ĐẠI HỌC QUỐC GIA HÀ NỘI
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN

TIỂU LUẬN

Một số vấn đề về đồ hoạ máy tính
Giám sát hệ thống micro-services và vấn đề logging

Đại học Quốc gia Hà Nội
Đại học Khoa học Tự nhiên
Khoa Toán cơ tin

Giảng viên: TS. Nguyễn Thị Bích Thuỷ

Học viên: Nguyễn Mạnh Linh, Nguyễn Đức Thịnh

Mục lục

Tóm tắt

Ngoài việc lập trình thì vận hành và giám sát là khâu quan trọng trong vòng đời của một sản phẩn công nghệ (phần mềm). Vận hành một sản phẩm không hề dễ dàng hơn việc tạo ra nó. Khi một phần mềm được triển khai thực tế, lập trình viên cũng như người quản trị hệ thống luôn cần nắm rõ *sức khoẻ* của sản phẩm ví dụ như phần mềm đang tiêu tốn bao nhiêu tài nguyên của máy tính, phần mềm xử lý yêu cầu nhanh hay chậm, hay có bao nhiêu lỗi xảy ra trong khung giờ... Những thông số hay độ đo này là thước đo để biết được sản phẩm có đủ tốt hoặc quan trọng hơn là cảnh báo sớm cho người làm phần mềm về những nguy cơ có thể xảy ra. Bài báo cáo này giới thiệu một loạt các kĩ thuật, độ đo, cách kết hợp những công cụ và việc vận dụng khéo léo giữa bản thân phần mềm (ghi log) và các công cụ đó. Trực quan hoá dữ liệu được sử dụng một cách triệt để nhằm tạo một cái nhìn từ tổng quan đến chi tiết giúp cho người vận hành hệ thống cũng như lập trình viên luôn giữ được phần mềm của mình trong tầm kiểm soát.

1 Giới thiệu

Trước hết, báo cáo này không phải một hướng dẫn cụ thể về cài đặt công cụ hay về một chuẩn mực nào đó trong thiết kế hay phát triển phần mềm. Nội dung báo cáo dựa trên kinh nghiệm thực tế của tác giả trong quá trình làm sản phẩm (phần mềm), vậy nên chúng tôi sẽ trình bày một cách *vui vẻ* và *gần gũi* thay vì sử dụng ngôn ngữ *hàn lâm*.

Có nhiều tiêu chuẩn để đánh giá một phần mềm là tốt hay không, ví dụ như hiệu năng cao, ít lỗi, sử dụng tài nguyên hiệu quả... Mỗi người hoặc mỗi sản phẩm có đặc thù riêng để đặt ra tiêu chuẩn riêng. Ngoài ra cũng không tồn tại một sản phẩm nào mà vừa phát triển nhanh vừa chạy ổn định (ít hoặc không có lỗi) lại có hiệu năng cao và sử dụng ít tài nguyên, chúng ta luôn phải đánh đổi. Nhưng điều đó không có nghĩa là chúng ta phải bỏ đi tiêu chuẩn nào đó mà không cố gắng làm cho phần mềm ngày một tốt hơn. Trước khi làm được điều đó, chí ít chúng ta cần phải nắm được phần mềm đang tệ hoặc tốt ở điểm nào. Giám sát hệ thống sinh ra từ đó.

Một hệ thống giám sát, cảnh báo tốt có khả năng chỉ ra những thông tin có giá trị một cách trực quan nhất để người quản trị có thể dựa vào đó để đưa ra định hướng phát triển hoặc sửa chữa những lỗi xảy ra trong phần mềm. Hơn nữa, khi đặt các ngưỡng cảnh báo một cách hợp lý, ta có thể ra phương án sớm để tăng tài nguyên khi hệ thống bị cao tải hoặc có thể biết được hệ thống đang bị tấn công hoặc bị lỗi để có kịch bản ứng cứu sớm.

2 Hệ thống giám sát với Grafana

Trong phần này chúng ta sẽ cùng tìm hiểu về hệ thống giám sát được xây dựng bằng giải pháp của Grafana. Nguyên lý hoạt động, tính năng và case study thực tế sẽ được trình bày để người đọc có cái nhìn tổng quát cũng như nắm bắt được những tình huống thực tế mà doanh nghiệp sử dụng Grafana để xây dựng hệ thống monitoring

2.1 Giới thiệu Grafana

Theo giới thiệu trên trang chủ của Grafana "Truy vấn, trực quan hóa, cảnh báo và hiểu dữ liệu của bạn không quan trọng nó được lưu trữ ở đâu. Với Grafana bạn có thể tạo, khai phá, chia sẻ dữ liệu của mình thông qua những dashboard đẹp và linh động"

Hình 2.1: Một dashboard demo trên trang chủ của Grafana

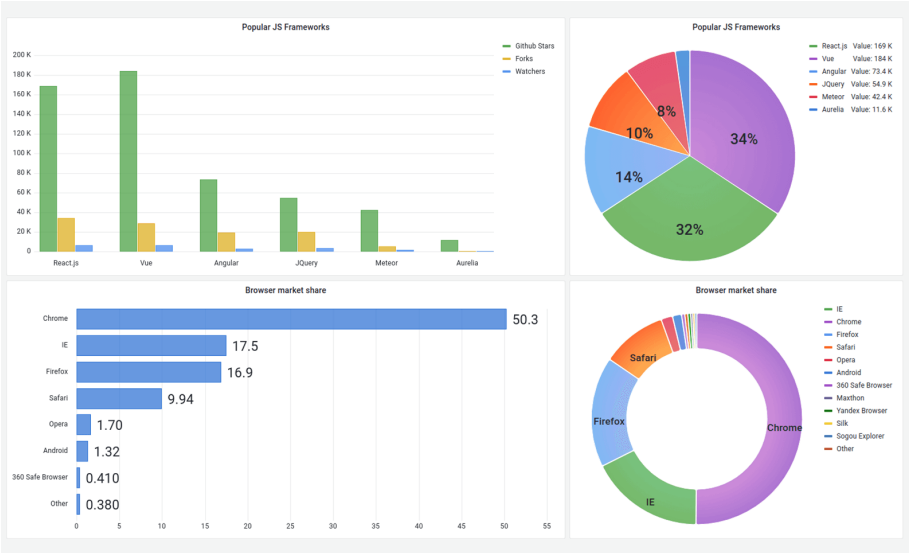
Biểu đồ trên cùng bên trái thể hiện lượng requests tới một website nào đó trong khoảng thời gian một giờ gần nhất. Không chỉ thể hiện tổng lượng request, ta còn có thể quan sát thấy thông lượng tới từng node từ web_server_01 đến web_server_04 thông qua lớp phủ màu. Ngoài ra khi giữ chuột vào một thời điểm nhất định, biểu đồ cũng cho biết số lượng request cụ thể vào từng node vào thời điểm đó.

Biểu đồ trên cùng bên phải thể hiện tài nguyên Memory và CPU được sử dụng của ứng dụng.

