**GIỚI THIỆT VỀ EFK STACK (ELASTICSEARCH + FLUENTD + KIBANA) VÀ ỨNG DỤNG VÀO MICROSERVICE APPLICATION**

# **1. Giới thiệu về EFK stack**

Khi xem xét các hệ thống nhỏ và đơn lẻ, việc quản lý log cổ điển được diễn ra một cách thủ công: những người quản lý server nói chung và quản lý log nói riêng sẽ thực hiện việc delpoy sản phẩm lên server, khi cần thì mở file log của server ra xem (tail log), hoặc phải tail nhiều log một lúc để theo dõi (log server, log web, log database, …). Việc làm này thỏa mãn được với các ứng dụng nhỏ và ít người dùng, nhưng lại có một số bất cập như không theo dõi lại được lịch sử, tốn thời gian, công sức, khó phát hiện lỗi hơn.

Với những hệ thống lớn, việc quản lý và phân loại log bằng việc xem file log của server để xác định thông tin của log, phân loại log là khá khó khăn. Cần thiết phải có một công cụ quản lý log một cách tốt hơn, sớm phát hiện những lỗi phát sinh của server hoặc kiểm tra các thông tin về log. Hiện nay cũng có khá nhiều công cụ để quản lý log khác nhau. Qua tìm hiểu thì bộ công cụ EFK có nhiều ưu điểm như phần mềm mã nguồn mở hoàn toàn miễn phí, cung cấp dịch vụ quản lý log rất tốt và dễ sử dụng, vì vậy team quyết định áp dụng EFK vào các dự án của công ty.

***EFK stack*** là cụm từ viết tắt của 3 công cụ kết hợp để thực hiện nhiệm vụ quản lý log: ***Elasticsearch*** (E), ***Fluentd*** (F) và ***Kibana*** (K). Cụ thể:

- ***Elasticsearch (E)***: Công cụ lưu trữ, tìm kiếm, phân tích và truy vấn log.

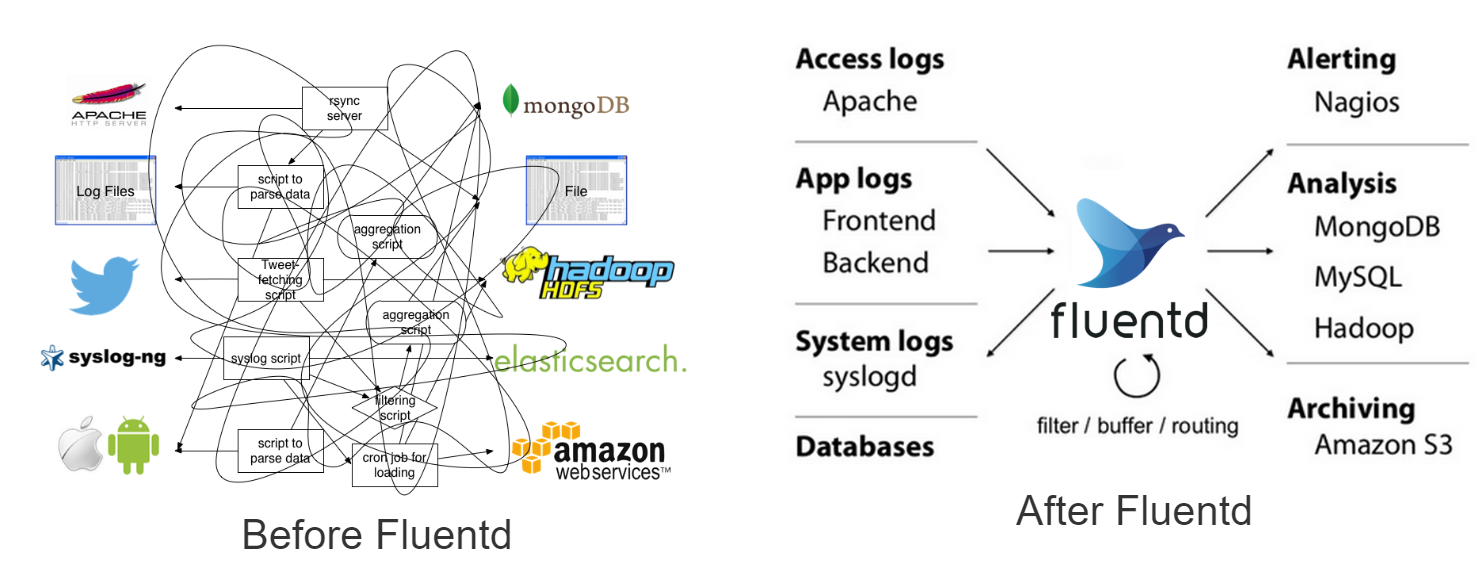
- ***Fluent (F)***: Công cụ “hứng” log (collect log) từ một hoặc nhiều nguồn khác nhau, sau đó có thể forward sang hệ thống khác, hoặc lưu trữ, hoặc ghi ra file.

