
Viet Anh @Nguyen.Viet.AnhB Theo dõi
★ 3.8K 146 52

Đã đăng vào thg 7 26, 2016 4:10 CH - 53 phút đọc

Tìm hiểu và hướng dẫn setup web server Nginx

...

I, Web server là gì ?

1, Khái niệm.

- Máy chủ Web (Web Server) là máy tính mà trên đó cài đặt phần mềm phục vụ web, đôi khi người ta cũng gọi chính phần mềm đó là web server. Tất cả các web server đều hiểu và chạy được các file *.htm và *.html. Tuy nhiên mỗi web server lại phục vụ một số kiểu file chuyên biệt chẳng hạn như llS của Microsoft dành cho *.asp, *.aspx...; Apache, Nginx dành cho *.php...; Sun Java system web server của SUN dành cho *.jsp...

- Ở phần lõi của máy chủ web là một dịch vụ web phục vụ nội dung tĩnh cho một trình duyệt bằng cách tải một tập tin từ đĩa và chuyển nó lên mạng, tới một người sử dụng trình duyệt web. Sự trao đổi hoàn toàn này được thực hiện gián tiếp thông qua một trình duyệt và một máy chủ kết nối tới một thiết bị khác sử dụng HTTP. Bất kỳ máy tính nào cũng có thể vào trong một dịch vụ web bằng cách cài đặt phần mềm dịch vụ và kết nối internet.

2, Tổng quan về Web server

- Máy Web Server là máy chủ có dung lượng lớn, tốc độ cao, được dùng để lưu trữ thông tin như một ngân hàng dữ liệu, chứa những website đã được thiết kế cùng với những thông tin liên quan khác. (các mã Script, các chương trình, và các file Multimedia)

- Web Server có khả năng gửi đến máy khách những trang Web thông qua môi trường Internet (hoặc Intranet) qua giao thức HTTP – giao thức được thiết kế để gửi các file đến trình duyệt Web (Web Browser), và các giao thức khác.

- Tất cả các Web Server đều có một địa chỉ IP (IP Address) hoặc cũng có thể có một Domain Name. Giả sử khi bạn đánh vào thanh Address trên trình duyệt của bạn một dòng http://www.abc.com sau đó gõ phím Enter bạn sẽ gửi một yêu cầu đến một Server có Domain Name là www.abc.com. Server này sẽ tìm trang Web có tên là index.htm rồi gửi nó đến trình duyệt của bạn.

- Bất kỳ một máy tính nào cũng có thể trở thành một Web Server bởi việc cài đặt lên nó một chương trình phần mềm Server Software và sau đó kết nối vào Internet.

- Khi máy tính của bạn kết nối đến một Web Server và gửi đến yêu cầu truy cập các thông tin từ một trang Web nào đó, Web Server Software sẽ nhận yêu cầu và gửi lại cho bạn những thông tin mà bạn mong muốn.

- Giống như những phần mềm khác mà bạn đã từng cài đặt trên máy tính của mình, Web Server Software cũng chỉ là một ứng dụng phần mềm. Nó được cài đặt, và chạy trên máy tính dùng làm Web Server, nhờ có chương trình này mà người sử dụng có thể truy cập đến các thông tin của trang Web từ một máy tính khác ở trên mạng (Internet, Intranet).

- Web Server Software còn có thể được tích hợp với CSDL (Database), hay điều khiển việc kết nối vào CSDL để có thể truy cập và kết xuất thông tin từ CSDL lên các trang Web và truyền tải chúng đến người dùng.

- Server phải hoạt động liên tục 24/24 giờ, 7 ngày một tuần và 365 ngày một năm, để phục vụ cho việc cung cấp thông tin trực tuyến. Vị trí đặt server đóng vai trò quan trọng trong chất lượng và tốc độ lưu chuyển thông tin từ server và máy tính truy cập.