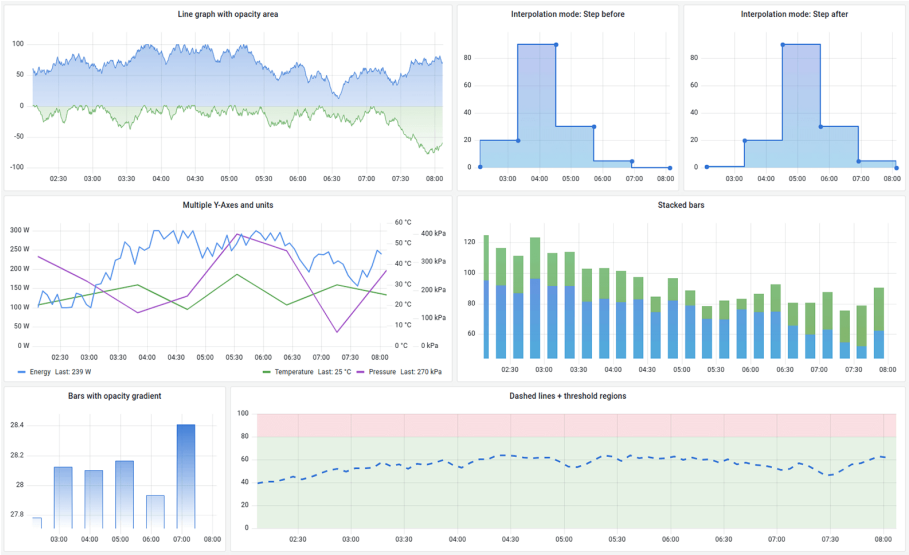
2.2 Các tính năng chính

2.2.1 Panels

Grafana cung cấp các bảng biểu, đồ thị đa dạng bao gồm bản đồ nhiệt, histograms, đồ thị, biểu đồ địa lý để trực quan hoá dữ liệu một cách linh động theo bất kì cách nào chúng ta muốn.

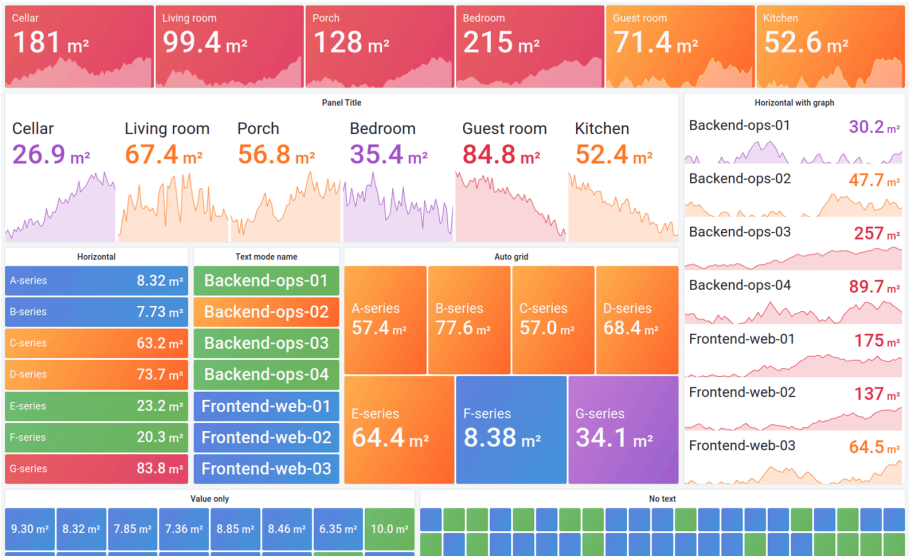


Hình 2.2: Biểu đồ cột và biểu đồ tròn



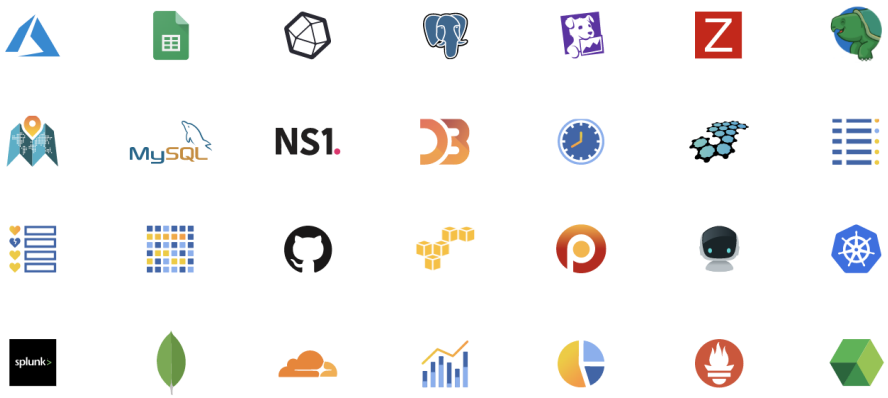
Hình 2.3: Biểu đồ dữ liệu hướng thời gian

Hình 2.4: Biểu dữ liệu dạng bảng



Hình 2.5: Biểu đồ thống kê

2.2.2 Plugins



Hình 2.6: Các tiện ích mở rộng

Grafna không yêu cầu phải lưu dữ liệu vào một cơ sở dữ liệu tập trung, thay vào đó nó cung cấp các tiện ích mở rộng cho phép ta kết nối tới nhiều loại cơ sở dữ liệu khách nhau. Ngoài ra Grafana cũng cho phép kết nối tới các công cụ nội bộ, cho phép doanh nghiệp triển khai một cách độc lập mà không cần phải thông qua một bên thứ 3 nữa. Ví dụ chúng ta có thể phân quyền người dùng với hệ thống xác thực tập trung riêng của doanh nghiệp hoặc cũng có thể tuỳ chỉnh hệ thống cảnh báo để gửi tin về hệ thống nhắn tin riêng mà không cần thông qua những hệ thống chung như email.

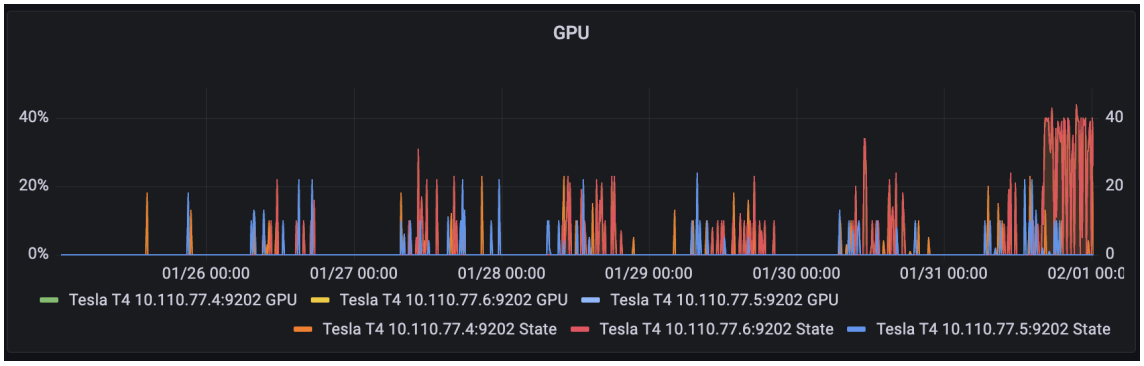
2.3 Case study

Trong phần này chúng ta sẽ cùng theo dõi và phân tích một dashboard thực tế mà tác giả cùng đội của mình xây dựng để giám sát hệ thống OCR đọc thông tin từ giấy tờ của người dùng. Tính năng chính của hệ thống là nhận vào url ảnh giấy tờ (trên hệ thống lưu trữ tập trung của doanh nghiệp) và trích xuất thông tin. Những thông số quan trọng nhất trong ứng dụng kiểu này đó là tài nguyên hệ thống, thời gian xử lý và phản hồi của các hệ thống có kết nối đến.

Dữ liệu giám sát phần cứng được Grafana giao tiếp với hệ điều hành, còn các dữ liệu do người lập trình tự định nghĩa được lưu trữ dưới dạng log ở Elasticsearch (sẽ được nói đến kĩ hơn ở phần sau của bài báo cáo).