- ***Kibana (K)***: Đây là công cụ hiển thị log thống kê được từ elasticsearch trên nền web. Thực hiện quản lý và thống kê bằng giao diện trực quan, gần gũi với người sử dụng.



## ***1.1 Fluentd***

Như đã đề cập phía trên, Fluentd là một công cụ thu thập dữ liệu bằng mã nguồn mở, giúp ta có thể đồng nhất việc gom dữ liệu và sử dụng chúng một cách tốt hơn.



Sau khi ta cấu hình cho một hệ thống nào đó (một webservice, database, log file của hệ thống nào đó, …) sẽ đẩy log thu được cho Fluentd, Fluentd sẽ gom log nhận được từ các hệ thống đó đảm bảo thời gian thực. Với mỗi log thu được Fluentd sẽ gắn cho nó một tag, tag này ta có thể cấu hình được. Việc gắn tag này làm cho việc tiền xử lý và phân loại log được rõ ràng và dễ hơn. Sau bước này, sẽ có nhiều lựa chọn tùy thuộc vào mục đích sử dụng, ta có thể ghi log này vào file, hoặc ghi xuống một cơ sở dữ liệu nào đó để theo dõi, hoặc là forward sang một hệ thống khác để thực hiện xử lý tiếp log mà ta thu được, cụ thể trong trường hợp mà ta đang xét đến đó chính là Elasticsearch. Sau khi thu được log, Fluentd sẽ đẩy log đã được gắn tag sang cho Elasticsearch để tiếp tục xử lý.

## ***1.2 Elasticsearch***

Elasticsearch là một công cụ tìm kiếm (search enginee) dựa trên Apache Lucene với trên 15 năm kinh nghiệm. Nó được xây dựng bằng Java để hoạt động như một server cloud theo cơ chế của RESTful. Điều này giúp nó có thể tương tác và sử dụng bởi rất nhiều ngôn ngữ, cũng chính do đó cũng là điểm yếu của nó khi độ bảo mật không cao.



Elasticsearch sở hữu rất nhiều ưu điểm như tìm kiếm nhanh chóng đảm bảo thời gian thực – realtime searching (hoặc gần như thời gian thực – nearly realtime searching) bằng cách đánh index cho dữ liệu; có khả năng phân tích dữ liệu; lưu trữ NoSQL (sử dụng JSON); sắp xếp kết quả truy vấn theo sự liên quan ([relevance](https://www.elastic.co/guide/en/elasticsearch/guide/current/relevance-intro.html)), …

Vì các ưu điểm nêu trên nên Elasticsearch được sử dụng trong xử lý dữ liệu lớn (big data) và được các “ông lớn” trong làng công nghệ sử dụng.