II, Tìm hiểu về Nginx

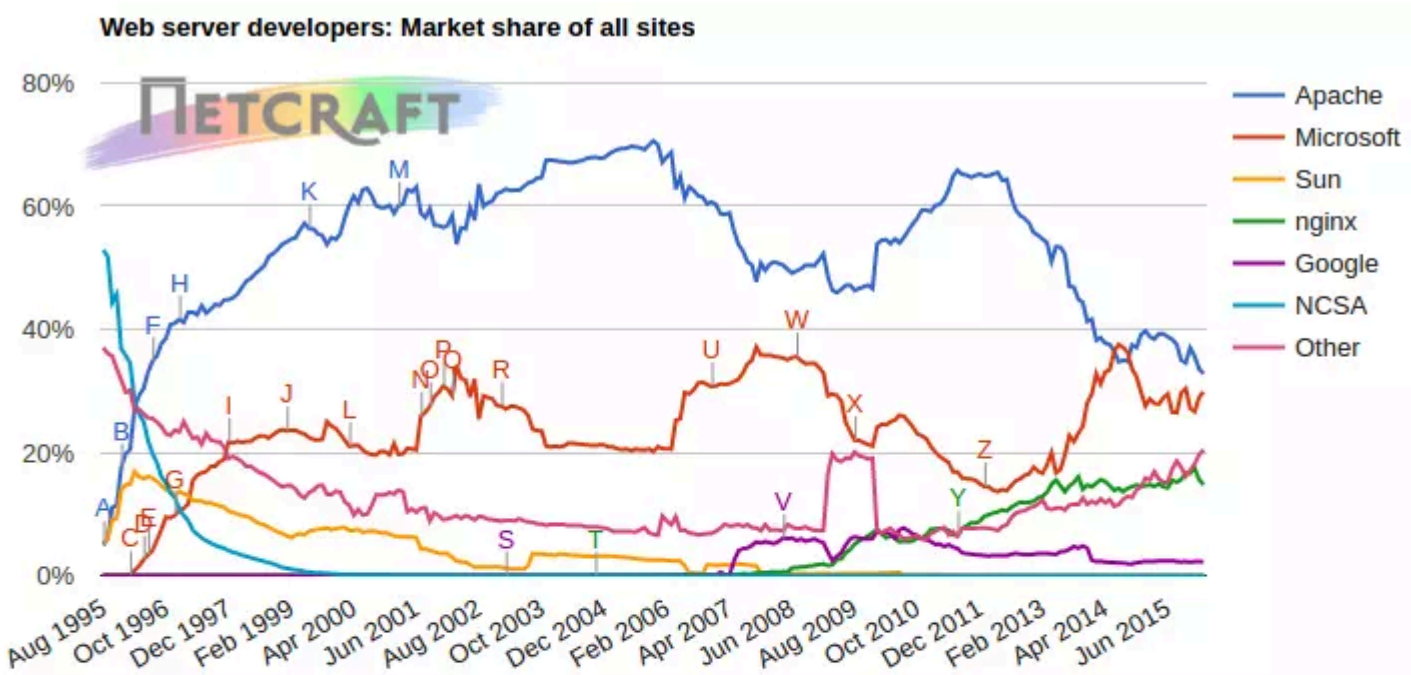
1, Tổng quan

- Nginx là 1 máy chủ reverse proxy mã nguồn mở cho các giao thức HTTP, HTTPS, SMTP, POP3 và IMAP, cũng như là 1 máy chủ cân bằng tải (load balancer), HTTP cache và web. Dự án Nginx được bắt đầu với việc tập trung vào tính đồng thời cao, hiệu năng cao và sử dụng tài nguyên thấp và được phát triển bởi Igor Sysoev vào nằm 2002, được phân phối ra công chúng lần đầu vào nằm 2004.