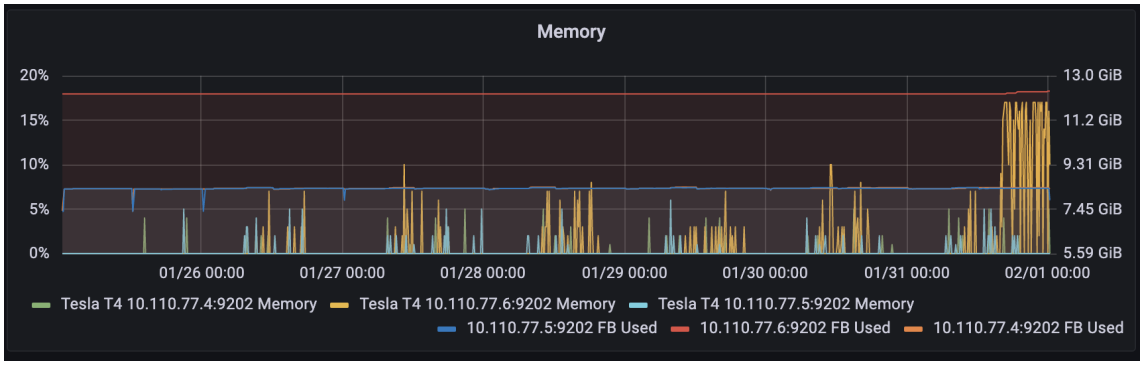
Hình 2.7: Thời gian ứng dụng trích xuất thông tin từ ảnh

Trên hình là biểu đồ theo thời gian của thời gian trung bình mà ứng dụng trích xuất kí tự từ ảnh. Biểu đồ đường được sử dụng kết hợp với lớp phủ màu. Thời gian xử lý chuyển từ sắc xanh, vàng sang cam, đỏ. Lớp phủ màu được thêm vào để tiện cho người dùng quan sát tổng quan, càng nhiều màu đỏ nghĩa là thời gian tính toán càng lâu, ngược lại, khi màu xanh chiếm đa số có nghĩa là ứng dụng đang tính toán nhanh. Trong khoảng thời gian được cắt ra trên hình ta có thể nhận thấy thời gian tính toán trung bình vào khoảng 0.5s và thời gian xử lý lâu nhất vào khoảng 1s. Điều này đang được chấp nhận trong thực tế do tài nguyên phần cứng của ứng dụng OCR này đang được dùng chung cho nhiều ứng dụng khác nữa. Hiện đội đang sử dụng GPU là Tesla T4 khá cũ.



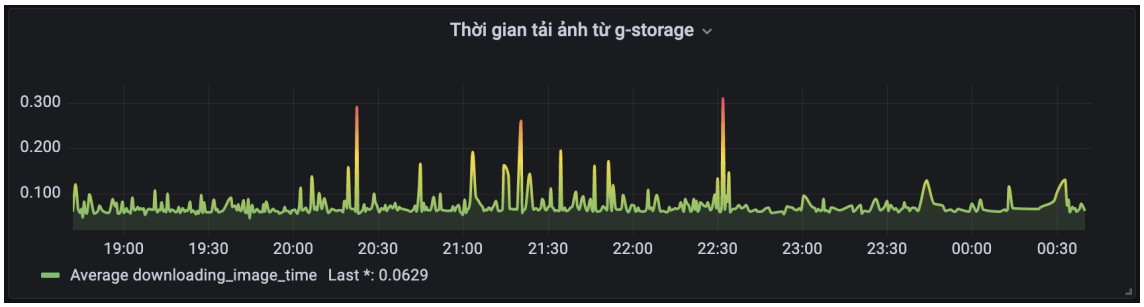
Hình 2.8: Mức độ sử dụng GPU trong 7 ngày gần nhất

Biểu đồ thể hiện mức độ sử dụng GPU được vẽ theo ngày (đương nhiên khoảng thời gian có thể được điều chỉnh). Mức độ sát với thực tế khi GPU được sử dụng ít khi người dùng mới trở lại sau kì nghỉ Tết. Chúng ta hoàn toàn có thể đặt ngưỡng cảnh báo khi GPU bị cao tải (vượt ngưỡng 90%)



Hình 2.9: Mức độ sử dụng memory của GPU trong 7 ngày gần nhất

Tương ứng với mức độ sử dụng GPU ta cũng có biểu đồ về dung lượng VRAM được sử dụng. Ngưỡng cảnh báo tương tự cũng được áp dụng (90%)



Hình 2.10: Thời gian tải ảnh

Thời gian ứng dụng tải ảnh từ storage lưu trữ tập trung cũng được thể hiện bằng biểu đồ đường tương tự như những biểu đồ khác.

Có thể nói Grafana được đội của tác giả sử dụng nhằm mục đính nắm tổng quan về hệ thống và đưa ra cảnh báo sớm. Các thông tin chi tiết hơn về ứng dụng hoặc một lỗi cụ thể sẽ được khám phá kĩ càng hơn với ELK Stack (sẽ được trình bày ở phần sau vủa bài).

3 Quản lý, giám sát và logging

3.1 Giới thiệu ELK-stack

ELK Stack là từ viết tắt được sử dụng để mô tả bộ dịch vụ bao gồm 3 dự án phổ biến: Elasticsearch, Logstash và Kibana. Thường được gọi là Elasticsearch, ELK Stack mang tới khả năng tổng hợp log từ tất cả các hệ thống và ứng dụng của bạn, phân tích những log này, hiển thị dữ liệu để giám sát ứng dụng và cơ sở hạ tầng, khắc phục sự cố nhanh hơn, phân tích bảo mật, v.v. Không quan trọng webserver sử dụng NGINX, Apache hay MSServer, không quan trọng log đến từ ứng dụng, database, cache server... chúng đều có thể được xử lý bởi ELK!

Hình 3.1: ELK stack

- *E = Elasticsearch*: Elasticsearch là công cụ tìm kiếm và phân tích phân tán được xây dựng trên Apache Lucene. Khả năng hỗ trợ đa dạng ngôn ngữ, hiệu suất cao và JSON doccument phi cấu trúc khiến Elasticsearch trở thành một lựa chọn lý tưởng cho nhiều trường hợp sử dụng tìm kiếm và phân tích log khác nhau.
- *L = Logstash*: Logstash là một công cụ thu nạp dữ liệu nguồn mở cho phép thu thập dữ liệu từ các nguồn khác nhau, chuyển đổi dữ liệu và gửi dữ

liệu tới điểm đích. Với các bộ lọc được tạo sẵn và hỗ trợ hơn 200 phần bổ trợ, Logstash cho phép người dùng dễ dàng thu nạp bất kỳ dữ liệu đến từ nguồn hay thuộc loại dữ liệu nào.

- *K = Kibana*: Kibana là một công cụ hiển thị trực quan và khám phá dữ liệu dành cho hoạt động đánh giá log và sự kiện. Kibana cung cấp các biểu đồ tương tác dễ sử dụng, các bộ lọc được tạo sẵn cũng như hỗ trợ không gian địa lý.