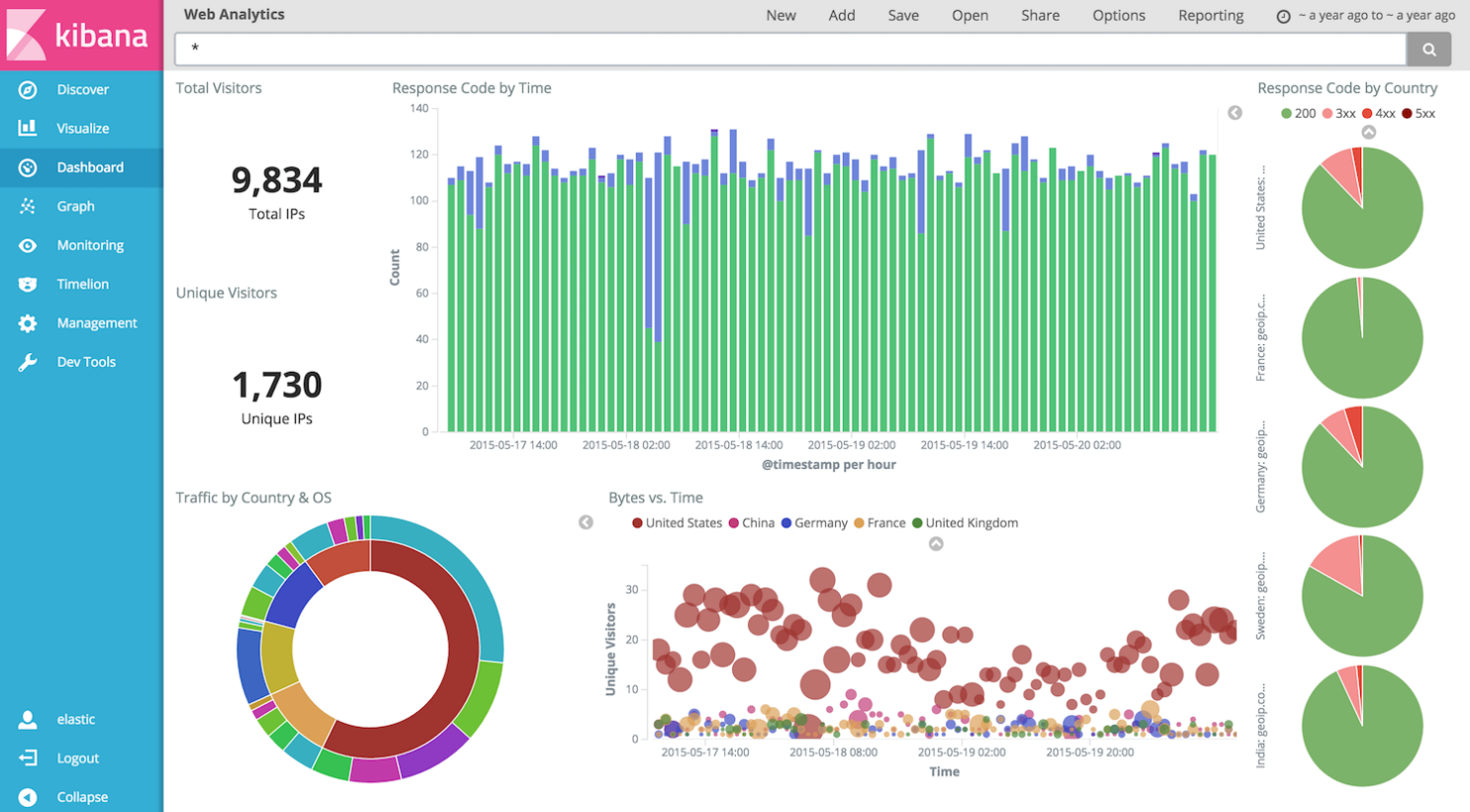
Để giải thích sơ bộ về cơ chế tìm kiếm thời gian thực của Elasticsearch chúng ta cần nói về index. Thông thường khi tìm kiếm một từ nào đó trong một cuốn sách, tương tự với việc tìm kiếm phần tử nào đó trong một bảng, một danh sách, ta thường dùng cách duyệt từ đầu tới cuối, lật từng trang một của cuốn sách và tìm. Nhưng với index thì không thế, việc đánh index cho dữ liệu cũng giống như việc một cuốn sách có phụ lục (index table) lưu các từ xuất hiện trong cuốn sách và trang tương ứng mà nó xuất hiện. Vì vậy nên khi muốn tìm một từ nào đó trong một cuốn sách, ta có thể đi tới phần phụ lục, tìm từ ta muốn và nhảy tới trang được ghi ở phụ lục ta có thể thấy từ đó, tránh phải đọc toàn bộ cuốn sách.

Ngoài cơ chế lưu dữ liệu NoSQL và sử dụng index là tiêu biểu của Elasticsearch còn có rất nhiều thuật toán và cách thức hoạt động phức tạp khác, trong phạm vi seminar sẽ không đề cập, bạn đọc có nhu cầu tìm hiểu xin vui lòng google thêm hoặc xem ở link trong phần danh mục tham khảo các mục [[4]](#_3._Reference), từ mục [[7]](#_3._Reference) tới mục [[13]](#_3._Reference).

## ***1.3 Kibana***

Là một nền tảng phân tích và trực quan hóa dữ liệu được thiết kế và hoạt động chặt chẽ với Elasticsearch, Kibana cung cấp các tính năng mạnh mẽ và dễ sử dụng như các loại biểu đồ thống kê (biểu đồ đường, biểu đồ quạt, biểu đồ miền, …. ) và hỗ trợ không gian địa lý thích hợp.

