NGINX

I. LỜI MỞ ĐẦU

- Phát triển ứng dụng web hiện tại vẫn đang là 1 lĩnh vực rất hot trong ngành công nghệ thông tin và tương lai vẫn sẽ như vậy. Hãy giả sử rằng, bạn vừa hoàn thành việc code ra 1 trang web của riêng mình, bây giờ bạn muốn có 1 web server để host trang web của mình lên đó cho mọi người cùng sử dụng. Khi bạn search trên Google những web server tốt nhất hiện nay, Nginx sẽ là 1 trong những cái tên xuất hiện đầu tiên cho tìm kiếm của bạn. Hôm nay nhóm chúng tôi sẽ có 1 bài thuyết trình về Nginx để giúp mọi người có cái nhìn tổng quát hơn về Nginx.

II. NGINX LÀ GÌ ?

- Nginx là một phần mềm web server mã nguồn mở, miễn phí được phát triển bởi lập trình viên người Nga - Igor Sysoev và được phát hành công khai vào năm 2004.
- Ban đầu Nginx chỉ được sử dụng như 1 web server, nhưng ngày nay Nginx còn có thể hoạt động như 1 proxy server cho HTTP, HTTPS, SMTP, IMAP, POP3; load balancer (cân bằng tải) cho các server giúp ổn định và cân bằng hệ thống; HTTP cache,...
- Nginx được thiết kế hướng đến mục đích cải thiện tối đa hiệu năng và sự ổn định, nên trong các trường hợp cần phục vụ nội dung tĩnh (file hình ảnh, html, css, js, text,..) hoặc các yêu cầu truy vấn đồng thời số lượng lớn, Nginx sẽ làm tốt hơn hầu hết các đối thủ còn lại.
- Đặc biệt, Nginx còn là 1 trong số ít những máy chủ có thể giải quyết được vấn đề C10K (kiểm soát 10,000 lượt truy vấn đồng thời).
- Chính vì những ưu điểm đó, hiện nay có rất nhiều các công ty, tập đoàn lớn sử dụng Nginx cho công ty của họ với nhiều mục đích khác nhau có thể kể đến như: Autodesk, Atlassian (Jira, Bitbucket, Trello,..), GitLab, Microsoft, IBM, Google, Adobe, LinkedIn, Facebook,...

III. CẤU TRÚC VÀ NGUYÊN LÝ HOẠT ĐỘNG

A. Cấu trúc

- Thông thường, khi người dùng truy cập trang Web, browser sẽ kết nối với Server chứa trang Web này. Sau đó, Server sẽ tìm đúng file yêu cầu của Website và gửi lại cho người dùng. Đây được gọi là trình tự xử lý Single Thread hay cấu trúc đơn luồng. Nghĩa là các bước chỉ thực hiện đúng theo một quy trình duy nhất. Và mỗi yêu cầu sẽ được tạo cho một luồng riêng biệt. Nhưng nếu có quá nhiều lượt truy cập cùng 1 lúc, mỗi truy cập

tạo 1 luồng xử lý riêng, điều này chắc chắn sẽ dẫn đến việc hệ thống bị quá tải khi phải sử dụng quá nhiều tài nguyên.

- Tuy nhiên, nguyên lý hoạt động của Nginx có sự khác biệt. Nó không tuân theo nguyên tắc này, mà nó hoạt động theo cơ chế hướng sự kiện (event-driven), dạng cấu trúc bất đồng bộ (asynchronous) và non-blocking I/O.
    - **event-driven:** nghĩa là nó sẽ phản hồi lại những hành động được tạo ra bởi users hoặc hệ thống.
    - **asynchronous:** là hệ thống sẽ không phụ thuộc vào thời gian đến của tín hiệu (messages) để hoạt động.
    - **non-blocking:** nghĩa là các hàm sẽ không bị dừng lại để chờ 1 hành động gì đó xảy ra.
- Kiến trúc của Nginx được chia thành 3 phần: Master process, Worker process và Cache (Cache loader, Cache manager).

1. **Master Process**
    - Không trực tiếp xử lý yêu cầu từ client, mà thực hiện việc quản lý cấu hình để đảm bảo các sự chính xác trước khi vận hành, quản lý socket và kiểm soát các tiến trình của worker process. Nó chỉ định công việc cho worker process mà không cần chờ phản hồi và gửi phản hồi của worker process trở lại cho máy khách. Xử lý các bản cập nhật mới.