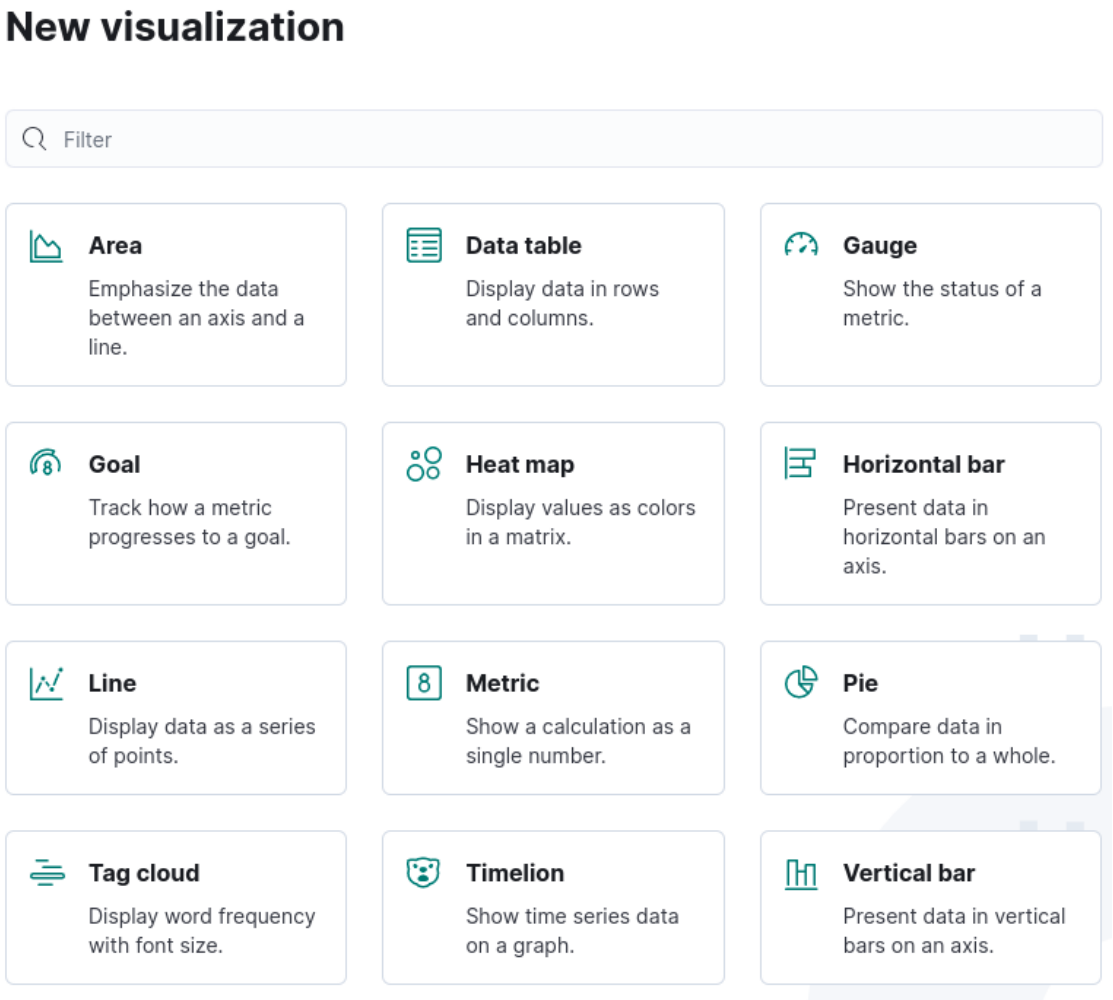
Hình 3.2: ELK stack với log ứng dụng

Nói một cách đơn giản thì Logstash là nơi tiếp nhận dữ liệu từ nhiều nguồn sau đó có thể biến đổi hoặc lọc bớt dữ liệu và ghi vào Elasticsearch. Elasticsearch vừa là nơi lưu trữ vừa là công cụ có khả năng tìm kiếm và phân tích. Kibana là công cụ trực quan hoá dữ liệu từ Elasticsearch, ngoài ra nó cũng cung cấp giao diện đơn giản để thực hiện truy vấn, phân tích với Elasticsearch dễ dàng hơn.

Câu hỏi đặt ra là tại sao ta lại cần một hệ thống *cồng kềnh* như Elasticsearch? Đương nhiên với hệ thống nhỏ, ta chỉ cần ghi log ra file là xong, tuy nhiên với những hệ thống lớn, triển khai trên nhiều servers với nhiều services khác nhau ta không thể nào truy cập vào từng instance của từng service để đọc log, điều tra lỗi. Mặt khác log chứa rất nhiều thông tin của ứng dụng mà đôi khi ta không phải lúc nào cũng muốn ghi xuống cơ sở dữ liệu (thường chậm hơn nhiều so với việc ghi log). Vậy để tổng hợp, phân tích những metrics đó, những công cụ mặc định của hệ điều hành là không đủ. ELK sinh ra để giải quyết tất cả những vấn đề đó bằng việc tập trung toàn bộ log vào Elasticsearch, nơi có thể trực quan hóa, tổng hợp, phân tích, tìm kiếm một cách hết sức thuận tiện. Ngoài ra với hiệu năng truy vấn đáng nể, Elasticsearch còn có thể được sử dụng như một *search engine*. Tuy nhiên trong khuôn khổ bài báo cáo này, tác giả chỉ tập trung đến khả năng thu thập, trực quan hóa và case study về điều tra lỗi thay vì đề cập nhiều đến khả năng truy vấn (vốn là tính năng mạnh mẽ nhất của Elasticsearch).

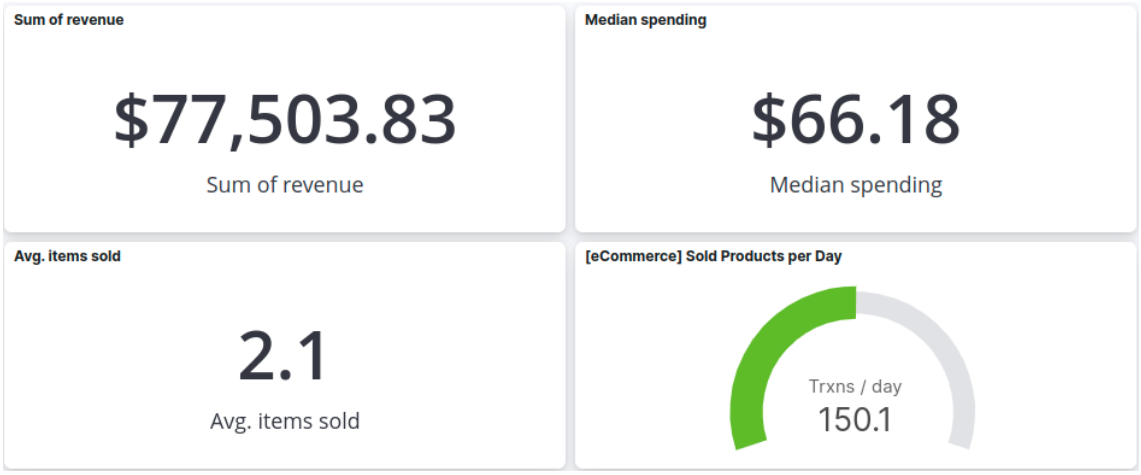
3.2 Trực quan hóa dữ liệu với Kibana

Trong phần này, chúng ta sẽ điểm qua những tính năng chính (trực quan hóa) của Kibana bằng cách khám phá tập dữ liệu mẫu (sẵn có của Elasticsearch) và xây dựng dashboard trên các tập dữ liệu này.



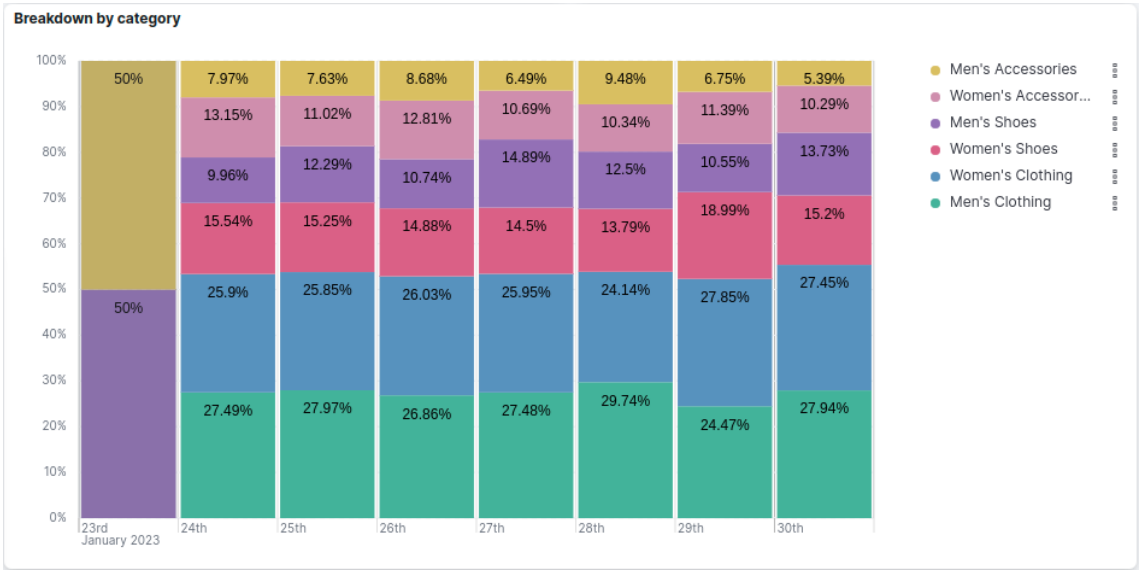
Hình 3.3: Các biểu đồ được Kibana dựng sẵn

