...
</match>
```

Ta thấy rằng khi sử dụng Fluentd, các điều hướng phức tạp sẽ được cấu hình một cách rõ ràng hơn so với cách cấu hình sử dụng Logstash.

3. **Hệ sinh thái plugin**

Cả Logstash và Fluentd đều sở hữu hệ thống plugin đa dạng, hỗ trợ nhiều loại đầu vào (file, TCP/UDP), bộ lọc (thay đổi dữ liệu, lọc theo trường), và đầu ra (Elasticsearch, AWS, GCP, Treasure Data, . . . ).

Điểm khác biệt của hai hệ sinh thái đó là cách thức quản lý plugin. Logstash quản lý tất cả plugin trong một kho Github. Fluentd áp dụng một cách tiếp cận phi tập trung hơn - có khoảng hơn 516 plugin của Fluentd, tuy nhiên chỉ có 10 plugin được đặt tại kho plugin chính thức của Fluentd

4. **Cách thức vận chuyển dữ liệu**

Logstash thiếu hàng đợi để duy trì dữ liệu. Hiện tại, Logstash có một hàng đợi trong bộ nhớ trong có thể phục vụ được tối đa 20 sự kiện và một hàng đợi ngoài sử dụng Redis để đảm bảo toàn vẹn dữ liệu khi có sự cố hoặc khởi động lại Logstash.

Trong khi đó, Fluentd được xây dựng với khả năng cấu hình mức cao đối với hàng đợi. Fluentd sử dụng cả bộ nhớ trong và bộ nhớ ngoài để triển khai hàng đợi do đó dữ liệu là đáng tin cậy hơn.

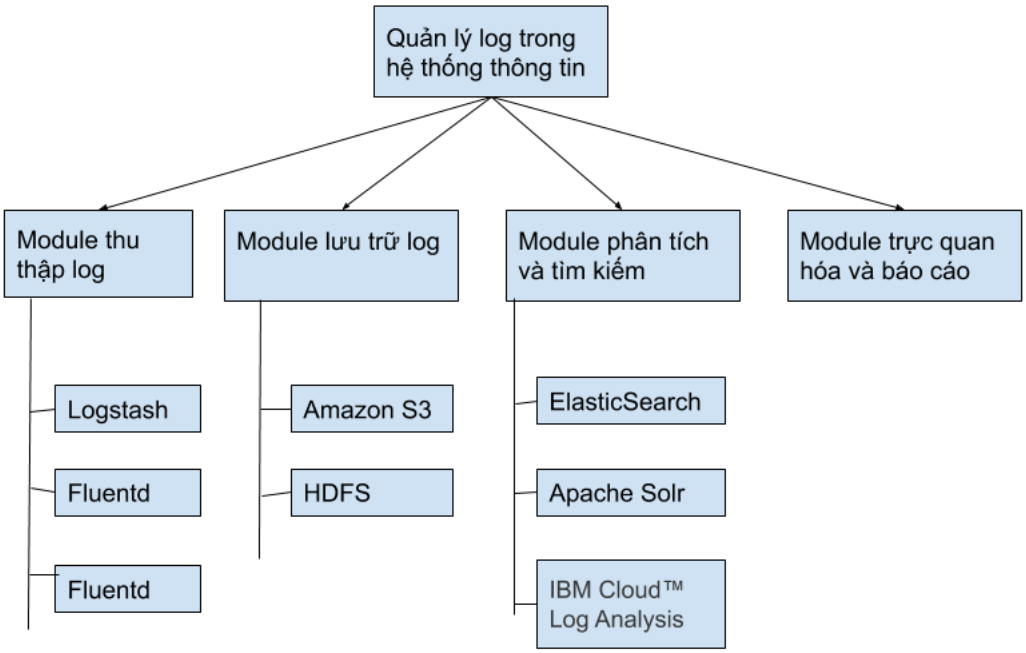
Tóm tắt lại các khía cạnh trên, ta có bảng sau:

| **Tiêu chí** | **Logstash** | **Fluentd** |
|---|---|---|
| Nền tảng hỗ trợ | Đa nền tảng | Đa nền tảng |
| Cách thức điều hướng sự kiện | Sử dụng các câu lệnh theo dạng if-else | Sử dụng thẻ được gắn vào sự kiện |
| Hệ sinh thái plugin | Kho plugin có kiến trúc tập trung | Kho plugin có kiến trúc phi tập trung |
| Cách thức vận chuyển dữ liệu | Cần sử dụng kết hợp với Redis để đảm bảo mức độ tin cậy | Tự đảm bảo độ tin cậy, tuy nhiên phức tạp trong cách cấu hình |

Bảng 2.3: So sánh Logstash và Fluentd [2]

# Chương 3

# Cây công nghệ trong quản lý log




Hình 3.1: Cây công nghệ trong quản lý log của hệ thống thông tin

# Kết luận

Tài liệu đã trình bày được tổng quan về hệ thống Log và các thách thức trong việc quản lý Log. Qua đó cũng giới thiệu một số công cụ hỗ trợ trong việc thu thập và xử lý dữ liệu Log gồm có: Apache Storm, Logstash và Fluentd. Trong đó, Apache Storm có thể được sử dụng với mục đích chủ yếu là xử lý dữ liệu nhờ khả năng tính toán hiệu năng cao của nó, Logstash và Fluentd thì được thiết kế chuyên dụng với mục đích thu thập và hợp nhất dữ liệu log.

Hệ thống Log là một thành phần không thể thiếu trong bất cứ một hệ thống thông tin nào. Với các vai trò cơ bản đã được trình bày trong tài liệu, mức độ quan trọng của việc quản lý Log từ đó mà cũng được nâng cao. Với sự hỗ trợ của các công cụ chuyên dụng, một số thách thức trong quản lý Log đã được giải quyết như hợp nhất định dạng log, quản lý log tập trung, ..., tuy nhiên trong tương lai sẽ có thể có thêm nhiều thách thức mới xuất hiện. Do vậy, việc nghiên cứu về các giải pháp quản lý log vẫn là rất cần thiết trong mọi hệ thống thông tin.

# Tài liệu tham khảo

[1] Log management - https://en.wikipedia.org/wiki/Log-management

[2] Centralized Logging - https://logdna.com/centralized-log-management/

[3] Document Apache Storm (v2.0.0) - http://storm.apache.org/releases/2.0.0/index.html

[4] Hệ thống xử lý dữ liệu luồng - https://techtalk.vn/stream-system.html

[5] Document Logstash (v7.4) - https://www.elastic.co/guide/en/logstash/current/index.html

[6] Document Fluentd (v1.0) - https://docs.fluentd.org/