- Không giống với các máy chủ web truyền thống, Nginx không dựa trên luồn (thread) để xử lý yêu cầu. Thay vào đó, nó sử dụng 1 kiến trúc bất đồng bộ hướng sự kiện linh hoạt . Kiến trúc này sử dụng ít, nhưng quan trọng hợn, là lượng bộ nhớ có thể dự đoán khi hoạt động. Đây chính là điểm mấu chốt khiến Nginx là 1 trong số ít những máy chủ được viết để giải quyết vấn đề C10K

- Nói đến đây một số bạn sẽ tò mò là vấn đề `C10K` là gì ? Hiểu đơn giản thì do các máy chủ web truyền thống xử lý các yêu cầu dựa trên luồn (thread), tức là mỗi khi máy chủ web nhận được 1 yêu cầu mới, nó sẽ tạo ra 1 luồng mới để xử lý cho yêu cầu này, và cứ thế khi số lượng các yêu cầu gửi đến máy chủ web ngày càng nhiều thì số lượng các luồn xử lý này trong máy chủ sẽ ngày càng tăng. Và điều này dẫn đến việc thiếu hụt tài nguyên cấp cho các luồn xử lý trên ... (Các bạn có thể tìm hiểu rõ hơn tại đây)

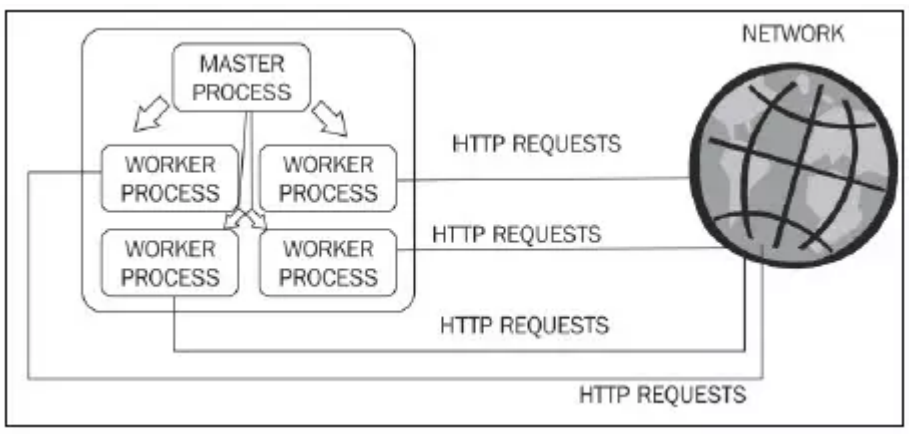
- Hiện nay, có khoảng 14,72 % (hơn 137 triệu) các website trên Internet đang sử dụng Nginx là máy chủ web.



Developer	January 2016	Percent	February 2016	Percent	Change
Apache	304,271,061	33.56%	306,292,557	32.80%	-0.76
Microsoft	262,471,886	28.95%	278,593,041	29.83%	0.88
nginx	141,443,630	15.60%	137,459,391	14.72%	-0.88
Google	20,799,087	2.29%	20,640,058	2.21%	-0.08

2, Kiến trúc của Nginx

- Khi được khởi chạy service, nginx khởi tạo mọt tiến trình chủ và cũng là tiến trình duy nhất tồn tại trong bộ nhớ `Master Process` . Tiến trình này không chịu trách nhiệm tự xử lý bất kỳ request nào từ phía client mà thay vào đó nó sinh ra các tiến trình con gọi là `Worker Process` để xử lý các request này.



- Để định nghĩa cho các `Worker Process` này, chúng ta cần sử dụng tệp tin cấu hình để xác định số tiến trình, số lượng kết nối , tài khoản và nhóm tài khoản mà mỗi `Worker Process` chạy