Trong phạm vi seminar, Kibana được sử dụng để thống kê các log sau khi phân tích từ Elasticsearch.



## ***1.4 Lợi ích của EFK***

- Quản lý log tập trung, đọc log từ nhiều nguồn.

- Dễ tích hợp với các hệ thống ngoài: Dù server có là ngnix hay apache, hay cơ sở dữ liệu là MySQL hay Oracle, … thì ta đều có thể tích hợp được một cách dễ dàng.

- Miễn phí: Bộ công cụ này đều là open-source và free.

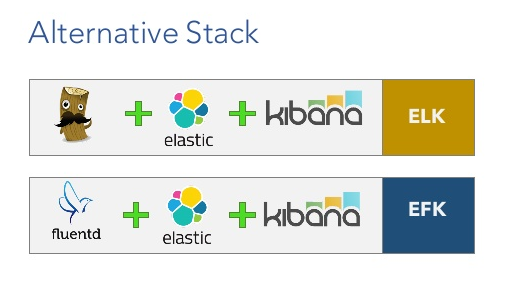
- Khả năng mở rộng và phát triển tốt: Fluentd và Elasticsearch chạy trên nhiều node nên hệ thống EFK cực kì dễ scale. Khi có thêm service, thêm người dùng, muốn log nhiều hơn, bạn chỉ việc thêm node cho Fluentd và Elasticsearch là xong.

- Có bộ lọc và công cụ tìm kiếm mạnh mẽ: Elasticsearch cho phép lưu trữ thông tin kiểu NoSQL, hỗ trợ luôn Full-Text Search nên việc query rất dễ dàng và mạnh mẽ.

- Cộng đồng hỗ trợ lớn, nhiều tutorial.

## ***1.5 Tại sao lại sử dụng EFK chứ không phải là ELK***

Trên thế giới hiện tại có rất nhiều logging stack/logging system đang được sử dụng, trong đó phổ biến hơn cả đó chính là EFK (elasticsearch + fluentd + kibana) và ELK (elasticsearch + logstash + kibana). Nhưng tại sao lại chọn EFK thay vì ELK?



Thực tế Logstash đã có và phát triển từ rất lâu nên đã phần nào chiếm được lòng tin của các DevOps khi triển khai. Ngược lại, Fluentd hãy còn non trẻ và dần dần đi gây dựng vị thế cho mình. Trong các dự án thực tế team từng biết (xin phép không đề cập tên tại đây), đội DevOps đã chọn ELK để phát triển như một logging stack đầu tiên. Nhưng vấn đề phát sinh ra từ hiệu năng của Logstash khi áp dụng cho một hệ thống lớn. Khi ấy, Logstash thể hiện một chút yếu kém về hiệu năng trong khi đó Fluentd lại ghi điểm với các xử lý có hiệu suất cao hơn người anh em đi trước Logstash của mình.

Ngoài ra khi tìm kiếm trên google với từ khóa *ELK vs EFK* hoặc *Comparison between Fluentd and Logstash* sẽ ra kết quả trực quan hơn:

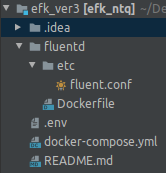
* <https://logz.io/blog/fluentd-logstash/>
* <https://www.reddit.com/r/devops/comments/9quyzo/from_elk_to_efk_why/>
* …

# **2. Ứng dụng EFK vào microservice application**

## ***2.1 Cài đặt EFK***

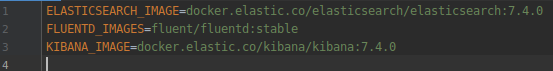
Trong mục này, ta sẽ đi vào phần cài đặt EFK sử dụng docker.

Cấu trúc thư mục của project:



Trước hết ta cần có file ***docker-compose.yml*** để cấu hình các images cần thiết. Các images này bao gồm: fluentd, elasticsearch và kibana. Link git của project config: *<insert link here>*

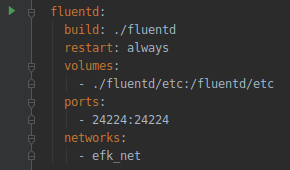
Trước tiên, các biến môi trường được định nghĩa ở file ***.env***



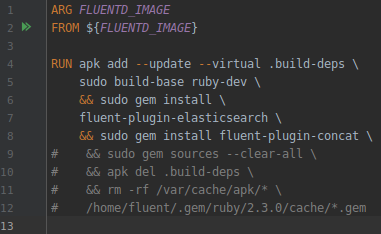
Với ES và Kibana có 2 loại: ***oos*** và ***non-oos***. Điểm khác biệt giữa 2 loại này đó chính là giấy phép, loại ***oos*** là loại giấy phép mã nguồn mở (*open source license*) còn ***non-oos*** là loại giấy phép cơ bản (*basic license*). Ở đây team khuyến nghị sử dụng ***non-oos*** vì loại này ta có thể custom được x-pack security, có nhiều danh mục menu Kibana hơn khi sử dụng [[14]](#_3._Reference).