Với một ứng dụng cổ điển, thường thì các lập trình viên sẽ phải xây dựng một trang web quản trị riêng nhằm cung cấp các biểu đồ trực quan để admin nắm được những thông tin cần thiết. Tuy nhiên việc này là khá tốn kém chi phí và công sức do người lập trình ngoài việc phải code phần giao diện còn phải xây dựng các API cần thiết cũng như ghi vào cơ sở dữ liệu các thông tin theo yêu cầu để hiển thị. Kibana giảm thiểu công sức cho chúng ta rất nhiều bằng cách cung cấp sẵn những biểu đồ đủ các thể loại từ biểu đồ diện tích, biểu đồ nhiệt, biểu đồ cột, biểu đồ đường, thậm chí cả biểu đồ địa lý... Mà quan trọng là ta không cần phải viết code hay truy vấn vào cơ sở dữ liệu (có thể dẫn đến quá tải hệ thống).



Hình 3.4: Bảng tổng hợp lợi nhuận (*tập dữ liệu eCommerce dựng sẵn của Elasticsearch*)

Để nắm tổng quan về tình hình kinh doanh, người quản trị có thể theo dõi biểu đồ tổng hợp với những con số được làm nổi bật.



Hình 3.5: Một biểu đồ đơn giản thể hiện phần trăm các loại mặt hàng được bán theo ngày của một website bán hàng trực tuyến (*tập dữ liệu eCommerce dựng sẵn của Elasticsearch*)

Nhìn vào biểu đồ, người quản trị có thể nắm được thông tin tổng quan là mặt hàng quần áo nam và nữ được bán với tỉ trọng lớn và khá đều đặn qua các ngày. Trong khi đó mặt hàng phụ kiện thì bán được khá ít. Lớp phủ màu được sử dụng để người xem dễ theo dõi tỉ trọng của mặt hàng. Ngoài ra tại mỗi khối, tỉ trọng cũng được ghi dưới dạng phần trăm để người xem nắm được con số cụ thể.

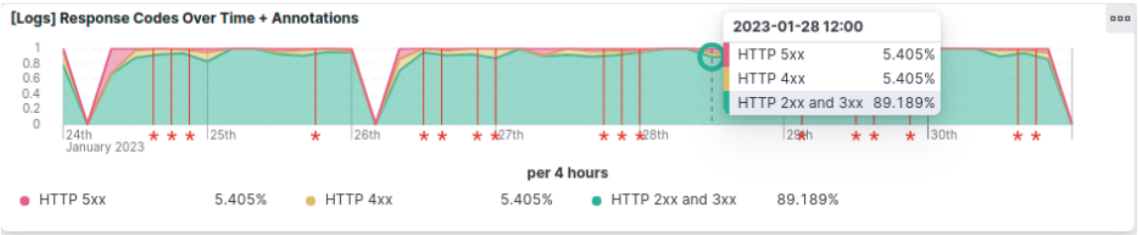
Daily comparison

order_date per day	This week	1 week ago	Difference
2023-01-23	$172	$0	172.00
2023-01-24	$11,205.5	$0	11,205.50
2023-01-25	$10,777.58	$0	10,777.58
2023-01-26	$10,925.23	$7,436.39	3,488.84
2023-01-27	$13,020.13	$9,938.16	3,081.98
2023-01-28	$10,487.83	$10,430.16	57.67
2023-01-29	$11,255.44	$10,973.65	281.79
2023-01-30	$9,660.12	$11,284.56	-1,624.45

Hình 3.6: So sánh doanh thu của tuần này và tuần trước (*tập dữ liệu eCommerce dựng sẵn của Elasticsearch*)

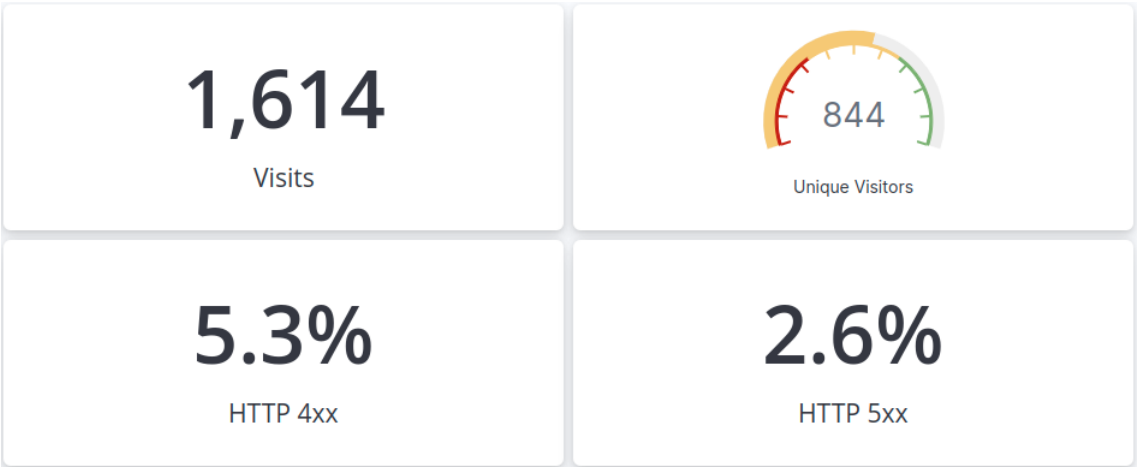
Một ví dụ khác về việc áp dụng màu sắc và con số. Màu xanh thể hiện doanh thu tăng và ngược lại màu đỏ thể hiện sự sụt giảm doanh thu. Từ đó người quản trị có thể điều tra nguyên nhân và đưa ra quyết định kinh doanh phù hợp.

Kibana là một tool khá mạnh về biểu diễn dữ liệu hướng thời gian bởi lý do rất tự nhiên, trong thực tế doanh nghiệp thường sử dụng ELK làm hệ thống quản lý log mà log là một loại dữ liệu hướng thời gian điển hình. Một trong số những độ đo quan trọng nhất mà một website hay một ứng dụng web cần quan tâm đó là phản hồi của ứng dụng với yêu cầu từ khách hàng (*http status*).



Hình 3.7: HTTP Status của một ứng dụng web (*tập dữ liệu Web Traffic dựng sẵn của Elasticsearch*)

Chúng ta thường kì vọng số lỗi 4xx hoặc 5xx là ít nhất có thể, đặc biệt là 5xx (các lỗi hệ thống). Bởi lẽ lỗi 500 là đại diện của sự mất kiểm soát, có thể trong quá trình lập trình ta không bắt hết các ngoại lệ hay ứng dụng bị crash, dịch vụ ngoài chết... Đây được coi là lỗi nguy hiểm mà ta cần ứng cứu ngay lập tức. Kibana cung cấp một biểu đồ theo thời gian rất trực quan mà chúng ta có thể quan sát thấy ngay những lỗi nguy hiểm (màu đỏ - do người xây dựng biểu đồ cấu hình). Ngoài ra ta cũng có thể tương tác với biểu đồ bằng cách giữ chuột ở một vị trí nhất định để biết được tỉ lệ của các mã phản hồi HTTP trong một khoảng thời gian nào đó.