III, Cấu hình Http trong Nginx

1, Giới thiệu module Http trong Nginx

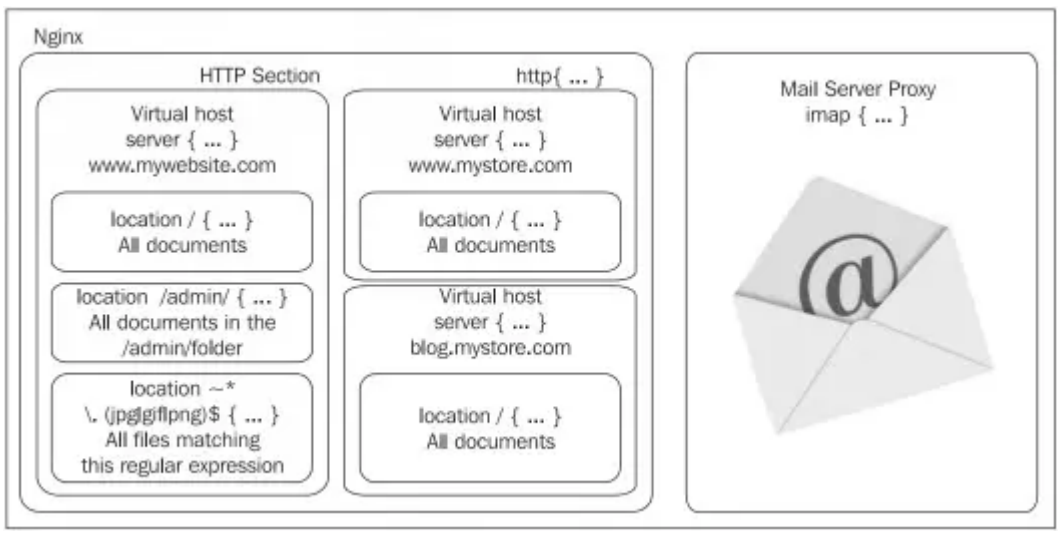
- Module HTTP Core là thành phần chứa tất cả các khối, chỉ thị và các biến cơ bản của máy chủ HTTP. Mặc định module này được cài đặt trong khi biên dịch, nhưng không được bật lên khi Nginx chạy, việc sử dụng module này là không bắt buộc

- Module này là 1 trong những module tiêu chuẩn lớn nhất của Nginx – nó cung cấp 1 số lượng lớn các chỉ thị và biến. Để có thể hiểu được tất cả các yếu tố này và vai trò của chúng, chúng ta sẽ bắt tay vào tìm hiểu 3 khối chỉ thị chính – `http` , `server` và `location` .

- `http` : được khai báo ở phần đầu của tập tin cấu hình. Nó cho phép chúng ta định nghĩa các chỉ thị và các khối từ tất cả các module liên quan đến khía cạnh HTTP của Nginx. Khối chỉ thị này có thể được khai báo nhiều lần trong tập tin cấu hình, và các giá trị chỉ thị được chèn trong khối http sau sẽ ghi đè lên các chỉ thị nằm trong khối http trước đó

- `server` : khối này cho phép chúng ta khai báo 1 website. Nói cách khác, 1 website cụ thể (được nhận diện bởi 1 hoặc nhiều hostname) được thừa nhận bởi Nginx và nhận cấu hình của chính nó. Khối này chỉ có thể được dùng bên trong khối http.

- `location` : cho phép chúng ta định nghĩa 1 nhóm các thiết lập được áp dụng cho 1 vị trí cụ thể trên website (thể hiện qua URL của website đó). Khối location có thể được dùng bên trong 1 khối server hoặc nằm chồng bên trong 1 khối location khác.



Trong biểu đồ trên, khu vực HTTP, được định nghĩa bởi khối http, bao quanh toàn bộ các cấu hình liên quan đến web. Nó cũng chứa 1 hoặc nhiều khối server, định nghĩa các tên miền của các website mà chúng ta có. Với mỗi website này, chúng ta có thể định nghĩa nhiều khối location mà cho phép chúng ta áp dụng các thiết lập bổ sung đến 1 URI yêu cầu cụ thể của website hoặc các URI yêu cầu khớp 1 mẫu nào đó.

2, Các chỉ thị

2.1 : Các chỉ thị về cấu hình HOST và SOCKET

- `listen` :
  - Sử dụng trong khối : `server`
  - Chỉ rõ địa chỉ IP và/hoặc port được dùng bởi socket phục vụ website. Các website thường được phục vụ trên port 80 (giá trị mặc định) qua HTTP, hoặc 443 qua HTTPS.
  - Cú pháp: `listen [address] [:port] [additional options];`
  - Các tùy chọn bổ sung:
    - `default hoặc default_server` : Chỉ rõ khối server này được dùng như website mặc định cho bất kỳ yêu cầu nhận được tại địa chỉ IP và port được chỉ rõ.
    - `ssl` : Chỉ rõ website sẽ sử dụng SSL.
    - Các tùy chọn khác liên quan đến các lời gọi hệ thống bind và listen gồm: `backlog=num` , `rcvbuf=size` , `sndbuf=size` , `accept_filter=filter` , `deferred` , `setfib=number` , và `bind` .
  - Ví dụ :

```
listen 192.168.1.1:80;
listen 127.0.0.1;
listen 80 default;
listen 443 ssl;
```