Danh mục phiên bản của ES và Kibana được liệt kê tại link <https://www.docker.elastic.co/>

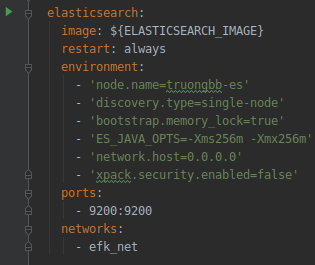
Ta đi vào cấu hình cho Fluent trong file ***docker-compose.yml***



Fluentd sẽ được expose tại cổng 24224 với images được build chứ không phải lấy image mặc định nữa, với volume cấu hình được định nghĩa ở thư mục ***fluentd/etc*** của project (sẽ đề cập chi tiết phần sau). Fluentd sử dụng trong project này cần một số plugin để cấu hình thực hiện filter, match với log thu được, mà các plugin này lại không được cài sẵn ngay trong Fluentd image, vậy nên ta phải build thành một image với các plugin được cài theo mục đích sử dụng. Dockerfile được định nghĩa để build Fluentd, nó được để trong thư mục ./fluentd



Cấu hình cho Elasticsearch:



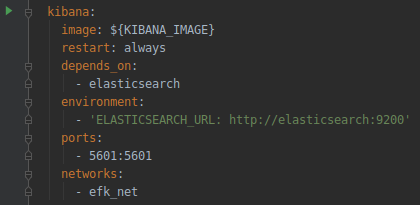
Elasticsearch được expose tại cổng 9200 với images được định nghĩa ở file ***.env*** bên trên. Với các cấu hình môi trường như node.name, discovery.type, … Những cấu hình này có thể được viết ở phần enviroment hoặc là chúng ta có thể tách thành file elasticsearch.yml riêng như config ES trong thư mục /etc/elasticsearch/elasticsearch.yml khi ta cài đặt bằng lệnh linux. Ví dụ như:

[1] <https://github.com/elastic/elasticsearch/blob/master/distribution/src/config/elasticsearch.yml>

[2] <https://gist.github.com/zsprackett/8546403>

Điểm chú ý ở đây có mục xpack.security.enabled, mục này quy định plugin xpack cho elasticsearch, nhưng hiện tại team chưa thực hiện cài đặt được xpack nên tạm thời disable nó đi. Chi tiết cài đặt ở mục [14].

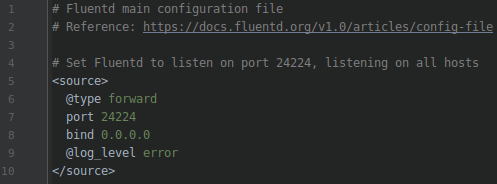
Cuối cùng là config cho Kibana:



Được public lên cổng 5601, Kibana sẽ phụ thuộc vào elasticsearch (ở mục depends\_on và enviroment).

Cuối cùng là file cấu hình fluentd – fluentd.conf, file này sẽ định nghĩa ra các cấu hình cho log, tag, …