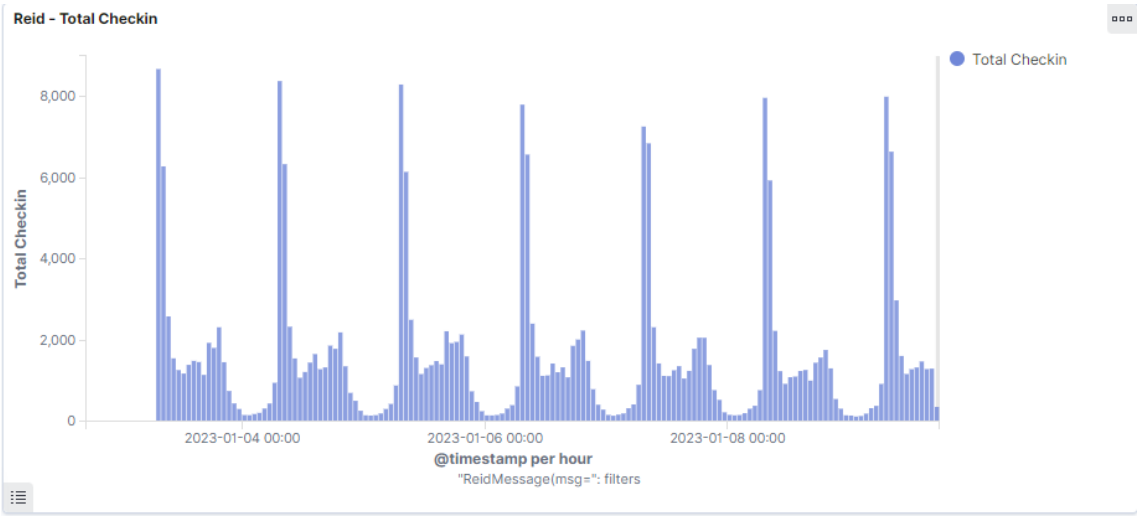


Hình 3.8: Tổng quan HTTP Status (*tập dữ liệu Web Traffic dựng sẵn của Elasticsearch*)

Tương tự ta cũng có thể xây dựng biểu đồ tổng quan để người dùng nắm được số HTTP Status đại diện cho lỗi trong khoảng thời gian cấu hình để có hướng điều tra hoặc cải thiện.

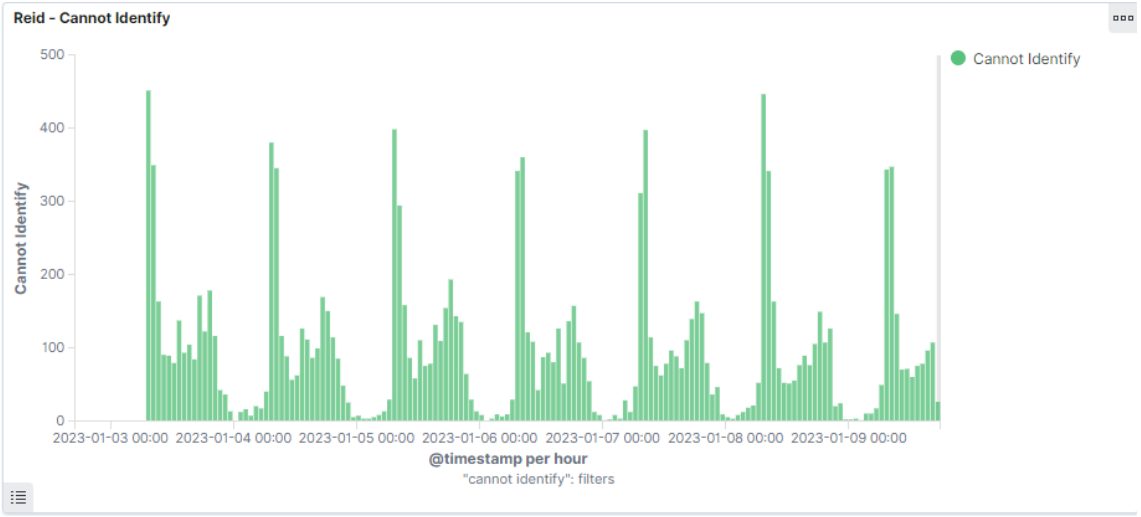
3.3 Case study

Trong phần này, chúng ta sẽ cùng xem xét tình huống thực tế mà tác giả đã sử dụng ELK stack để theo dõi và cải thiện một ứng dụng điểm danh bằng khuôn mặt. Tính năng cốt lõi của ứng dụng là nhận vào một hình ảnh khuôn mặt được truyền từ thiết bị di động và tìm kiếm trong cơ sở dữ liệu thông tin của ảnh đó. Thông tin trả ra là một ID mà sẽ được dùng để tham chiếu tới thông tin cá nhân của người điểm danh. Nếu tìm thấy ảnh trong cơ sở dữ liệu (điểm trên một ngưỡng nào đó), ứng dụng sẽ ghi ra một dòng log có dạng `"reid_data: ReidMessage(msg='success', score=0.8236697316169739)"`. Ngược lại, khi độ đo score dưới ngưỡng (có thể coi là không tìm thấy ảnh trong cơ sở dữ liệu), log có dạng `"reid_data: ReidMessage(msg='cannot identify', score=0.2316697316169739)"`.

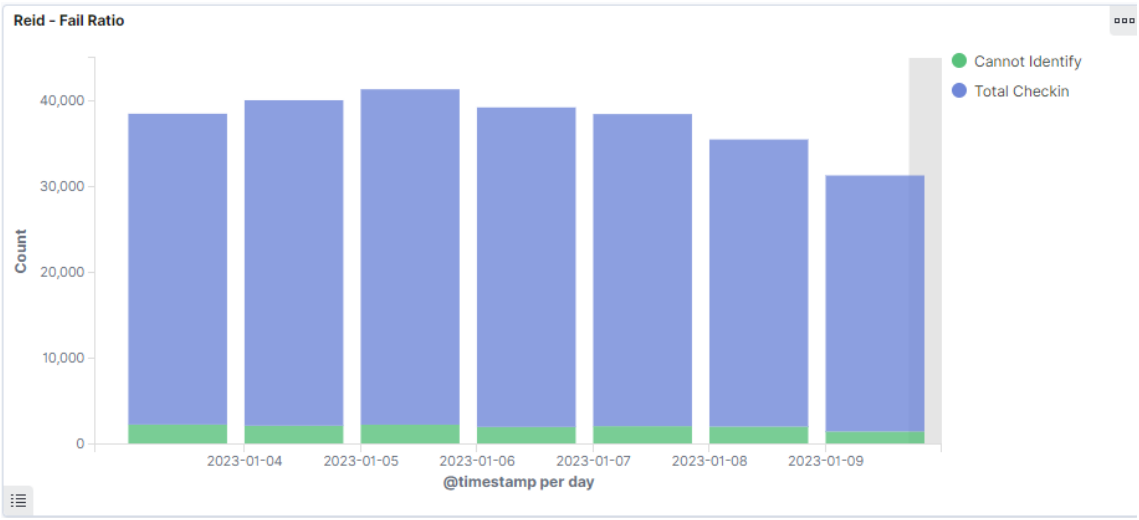


Hình 3.9: Log điểm danh trong 7 ngày gần nhất

Với khả năng tìm kiếm full text mạnh mẽ của Elasticsearch, chúng ta có thể áp dụng bộ lọc lấy ra những log có chứa cụm từ `"ReidMessage(msg="` nghĩa là bao trùm cả trường hợp điểm danh thành công hoặc thất bại, hay nói cách khác chính là log điểm danh mà hệ thống ghi nhận được. Tương tự như vậy ta có thể lọc ra những log chứa cụm từ `"ReidMessage(msg='cannot identify'"` để lấy ra các log điểm danh thất bại và trực quan hoá với biểu đồ dưới đây. Do dữ liệu là dạng dữ liệu hướng thời gian nên tác giả chọn biểu diễn bằng biểu đồ cột. Ngoài ra ta cũng có thể giữ chuột tại một vị trí nào đó để biết số lượng điểm danh (thất bại) trong một khung thời gian. Khoảng thời gian hoàn toàn có thể được tuỳ chỉnh một cách dễ dàng với thanh điều khiển của Kibana.