- `server_name` :
  - Sử dụng trong khối : `server`
  - Đăng ký 1 hoặc nhiều hostname cho khối server. Khi Nginx nhận 1 yêu cầu HTTP, nó so sánh giá trị Host trong phần header của yêu cầu với tất cả các khối server đang có. Khối server đầu tiên khớp với hostname này sẽ được chọn.
  - Nếu không có khối `server` nào khớp với `hostname` trên, Nginx chọn khối server đầu tiên khớp với các thông số của chỉ thị listen (ví dụ như listen *:80 sẽ bắt tất cả các yêu cầu nhận được trên port 80), ưu tiên khối đầu tiên có tùy chọn mặc định được cho phép trên chỉ thị listen.
  - Cú pháp: `server_name hostname1 [hostname2…];`
  - Ví dụ :

```
server_name www.acb.com;
server_name www.abc.com abc.com;
server_name *.website.com; # nhận tất cả các domain có đuôi là .website.com
server_name .website.com; # Kết hợp cả *.website.com và website.com
server_name *.website.*;
server_name ~^\.example\.com$;
```

Lưu ý rằng chúng ta có thể sử dụng chuỗi rỗng như 1 giá trị của chỉ thị để bắt tất cả các yêu cầu không có giá trị Host trong phần header, nhưng chỉ sau ít nhất 1 tên thông thường (hoặc “_”)

```
server_name abc.com “”;
server_name _ “”;
```

- `server_name_in_redirect` :
  - Sử dụng trong khối : `server` , `http` , `location`
  - Chỉ thị này áp dụng cho trường hợp của các chuyển hướng nội bộ (internal redirect). Nếu thiết lập thành on, Nginx sẽ sử dụng hostname đầu tiên được chỉ rõ trong chỉ thị server_name. Nếu thiết lập thành off, Nginx sẽ sử dụng giá trị Host trong phần header của yêu cầu HTTP.
  - Cú pháp: `server_name_in_redirect [on | off];`
  - Giá trị mặc định: off

- `server_names_hash_max_size` :
  - Sử dụng trong khối : `http`
  - Nginx sử dụng các bảng `hash` cho các tập hợp dữ liệu khác nhau để tăng tốc việc xử lý yêu cầu. Chỉ thị này định nghĩa kích thước tối đa của bảng `hash` chứa các server_name. Giá trị mặc định phù hợp với đa số cấu hình. Nếu giá trị này cần thay đổi, Nginx sẽ tự động thông báo cho chúng ta khi khởi động, hay khi chúng ta tải lại cấu hình.
  - Cú pháp: `server_names_hash_max_size 512;`
  - Giá trị mặc định: 512 bytes

- `server_names_hash_bucket_size` :
  - Sử dụng trong khối : `http`
  - Thiết lập kích thước vùng chứa các bảng hash server_name. Tương tự, chúng ta chỉ nên thay đổi giá trị này khi Nginx yêu cầu.
  - Cú pháp: `server_names_hash_bucket_size 32;`
  - Giá trị mặc định: 32 byte (hoặc 64, hay 128, dựa vào bộ nhớ cache của CPU).

- `port_in_redirect` :
  - Sử dụng trong khối : `server` , `http` , `location`
  - Chỉ thị này định nghĩa việc Nginx có hoặc không thêm giá trị port vào URL chuyển hướng.
  - Cú pháp: `port_in_redirect [on or off];`
  - Giá trị mặc định: on

- `tcp_nodelay` :
  - Sử dụng trong khối : `server` , `http` , `location`
  - Cho phép hoặc vô hiệu hóa tùy chọn socket TCP_NODELAY cho chỉ các kết nối keep-alive.
  - Cú pháp: `tcp_nodelay [on or off];`
  - Giá trị mặc định: on
  - NOTE : Lưu ý: Keep-alive (hoặc Keepalive) là 1 thông điệp được gửi bởi 1 thiết bị đến các thiết bị khác để kiểm tra rằng liên kết giữa 2 thiết bị đang hoạt động, hoặc để ngăn ngừa việc liên kết này bị mất.