2. **Worker Process**
    - Thực hiện theo yêu cầu của Master Process. Lắng nghe kết nối từ client, xử lý yêu cầu. Tiết kiệm bộ nhớ bằng cách chia sẻ cùng một không gian. Sau khi xử lý, các worker gửi phản hồi đến Master.
    - bên trong 1 worker process:
        - các yêu cầu được gửi đến chưa được chấp nhận bởi worker process trước tiên sẽ đợi trong listen socket.
        - sau khi worker process chấp nhận yêu cầu gửi đến trong listen socket, nó không chỉ ngồi và chờ đợi. nếu connection này đang đợi những thứ như (I/O): đọc file, fetching data từ database, hệ thống sẽ không bị block (cơ chế non-blocking). Nó sẽ để connection này qua 1 bên, và xử lý tới connection tiếp theo.
        - connection socket: nơi mà các request đã tồn tại chờ sau khi đã hoàn thành I/O để tiếp tục.

3. Cache

- Nginx tận dụng cache để tăng tốc độ phân phối trang bằng cách phục vụ nội dung trực tiếp từ cache thay vì phải lấy nội dung từ máy chủ mỗi lần. Khi một trang được yêu cầu lần đầu tiên, trang đó sẽ được lưu trữ trong cache để truy cập nhanh hơn trong các yêu cầu tiếp theo.
- cache Nginx bao gồm hai thành phần chính: Cache loader và Cache manager.
    - Cache loader: Thành phần này chịu trách nhiệm tải dữ liệu về nội dung được lưu trong bộ nhớ đệm vào vùng bộ nhớ dùng chung.
    - Cache Manager: Quy trình này chạy theo các khoảng thời gian đều đặn để theo dõi trạng thái của bộ đệm. Nếu kích thước bộ đệm vượt quá giới hạn được xác định bởi tham số max_size, Cache Manager sẽ xóa dữ liệu ít được truy cập gần đây nhất để duy trì kích thước và hiệu suất bộ đệm tối ưu.

4. Ví dụ 1 luồng xử lý request

- Bước 1: Client gửi yêu cầu truy cập một website đến địa chỉ IP/tên miền của server Nginx, server này đang lắng nghe trên các port tiêu chuẩn như 80 (HTTP) hoặc 443 (HTTPS). Yêu cầu khi được gửi đến sẽ nằm trong “listen socket” của worker process đang rảnh rỗi.
- Bước 2: Khi worker process này chấp nhận kết nối, nó chưa thực hiện ngay việc xử lý các yêu cầu.
- Bước 3: Xử lý Non-Blocking và Vòng lặp sự kiện
    - Đây là điểm cốt lõi của Nginx! Thay vì tạo riêng 1 luồng cho request của user này (như Apache truyền thống), Worker Process sử dụng một vòng lặp sự kiện (Event Loop) duy nhất.
    - Khi Worker Process cần chờ phản hồi (đọc file từ ổ đĩa, chờ dữ liệu từ máy chủ backend), nó không bị "chặn" (đứng chờ) mà tạm dừng xử lý yêu cầu hiện tại, đây là cơ chế non-blocking.
    - Nhờ Event Loop Nginx đăng ký một "sự kiện" với hệ điều hành (kernel) và tiếp tục xử lý các yêu cầu khác.

- Bước 4: Kernel thông báo sự kiện
    - Khi dữ liệu sẵn sàng (file đã được đọc xong, phản hồi từ backend đã về), hệ điều hành sẽ thông báo cho Worker Process (đứng đợi trong connection socket)..
    - Worker Process nhận được thông báo này và quay lại xử lý yêu cầu tương ứng.
- Bước 5: Nginx xử lý yêu cầu và gửi phản hồi
    - Nếu là nội dung tĩnh: Worker Process sẽ đọc file tĩnh (HTML, CSS, JS, ảnh) từ ổ đĩa và gửi trực tiếp về cho client.
    - Nếu là nội dung động (Reverse Proxy): Worker Process sẽ chuyển tiếp yêu cầu đó đến một máy chủ backend như Gunicorn/uWSGI (cho Flask), Node.js, PHP-FPM.
    - Sau khi nhận được phản hồi từ backend, Nginx sẽ gửi phản hồi đó về cho client.