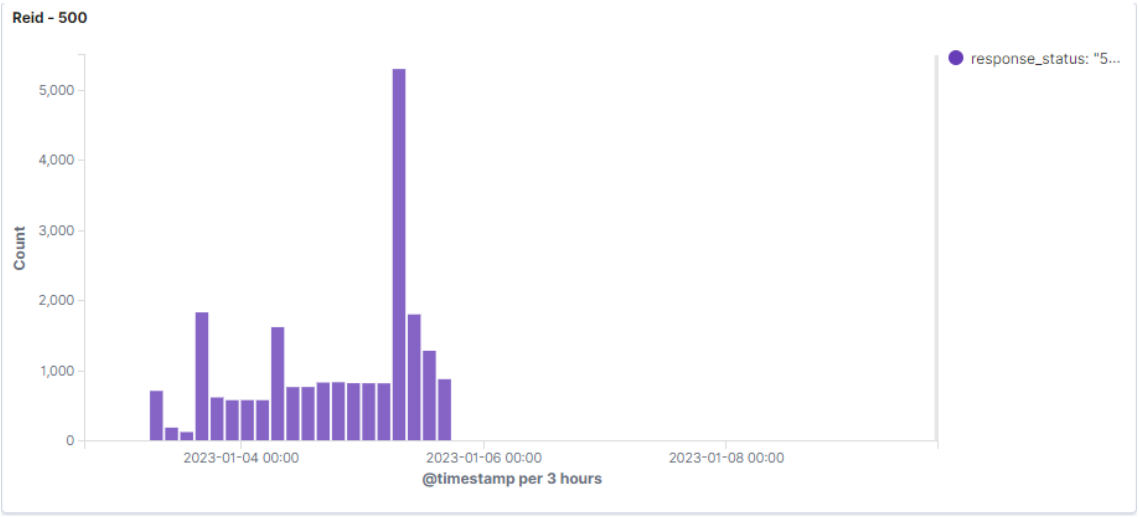


Hình 3.10: Log điểm danh thất bại trong 7 ngày gần nhất



Hình 3.11: Tỉ lệ điểm danh thất bại trên tổng log điểm danh trong 7 ngày gần nhất

Quan sát biểu đồ, ta có thể nhận thấy số khung giờ cao điểm là khoảng 7 đến 8 giờ sáng và 17 đến 19 giờ tương ứng với thời điểm checkin và checkout của nhân viên. Ban đêm hầu như không có log điểm danh. Tỉ lệ điểm danh lỗi hàng ngày vào khoảng 6-8%. Câu hỏi đặt ra là liệu người làm ứng dụng có thể làm tốt hơn? Đôi khi vài phần trăm không phải là vấn đề lớn trong tính toán nhưng với những nhân viên thường xuyên điểm danh thất bại hoặc thời gian điểm danh lâu thì đó là một trải nghiệm rất tệ và ảnh hưởng trực tiếp tới ngày công của họ.

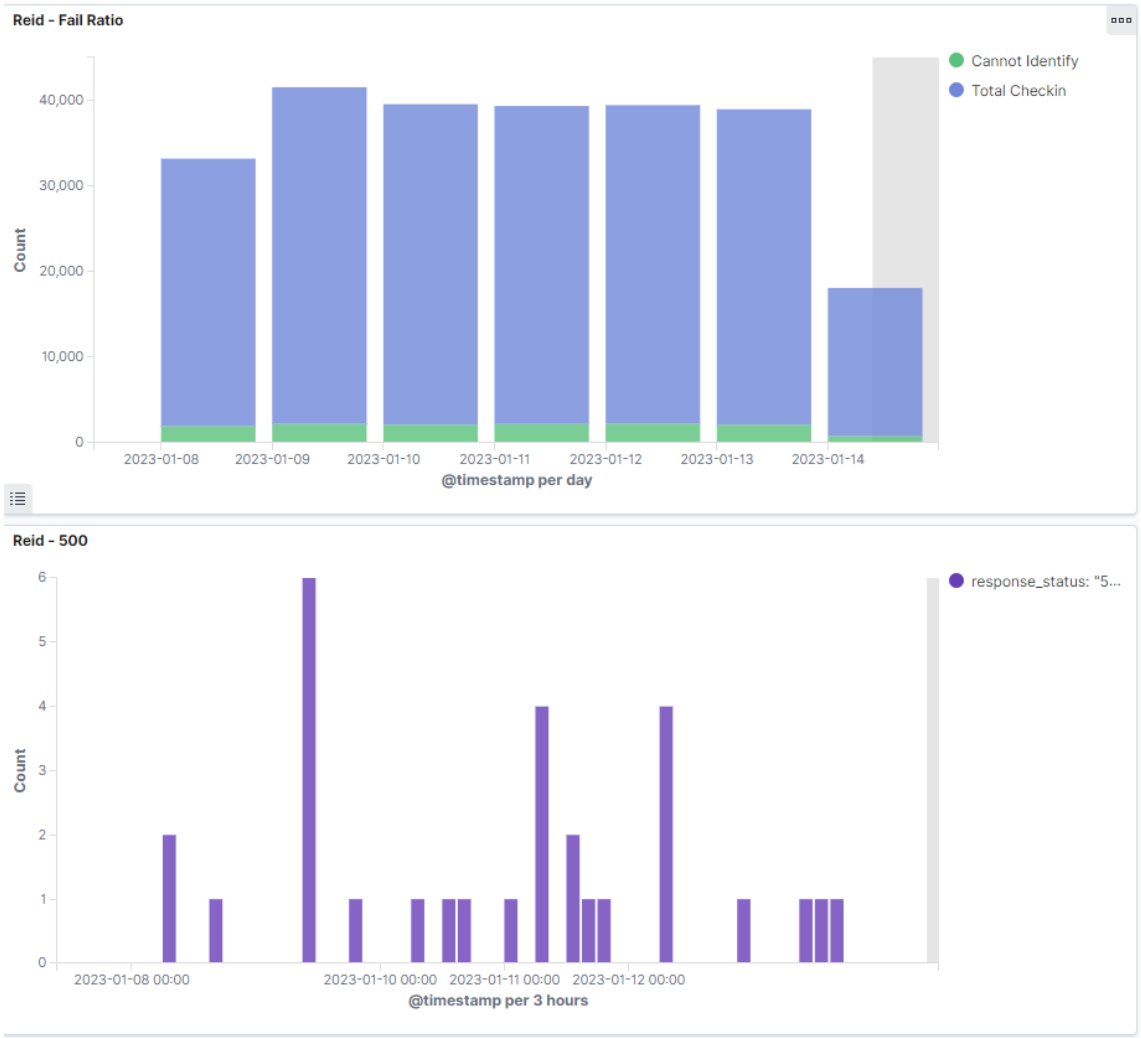


Hình 3.12: Lỗi 500 xảy ra trong hệ thống

Và đây là điều bất ngờ khi tác giả biểu diễn số request bị lỗi 500. Như đã nói ở phần trước, lỗi phản hồi HTTP 500 là một lỗi nghiêm trọng vì chúng ta ...không biết nguyên nhân của nó! Ngay khi vừa vẽ biểu đồ biểu diễn số lượng lỗi 500, chúng tôi đã vô cùng hoang mang vì trước nay tất cả đều tưởng ứng dụng đang chạy ...tốt!

Khi quan sát các biểu đồ trên và xếp kề theo một trục thẳng đứng, chúng ta thấy có một sự trùng hợp đó là thời điểm số log lỗi 500 trùng với thời điểm cao tải điểm danh. Tiếp tục truy vết và nghiên cứu code chúng tôi phát hiện ra rằng, khi có quá nhiều request vào hệ thống, ứng dụng bị nghẽn! Khi đó ứng dụng không thể tiếp nhận thêm yêu cầu và trả ra lỗi 500. Vậy là sau vài giây đứng chờ người dùng sẽ nhận được thông báo điểm danh thất bại. Đây quả thực là một trải nghiệm tệ.