- `tcp_nopush` :
  - Sử dụng trong khối : `server` , `http` , `location`
  - Cho phép hoặc vô hiệu hóa tùy chọn socket TCP_NOPUSH (FreeBSD) hoặc TCP_CORK (Linux). Chỉ thị này chỉ áp dụng khi chỉ thị sendfile được cho phép. Nếu chỉ thị tcp_nopush được thiết lập là on, Nginx sẽ cố gắng truyền toàn bộ phần header phản hồi HTTP trong 1 gói tin TCP.
  - Cú pháp: `tcp_nopush [on or off];`
  - Giá trị mặc định: off

- `sendfile_max_chunk` :
  - Sử dụng trong khối : `server` , `http`
  - Chỉ thị này định nghĩa 1 kích thước tối đa của dữ liệu được dùng cho mỗi lời gọi sendfile.
  - Cú pháp: `sendfile_max_chunk 0;`
  - Giá trị mặc định: 0

- `reset_timedout_connection` :
  - Sử dụng trong khối : `server` , `http` , `location`
  - Khi 1 kết nối hết thời hạn, thông tin tương ứng của nó có thể được giữ trong bộ nhớ dựa vào trạng thái của kết nối đó. Việc sử dụng chỉ thị này sẽ xóa tất cả bộ nhớ tương ứng đến kết nối sau khi nó hết thời hạn.

- Cú pháp: `reset_timedout_connection off;`
- Giá trị mặc định: off

2.2 : Cấu hình đường dẫn và tài liệu

- `root` :
  - Sử dụng trong khối : `server` , `http` , `location` , `if` Các biến được chấp nhận.
  - Định nghĩa tài liệu gốc, chứa các tập tin mà bạn muốn phục vụ cho khách.
  - Cú pháp: `root /path/resource/;`
  - Giá trị mặc định: html

- `alias` :
  - Sử dụng trong khối : `location` Các biến được chấp nhận.
  - 1 chỉ thị mà chúng ta chỉ có thể đặt trong khối location. Nó đăng ký 1 đường dẫn khác cho Nginx lấy các tài liệu cho 1 yêu cầu cụ thể :

```
http {
server {
    server_name abc.com;
    root /var/www/abc.com/;
    location /admin/ {
        alias /var/www/abc.net/;
    }
}
```

Chú thích : Khi 1 yêu cầu cho http://abc.com/ được nhận, các tập tin được phục vụ từ thư mục /var/www/abc.com. Tuy nhiên, nếu Nginx nhận 1 yêu cầu cho http://abc.com/admin/, đường dẫn được dùng để lấy tập tin là /var/www/abc.net/. Hơn thế nữa, giá trị của chỉ thị root không được thay đổi. Quá trình này vô hình trong mắt của các script động

- `error_page` :
  - Sử dụng trong khối : `server` , `http` , `location` , `if` Các biến được chấp nhận.
  - Cho phép chúng ta ảnh hưởng các URI đến mã phản hồi HTTP và tùy chọn để thay thế code với cái khác.
  - Cú pháp: `error_page code1 [code2…] [=replacement code] [=@block | URI]`
  - Ví dụ :

```
error_page 404 /not_found.html;
error_page 500 501 502 503 504 /server_error.html;
error_page 403 http://website.com/;
error_page 404 @notfound; #nhảy đến 1 khối location được đặt tên là notfound
error_page 404 =200 /index.html; # trong trường hợp lỗi 404, chuyển hướng đến index.html với mã phản hồi là
```

- `if_modified_since` :
  - Sử dụng trong khối : `server` , `http` , `location`
  - Định nghĩa cách Nginx xử lý header If-Modified-Since trong gói tin HTTP. Header này được sử dụng đa số bởi các bot của công cụ tìm kiếm (như Google bot). Các bot này chỉ ra ngày tháng và thời gian của lần

truyền cuối cùng. Nếu tập tin được yêu cầu không được sửa đổi từ thời gian đó, máy chủ sẽ trả về 1 mã phản hồi 304 Not Modified.