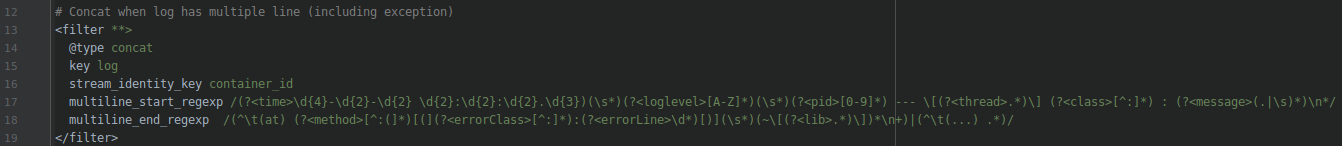
Phần đầu file sẽ là cấu hình chung



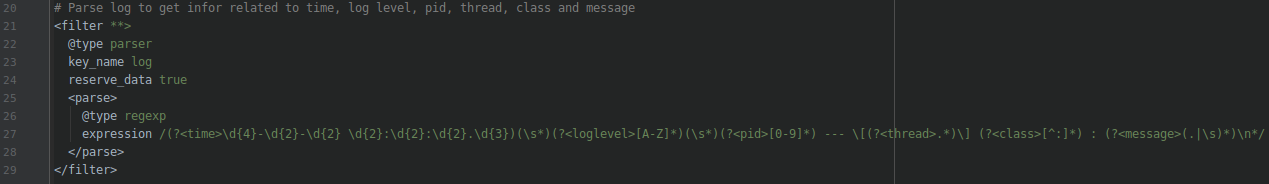
Kiểu của fluentd ở đây là forward, do nó sẽ chuyển log cho elasticsearch để phân tích. Nó lắng nghe tại cổng 24224.

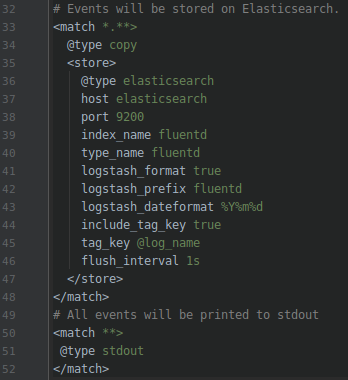
Phần tiếp theo đó chính là 2 mục filter:

- Mục filter đầu tiên sẽ sử dụng plugin concat nhằm mục đích nối các dòng log khi gặp log nhiều dòng, ví dụ như exception trong java chẳng hạn.

Log nhiều dòng mà team muốn bắt chủ yếu là exception, vì vậy nên nó có kiểu nhận dạng ở dòng đầu và dòng cuối được quy định ở multiline\_start\_regexp và multiline\_end\_regexp. Ngoài ra, nếu muốn cấu hình thêm cho filter này để lọc log một cách chi tiết hơn, xin vui lòng xem tại [15].

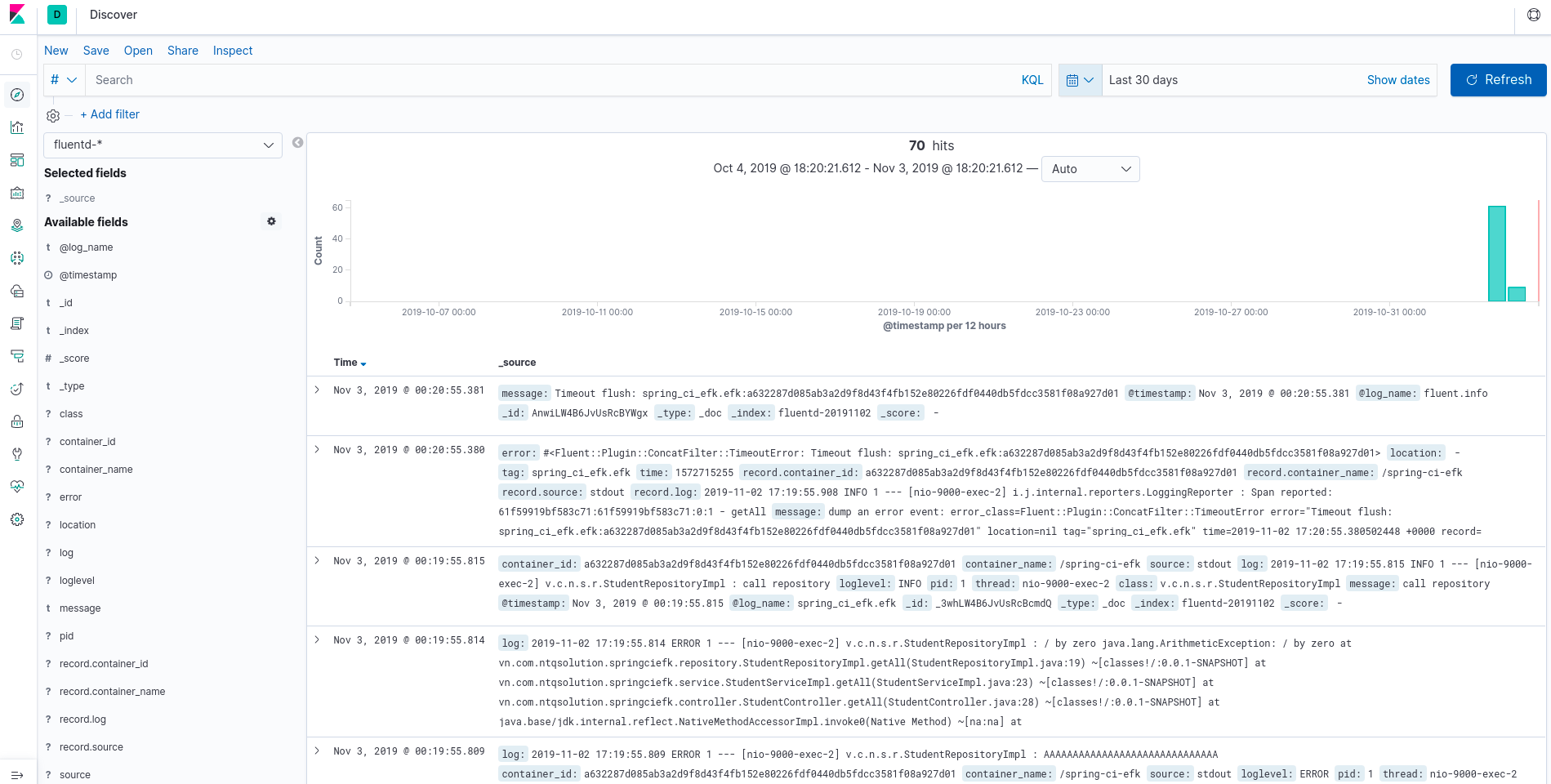
- Mục filter thứ 2 nhằm mục đích parse log thu được thành các thành phần nhỏ để tiện theo dõi và tìm kiếm như: thời gian log, log level, pid, class, thread, message, …