Để cải thiện, ban đầu, giải pháp được đưa ra là tăng phần cứng lên, ghép hêm CPU, GPU và tăng số lượng instances của ứng dụng. Nhưng chúng ta đều biết đây là phương án rất đắt đỏ bởi GPU là phần cứng không hề rẻ tiền và không sẵn có. Cuối cùng chúng tôi nghiên cứu code và chỉnh sửa và đây là kết quả. Số lỗi từ đỉnh trên 5000 ngày hôm trước còn dưới 10 vào ngày hôm sau!



Hình 3.13: Lỗi 500 xảy ra trong hệ thống sau khi sửa code

Chỉ bằng những biểu diễn trực quan hết sức ...thô sơ chúng tôi đã phát hiện lỗi và cải thiện ứng dụng một cách nhanh chóng và mang lại lợi ích đáng kể (thay vì phải mua thêm khoảng 4 GPU trị giá khoảng gần 200 triệu đồng).

4 Kết luận

Bài báo cáo nhấn mạnh tầm quan trọng của trực quan hoá dữ liệu trong việc giám sát và vận hành một hệ thống phần mềm. Hai bộ công cụ nổi tiếng là Grafana và ELK được giới thiệu và xem xét các tình huống sử dụng trong thực tế. Grafana và ELK có khá nhiều tính năng tương đồng nhưng cũng có những tính năng nổi bật riêng mà thực tế chúng thường được sử dụng kết hợp với nhau. Bài báo cáo cũng chỉ ra rằng không nhất thiết phải sử dụng những bảng biểu, đồ thị quá phức tạp mà đôi khi sự đơn giản lại mang đến hiệu quả to lớn. Không chỉ dừng lại ở khuôn khổ một bài báo cáo về trực quan hoá dữ liệu, tác giả đã giới thiệu một giải pháp để giám sát và vận hành hệ thống phần mềm.

Tài liệu tham khảo

[1] Chen, Pin-Yu, et al. "Ead: elastic-net attacks to deep neural networks via adversarial examples." Proceedings of the AAAI conference on artificial intelligence. Vol. 32. No. 1. 2018.

[2] Shao, Weijia, Fikret Sivrikaya, and Sahin Albayrak. "Optimistic Optimisation of Composite Objective with Exponentiated Update." (2022).

[3] Beck, Amir, and Marc Teboulle. "A fast iterative shrinkage-thresholding algorithm for linear inverse problems." SIAM journal on imaging sciences 2.1 (2009): 183-202.

[4] Candès, Emmanuel J., and Michael B. Wakin. "An introduction to compressive sampling." IEEE signal processing magazine 25.2 (2008): 21-30.

[5] Carlini, Nicholas, and David Wagner. "Adversarial examples are not easily detected: Bypassing ten detection methods." Proceedings of the 10th ACM workshop on artificial intelligence and security. 2017.

[6] Carlini, Nicholas, and David Wagner. "Towards evaluating the robustness of neural networks." 2017 ieee symposium on security and privacy (sp). Ieee, 2017.

[7] Dong, Yinpeng, et al. "Towards interpretable deep neural networks by leveraging adversarial examples." arXiv preprint arXiv:1708.05493 (2017).

[8] Duchi, John, and Yoram Singer. "Efficient online and batch learning using forward backward splitting." The Journal of Machine Learning Research 10 (2009): 2899-2934.

[9] Evtimov, Ivan, et al. "Robust physical-world attacks on machine learning models." arXiv preprint arXiv:1707.08945 2.3 (2017): 4.

[10] Feinman, Reuben, et al. "Detecting adversarial samples from artifacts." arXiv preprint arXiv:1703.00410 (2017).

[11] Fu, Haoying, et al. "Efficient minimization methods of mixed l2-l1 and l1-l1 norms for image restoration." SIAM Journal on Scientific computing 27.6 (2006): 1881-1902.

[12] Goodfellow, Ian J., Jonathon Shlens, and Christian Szegedy. "Explaining and harnessing adversarial examples." arXiv preprint arXiv:1412.6572 (2014).

[13] Grosse, Kathrin, et al. "On the (statistical) detection of adversarial examples." arXiv preprint arXiv:1702.06280 (2017).

[14] Hinton, Geoffrey, Oriol Vinyals, and Jeff Dean. "Distilling the knowledge in a neural network (2015)." arXiv preprint arXiv:1503.02531 2 (2015).

[15] Kingma, Diederik P., and Jimmy Ba. "Adam: A method for stochastic optimization." arXiv preprint arXiv:1412.6980 (2014).

[16] Koh, Pang Wei, and Percy Liang. "Understanding black-box predictions via influence functions." International conference on machine learning. PMLR, 2017.

[17] Kurakin, Alexey, Ian J. Goodfellow, and Samy Bengio. "Adversarial examples in the physical world." Artificial intelligence safety and security. Chapman and Hall/CRC, 2018. 99-112.

[18] Kurakin, Alexey, Ian Goodfellow, and Samy Bengio. "Adversarial machine learning at scale." arXiv preprint arXiv:1611.01236 (2016).

[19] Liu, Yanpei, et al. "Delving into transferable adversarial examples and black-box attacks." arXiv preprint arXiv:1611.02770 (2016).

[20] Lu, J.; Issaranon, T.; and Forsyth, D. 2017. Safetynet: Detecting and rejecting adversarial examples robustly

[21] Madry, Aleksander, et al. "Towards deep learning models resistant to adversarial attacks." arXiv preprint arXiv:1706.06083 (2017).

[22] Moosavi-Dezfooli, Seyed-Mohsen, et al. "Universal adversarial perturbations." Proceedings of the IEEE conference on computer vision and pattern recognition. 2017.

[23] Moosavi-Dezfooli, Seyed-Mohsen, Alhussein Fawzi, and Pascal Frossard. "Deepfool: a simple and accurate method to fool deep neural networks." Proceedings of the IEEE conference on computer vision and pattern recognition. 2016.

[24] Papernot, Nicolas, et al. "The limitations of deep learning in adversarial settings." 2016 IEEE European symposium on security and privacy (EuroS&P). IEEE, 2016.

[25] Papernot, Nicolas, et al. "Distillation as a defense to adversarial perturbations against deep neural networks." 2016 IEEE symposium on security and privacy (SP). IEEE, 2016.

[26] Papernot, Nicolas, et al. "Practical black-box attacks against machine learning." Proceedings of the 2017 ACM on Asia conference on computer and communications security. 2017.

[27] Parikh, Neal, and Stephen Boyd. "Proximal algorithms." Foundations and trends® in Optimization 1.3 (2014): 127-239.

[28] Szegedy, Christian, et al. "Intriguing properties of neural networks." arXiv preprint arXiv:1312.6199 (2013).

[29] Szegedy, Christian, et al. "Rethinking the inception architecture for computer vision." Proceedings of the IEEE conference on computer vision and pattern recognition. 2016.

[30] Tramèr, Florian, et al. "Ensemble adversarial training: Attacks and defenses." arXiv preprint arXiv:1705.07204 (2017).

[31] Xu, Weilin, David Evans, and Yanjun Qi. "Feature squeezing: Detecting adversarial examples in deep neural networks." arXiv preprint arXiv:1704.01155 (2017).

[32] Zantedeschi, Valentina, Maria-Irina Nicolae, and Ambrish Rawat. "Efficient defenses against adversarial attacks." Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security. 2017.

[33] Zheng, Stephan, et al. "Improving the robustness of deep neural networks via stability training." Proceedings of the ieee conference on computer vision and pattern recognition. 2016.

[34] Zou, Hui, and Trevor Hastie. "Regularization and variable selection via the elastic net." Journal of the royal statistical society: series B (statistical methodology) 67.2 (2005): 301-320.