- Chỉ thị này chấp nhận 3 giá trị:
  - `off` : bỏ qua header If-Modified-Since
  - `exact` : trả về mã 304 Not Modified nếu ngày tháng và thời gian được chỉ rõ trong phần header chính xác khớp với ngày tháng sửa đổi thật sự của tập tin được yêu cầu. Nếu ngày tháng sửa đổi tập tin là trước hoặc sau, tập tin được phục vụ bình thường. (mã phản hồi 200 OK).
  - `before` : trả về mã 304 Not Modified nếu ngày tháng và thời gian được chỉ rõ trong phần header là trước hoặc bằng với ngày tháng sửa đổi tập tin được yêu cầu.
- Cú pháp: `if_modified_since off | exact | before`
- Giá trị mặc định: exact

- `index` :
  - Sử dụng trong khối : `server` , `http` , `location` . Các biến được chấp nhận.
  - Định nghĩa trang mặc định mà Nginx sẽ phục vụ nếu không có tên tập tin được chỉ rõ trong yêu cầu (nói cách khác, trang chỉ mục). Chúng ta có thể chỉ rõ nhiều tên tập tin và tập tin đầu tiên được tìm thấy sẽ được sử dụng. Nếu không có tập tin cụ thể nào được tìm thấy, Nginx sẽ hoặc là cố gắng phát sinh 1 chỉ mục tự động của các tập tin, nếu chỉ mục autoindex được cho phép hoặc trả về 1 trang lỗi 403 Forbidden. Tùy chọn, chúng ta có thể nhập 1 tên tập tin tuyệt đối (như là /page.html, tính từ thư mục gốc của website) nhưng đây chỉ có thể là tham số cuối cùng của chỉ thị này..
  - Cú pháp: `ndex file1 [file2…] [absolute_file];`
  - Giá trị mặc định: index.php index.html index.htm;

- `recursive_error_pages` :
  - Sử dụng trong khối : `server` , `http` , `location` ,.
  - Khi 1 trang lỗi (được khai báo trong chỉ thị **error_page**) gặp lỗi, nó sử dụng giá trị của chỉ thị **error_page** (tức là chính trang lỗi đó) để xử lý lỗi trên, và điều này lặp đi lặp lại, tạo ra vòng lặp vô hạn với lỗi trên (tình trạng này được gọi là đệ quy). Chỉ thị này cho phép hoặc vô hiệu hóa các trang lỗi đệ quy như trên.
  - Cú pháp: `recursive_error_pages off/on;`
  - Giá trị mặc định: off

- `try_files` :
  - Sử dụng trong khối : `server` , `location` Các biến được chấp nhận.
  - Cố gắng phục vụ các tập tin được chỉ rõ (các tham số từ 1 đến N-1 trong chỉ thị), nếu không có tập tin nào tồn tại, nhảy đến khối location được khai báo (tham số cuối cùng trong chỉ thị) hoặc phục vụ 1 URI được chỉ định.
  - Cú pháp: Nhiều đường dẫn tập tin, theo sau bởi 1 khối location được đặt tên hoặc 1 URI.

```
        location / {
            try_files $uri $uri.html $uri.php $uri.xml @proxy;
        }
        # the following is a "named location block"
        location @proxy {
            proxy_pass 127.0.0.1:8080;
        }
```

- Trong ví dụ trên, Nginx sẽ cố gắng phục vụ các tập tin 1 cách bình thường. Nếu URI của yêu cầu không tương ứng với bất kỳ tập tin đang có nào, Nginx sẽ thêm .html vào URI và cố gắng phục vụ tập tin này 1 lần nữa. Nếu nó vẫn tiếp tục thất bại, Nginx thử với .php, sau đó là .xml. Nếu tất cả đều thất bại, 1 khối location khác (@proxy) xử lý yêu cầu này.