Phần cuối cùng là lưu các log vừa thu được thông qua các filter vào Elasticsearch và in ra terminal



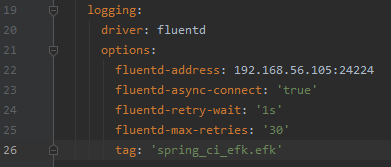
Mục này sẽ có kiểu là copy, và ý nghĩa của nó là copy các log thu được để lưu vào elasticsearch ở cổng 9200 với định dạng của logstash (logstash\_format), tiền tố là fluentd (logstash\_prefix), truyền cả tag key được gán (include\_tag\_key) và đẩy đi cứ 5s một lần.

Sau khi cấu hình project EFK xong, ta thực hiện lệnh ***docker-compose up*** để EFK hoạt động. Vào thử đường dẫn 192.168.1.201:5601 (cổng của kibana) để xem kết quả thu được. Tất nhiên, trước khi show được log thì ta phải thực hiện add thêm index thu được từ Elasticsearch, index đó có tiền tố fluentd-\* mà ta đã config ở trong fluent.conf:



## ***2.2 Cấu hình một service gửi log tới EFK***

Trong mục này, team sẽ thực hiện demo với prioject spring boot cơ bản. Tất cả công việc cần làm ở đây là thêm đoạn cấu hình logging vào file docker-compose.yml để log của project này sẽ được hứng bởi fluentd:



Để xem kết quả, ta truy cập vào Kibana ở địa chỉ 192.168.1.201:5601

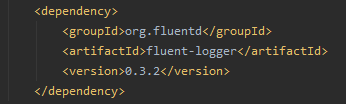
# **3. Ưu nhược điểm, những điều được và chưa được của hệ thống**