B. File Cấu hình

- **/etc/nginx/:** Đây là thư mục cấu hình chính cho máy chủ Nginx. Thư mục này đóng vai trò là vị trí gốc nơi lưu trữ tất cả các tệp cấu hình, hướng dẫn Nginx cách vận hành và quản lý lưu lượng truy cập web 1 cách hiệu quả.
- **/etc/nginx/nginx.conf:** Đây là file cấu hình chính cho Nginx. File này chứa các thiết lập chung ảnh hưởng đến toàn bộ máy chủ, chẳng hạn như số lượng worker process, điều chỉnh các tham số để tối ưu hóa hiệu suất, cấu hình ghi nhật ký và chỉ thị để tải các module động. Ngoài ra, tệp này bao gồm các tham chiếu đến các tệp cấu hình khác, cho phép quản lý theo module và có tổ chức các thiết lập máy chủ.
- **/etc/nginx/sites-available**: dùng để lưu trữ cấu hình khả dụng. Chứa toàn bộ các file cấu hình của các website/server block mà bạn có thể chạy bằng Nginx. Mỗi file đại diện cho một website hoặc một cấu hình máy chủ riêng biệt (ví dụ: example.com, myapp.local,...). Các file trong đây không tự động được kích hoạt mà chỉ đơn giản là “available”.
- **/etc/nginx/sites-enabled**: dùng để lưu trữ cấu hình đang được kích hoạt. Chứa các symbolic link (liên kết tượng trưng) trỏ tới các file cấu hình thật trong sites-available. Nginx chỉ đọc và sử dụng các file trong thư mục này khi khởi động hoặc reload lại. Việc thêm một file vào sites-enabled là cách để kích hoạt cấu hình tương ứng.

- **/var/log/nginx/:** Thư mục này là vị trí mặc định cho các tệp nhật ký Nginx. Thư mục này chứa các nhật ký quan trọng như access.log, ghi lại tất cả các yêu cầu của máy khách đến máy chủ và error.log, ghi lại mọi lỗi hoặc sự cố mà Nginx gặp phải.
- **/var/www/html:** là thư mục mặc định chứa source code của ứng dụng, cũng có thể để ở bất kỳ thư mực nào khác.

1. **/etc/nginx/nginx.conf**
    - Đây là file cấu hình chính của Nginx. File này được viết theo cấu trúc ngữ cảnh (context). Có 2 context chính: Event và HTTP. Mỗi context có thể có những context khác được lồng vào trong, nó sẽ kế thừa mọi thứ từ context cha, nhưng có thể được thiết lập lại nếu cần thiết.
    - **event context:**
        - Nó được sử dụng để thiết lập các tùy chọn toàn cục ảnh hưởng đến cách Nginx xử lý các kết nối ở cấp độ chung. Chỉ có thể có duy nhất 1 event context được xác định trong file cấu hình Nginx.
        - Nginx sử dụng mô hình xử lý kết nối dựa trên sự kiện (event-driven), do đó các chỉ thị được xác định trong này sẽ xác định cách các tiến trình công việc xử lý kết nối.
        - Về cơ bản, các chỉ thị tìm thấy ở đây được sử dụng để chọn kỹ thuật xử lý kết nối cần sử dụng hoặc để sửa đổi cách triển khai các phương pháp này.
    - **http context:**
        - Khi cấu hình Nginx làm máy chủ web hoặc reverse proxy, http context sẽ chứa phần lớn trong file cấu hình. Context này sẽ chứa tất cả các chỉ thị và các context khác khác cần thiết để xác định cách chương trình sẽ xử lý kết nối HTTP hoặc HTTPS.
        - http context là anh chị em của event context, vì vậy chúng nên được liệt kê cạnh nhau, thay vì lồng nhau. Cả hai đều là con của main context.

2. **/etc/nginx/sites-available**
    - **server context:**
        - server context là con của http context.
        - Trong 1 http context, có thể có nhiều server context, vì mỗi server context là mỗi phiên bản xác định một máy chủ ảo cụ thể để xử lý các yêu cầu của máy khách. Có thể có

nhiều khối máy chủ tùy theo nhu cầu, mỗi khối có thể xử lý một tập hợp kết nối cụ thể.

- **location context:**
    - Mỗi location được sử dụng để xử lý một loại yêu cầu nhất định từ client và mỗi vị trí được chọn bằng cách so khớp định nghĩa vị trí với yêu cầu của client thông qua thuật toán lựa chọn.
    - hình
    - Các location context nằm trong các server context và được sử dụng để quyết định cách xử lý URI - Uniform Resource Identifier yêu cầu (phần yêu cầu theo sau tên miền hoặc địa chỉ IP/cổng).