## ***3.1 Ưu điểm và những điểm làm được của hệ thống***

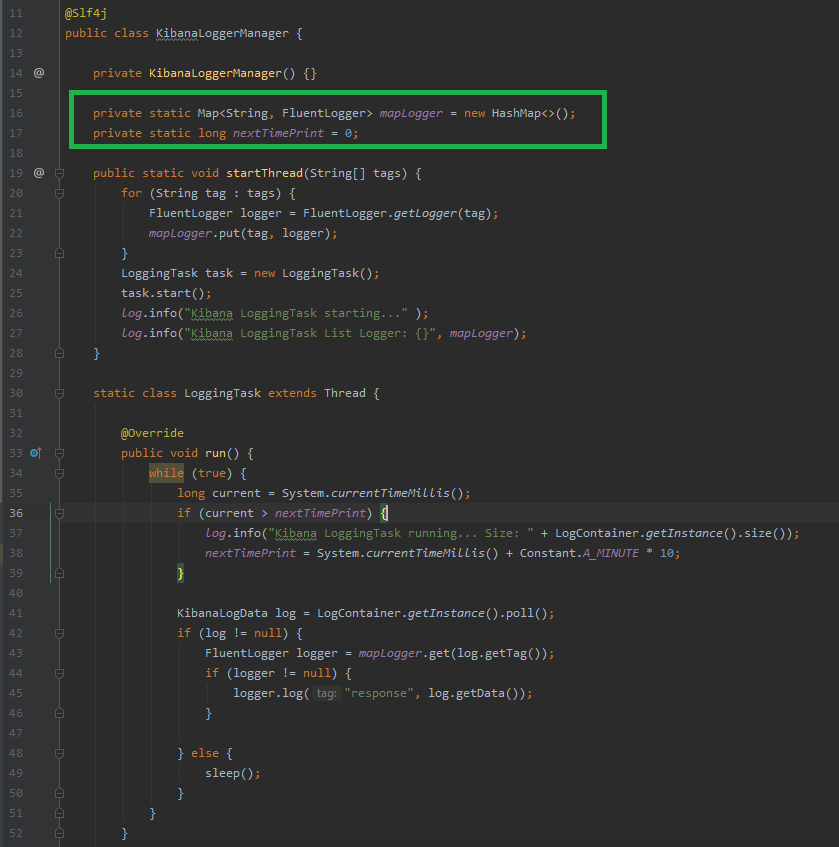
Ưu nhược điểm của hệ thống được xét đến dựa vào sự so sánh với hệ thống EFK có sẵn của team server.

- Thực hiển bắt log từ ứng dụng spring boot một cách tự động thay vì phải add log một các thủ công như trước.

+, Add thư viện



+, Log thủ công



- Sử dụng docker để thực hiện việc build và deploy khiến việc build trở nên dễ dàng hơn bao giờ hết.

## ***3.2 Nhược điểm và những điều chưa làm được của hệ thống***

- Tất cả các java service đều có chung một tiền tố index được config trong file ***fluentd.conf***, chưa tách bạch thành các index riêng biệt như hệ thống cũ, dẫn đến việc tìm kiếm và quản lý khó khăn hơn. Hiện tại vấn đề này chưa có giải pháp cụ thể để giải quyết.

- Chưa tích hợp được x-pack security vào EFK để tăng độ bảo mật cho hệ thống. Hy vọng rằng việc này có thể thực hiện trong thời gian tới [[14]](#_4._Reference).

# **4. References**

[1] <https://blog.vietnamlab.vn/2018/05/30/quan-ly-log-voi-logstash-elasticsearch-kibana/>

[2] <https://toidicodedao.com/2018/03/20/elk-stack-logging/>

[3] [https://www.linkedin.com/pulse/big-data-t%E1%BB%95ng-quan-v%E1%BB%81-elasticsearch-donald-trung-manh-nguyen](https://www.linkedin.com/pulse/big-data-tổng-quan-về-elasticsearch-donald-trung-manh-nguyen)

[4] <https://viblo.asia/p/elasticsearch-khai-niem-va-cac-cau-truy-van-co-ban-djeZ1VwRlWz>

[5] <https://hicubex2k.wordpress.com/2016/11/03/tim-hieu-ve-fluentd/>

[6] <https://stolennguyen.website/kibana-la-gi/>

[7] <https://stackjava.com/elasticsearch>

[8] [https://viblo.asia/p/elasticsearch-distributed-search-ZnbRlr6lG2Xo#replica-shard-6](https://viblo.asia/p/elasticsearch-distributed-search-ZnbRlr6lG2Xo" \l "replica-shard-6)

[9] <https://viblo.asia/p/elasticsearch-la-gi-1Je5E8RmlnL>

[10] <https://viblo.asia/p/phan-1-nhung-khai-niem-co-ban-trong-elasticsearch-V3m5WzzglO7>

[11] <https://viblo.asia/p/series-elasticsearch-elasticsearch-la-gi-cac-khai-niem-trong-elasticsearch-WAyK8VBolxX>

[12] <https://viblo.asia/p/elasticsearch-mapping-a-closer-look-gDVK2kYmZLj>

[13] <https://viblo.asia/p/elasticsearch-mapping-a-closer-look-part-2-RQqKLvMrl7z>

[14] <http://codingfundas.com/setting-up-elasticsearch-6-8-with-kibana-and-x-pack-security-enabled/index.html>

[15] <https://github.com/fluent-plugins-nursery/fluent-plugin-concat>