- Bên cạnh các context trên, còn có các context tùy chọn khác như: upstream context, split_clients context, map context, geo context,...
- **worker_processes:** định nghĩa số lượng tiến trình worker process (mặc định là 1). Trong hầu hết các trường hợp, chạy 1 process làm việc trên mỗi lõi CPU hoạt động tốt. Tuy nhiên, cài đặt được khuyến nghị là tự động.
- **worker_connections:** Worker connection là số lượng kết nối tối đa mà mỗi worker_processes có thể xử lý đồng thời. Theo mặc định là 512, nhưng nhiều hệ thống có đủ tài nguyên để hỗ trợ số lượng lớn (tức là 1024, 2048 cho mỗi quy trình làm việc hoặc thậm chí nhiều hơn)
- **access.log & error.log**: Được sử dụng để ghi lại các lỗi xảy ra trong Nginx và cũng được sử dụng cho mục đích gỡ lỗi.
- **gzip:** Đây là cài đặt cho nén gzip trên phản hồi Nginx. Gzip trên Nginx là một module giúp nén dữ liệu trước khi truyền tải qua mạng, giúp giảm kích thước dữ liệu và tăng tốc độ tải trang web.

C. Các lệnh cơ bản

Command	Description
nginx -h	Shows the NGINX help menu.
nginx -v	Shows the NGINX version.
nginx -V	Shows the NGINX version, build information, and configuration arguments
nginx -t	Tests the NGINX configuration.
nginx -T	Tests the NGINX configuration and prints the validated configuration to the screen
nginx -s signal	The -s flag sends a signal to the NGINX master process. You can send signals such as stop , quit, reload and reopen
systemctl start nginx	Start the nginx process , can also use systemctl restart nginx
systemctl stop nginx	Stop nginx Process
systemctl status nginx	Show the nginx status either running or failed

IV. CÁC CHỨC NĂNG CỦA NGINX

A. Web Server

- Đây là chức năng cơ bản nhất của Nginx, hoạt động như 1 web server, vừa là máy chủ vật lý, vừa chứa phần mềm, database chạy trên đó. Có nhiệm vụ chính là lắng nghe và xử lý các yêu cầu từ browser của client và phản hồi lại cho browser.

B. Proxy Server

1. Load Balancer

- Giả sử rằng trang web của bạn đang hoạt động rất ổn với 1 server. Cho đến 1 ngày, trang web của bạn bất ngờ được biết tới rộng rãi, lúc này sẽ có đến hàng trăm nghìn, thậm chí là vài triệu lượt yêu cầu truy cập vào trang web của bạn. Chỉ với 1 server thì không thể kiểm soát được lượng lớn truy cập như vậy, sẽ dẫn tới xảy ra các vấn đề kĩ thuật trong server, và server của bạn chắc chắn sẽ sập.
- Lúc này bạn sẽ phải có thêm nhiều server khác để giải quyết vấn đề này. Nhưng lúc này làm sao bạn xác định được request từ client sẽ đi vào server nào, server nào trong số các server của bạn sẽ xử lý request này, đó là lúc Nginx hoạt động như 1 Load Balancer (cân bằng tải) cho hệ thống của bạn.
- 1 server Nginx như vậy sẽ đóng vai trò là Load Balancer, nhận request từ client và phân phối request này qua các server khác, để đảm bảo cân bằng hệ thống, cải thiện hiệu suất và tăng tính ổn định. Việc lựa chọn sẽ phân phối đến server nào sẽ phụ thuộc vào thuật toán được cài đặt trên đó.
- Các thuật toán điều phối request:
    - **Round-Robin:** đây là thuật toán mặc định của Nginx. Sử dụng thuật toán round-robin để quyết định sẽ chuyển request đến server nào (request sẽ được gửi từng cái 1 tới từng server)
    - **Least-Connected:** thuật toán này sẽ chuyển request đến server nào hiện đang có ít connection nhất.
    - **Nhược điểm của 2 thuật toán** trên là kết nối tiếp theo từ cùng 1 client sẽ không đi đến cùng 1 server đó. Nếu web của bạn không phụ thuộc vào session, thì điều này hoàn toàn ổn. Nhưng nếu có, thì khi kết nối đầu tiên của client được thiết lập trên server A,

điều bạn muốn là tất cả các kết nối tiếp theo trong tương lai của client này đều đi đến server A. Nhưng 2 thuật toán trên sẽ không đảm bảo việc này.

- **IP-Hash:** thuật toán này sẽ dùng địa chỉ IP của client để quyết định request của client này sẽ được gửi đến server nào. Phương pháp này đảm bảo rằng các yêu cầu từ cùng 1 client sẽ luôn được chuyển đến cùng 1 server, trừ khi server này đang bị down.

- Lợi ích:
  - **Cân bằng tải hệ thống:** Tránh server quá tải, tối ưu hiệu suất.
  - **Tăng khả năng chịu lỗi:** Nếu một server chết, Nginx tự chuyển hướng request đến server còn sống.
  - **Mở rộng dễ dàng (scalability):** Thêm server mới mà không ảnh hưởng client.

2. Caching

- Hãy lấy ví dụ là trang báo VnExpress - trang báo rất thịnh hành ở Việt Nam. Mỗi ngày có hàng triệu lượt truy cập lên trang để đọc báo. Hãy tưởng tượng với mỗi 1 request cho 1 bài báo nào đó, request được gửi từ browser đến proxy server, proxy server chuyển request vào server chính, server chính truy cập vào database để lấy dữ liệu, hình ảnh, văn bản, gộp lại với nhau và gửi phản hồi về cho browser của client để hiện thị lên trang web. Việc này là vô cùng mất thời gian và sẽ làm cho thời gian chờ phản hồi về cho client dài hơn, giảm hiệu suất của trang web.
- Cách giải quyết là chỉ làm điều này 1 lần (truy cập database lấy dữ liệu, hình ảnh, text, link gộp lại) lưu trữ ở proxy server. Khi có request từ client, thì ngay lập tức chỉ cần gửi file này lại cho client là xong.
- Lợi ích:
  - **Tăng tốc độ phản hồi:** Nội dung được phục vụ trực tiếp từ cache (không cần truy backend).
  - **Giảm tải backend:** Giảm số lượng request backend phải xử lý, giảm CPU/RAM sử dụng ở backend.
  - **Tối ưu SEO & UX:** Trang web tải nhanh hơn → trải nghiệm người dùng tốt hơn.

3. **Security (One entrypoint)**
    - Giả sử trang web của bạn hiện có 10 cái servers và tất cả 10 servers này đều có thể truy cập trực tiếp (không qua proxy server). Điều này làm cho công việc của các hackers trở nên dễ dàng hơn. Vì các servers này có thể truy cập trực tiếp, các hackers có thể tìm thấy lỗ hổng trên 1 trong những server đó, từ đó hackers có thể sẽ đột nhập được vào toàn bộ hệ thống của bạn. Lúc này bạn sẽ phải lo việc bảo mật cho toàn bộ 10 cái servers này. Khi có những lỗ hổng mới được phát hiện trên 1 server nào đó, thì bạn lại phải đi fix lỗ hổng đó trên 9 cái server còn lại. 1 công việc được lặp đi lặp lại quá nhiều lần là không cần thiết.
    - Nên thay vì phải lo bảo mật cho toàn bộ 10 cái servers đó, bạn chỉ cần 1 server đặt trước những cái server này, chỉ duy nhất 1 cái server này là có thể truy cập trực tiếp, những cái còn lại thì không thể. Lúc này bạn chỉ cần lo việc bảo mật cho 1 server này là đủ. Vi không còn con đường nào khác, mà không đi qua server này để vào các server phía sau. Nó là entrypoint duy nhất để truy cập vào hệ thống của bạn, như 1 tấm khiên bảo vệ toàn bộ hệ thống.
    - Lợi ích:
        - **Bảo mật cao hơn:**
            - Ẩn toàn bộ hạ tầng backend.
            - Chặn IP xấu, bot, DDoS nhỏ.
            - Kiểm tra SSL, JWT, xác thực HTTP.
        - **Quản lý dễ dàng:**
            - Tập trung logging, giám sát, firewall.
            - Cấu hình redirect, rewrite, HTTPS ở một nơi duy nhất.
        - **Tăng tính linh hoạt:**
            - Dễ thay đổi hệ thống backend mà không ảnh hưởng client.

V. TÀI LIỆU THAM KHẢO

1. https://viblo.asia/p/tim-hieu-tong-quan-ve-nginx-63vKjOExZ2R
2. https://www.slideshare.net/DhrubajiMandal/nginx-176332309
3. https://www.youtube.com/watch?v=iInUBOVeBCc&t=10s
4. https://www.youtube.com/watch?v=I6dpN0geIb4