2.3 : Cấu hình các request từ client

- `keepalive_requests` :
    - Sử dụng trong khối : `server` , `http` , `location`
    - Xác định số lượng tối đa các yêu cầu được phục vụ trên 1 kết nối keep-alive.
    - Cú pháp: `keepalive_requests 100;`
    - Giá trị mặc định: 100
- `keepalive_timeout` :
    - Sử dụng trong khối : `server` , `http` , `location`
    - Định nghĩa số giây mà máy chủ sẽ chờ trước khi đóng 1 kết nối keep-alive.
    - Cú pháp: `keepalive_timeout time1 [time2];`
    - Giá trị mặc định: 75

```
        keepalive_timeout 75;
        keepalive_timeout 75 60;
```

- `keepalive_disable` :
    - Sử dụng trong khối : `server` , `http` , `location`
    - Tùy chọn này cho phép chúng ta vô hiệu hóa chức năng keepalive cho các trình duyệt web.
    - Cú pháp: `keepalive_disable browser1 browser2;`
    - Giá trị mặc định: msie6
- `send_timeout` :
    - Sử dụng trong khối : `server` , `http` , `location`
    - Số thời gian sau đó Nginx đóng 1 kết nối không hoạt động. 1 kết nối trở thành không hoạt động khi client ngừng truyền dữ liệu.
    - Cú pháp: `send_timeout [giá trị thời gian tính bằng giây];`
    - Giá trị mặc định: 60
- `client_body_in_file_only` :

- Sử dụng trong khối : `server` , `http` , `location`
- Nếu chỉ thị này được cho phép, phần nội dung của 1 yêu cầu HTTP sẽ được lưu trong các tập tin thực sự trên đĩa cứng. Phần client_body tương ứng với dữ liệu thô trong yêu cầu HTTP của client, trừ phần header (nói cách khác, nội dung được truyền trong các yêu cầu POST). Các tập tin được lưu như các tập tin text.
- Chỉ thị này chấp nhận 3 giá trị:
  - `off` : không lưu dữ liệu yêu cầu trong 1 tập tin.
  - `clean` : Lưu dữ liệu yêu cầu trong 1 tập tin và xóa tập tin đó sau khi yêu cầu được xử lý.
  - `on` : Lưu dữ liệu yêu cầu trong 1 tập tin, nhưng không xóa tập tin đó sau khi yêu cầu đó được xử lý (tùy chọn này chỉ nên được xử lý cho những mục đích kiểm tra lỗi).
- Cú pháp: `client_body_in_file_only on | clean | off;`
- Giá trị mặc định: off

- `client_body_in_single_buffer` :
  - Sử dụng trong khối : `server` , `http` , `location`
  - Định nghĩa Nginx lưu trữ/hoặc không lưu trữ dữ liệu của yêu cầu trong 1 vùng đệm trong bộ nhớ.
  - Cú pháp: `client_body_in_single_buffer [on / off];`
  - Giá trị mặc định: off

- `client_body_buffer_size` :
  - Sử dụng trong khối : `server` , `http` , `location`
  - Chỉ rõ kích thước vùng bộ nhớ đệm lưu giữ dữ liệu của các yêu cầu. Nếu kích thước này bị vượt qua, dữ liệu (hoặc ít nhất là 1 phần của nó) của yêu cầu sẽ được ghi trên đĩa. Lưu ý rằng nếu chỉ thị `client_body_in_file_only` được cho phép, dữ liệu của yêu cầu sẽ luôn được lưu thành 1 tập tin trên đĩa cứng, mà không quan tâm đến kích thước của chúng (dù chúng có phù hợp với kích thước bộ nhớ đệm hay không).
  - Cú pháp: `client_body_buffer_size [giá trị của bộ nhớ];`
  - Giá trị mặc định: 8k hoặc 16k

- `client_body_temp_path` :
  - Sử dụng trong khối : `server` , `http` , `location`
  - Cho phép chúng ta định nghĩa đường dẫn đến thư viện chứa các tập tin dữ liệu yêu cầu từ client. 1 tùy chọn bổ sung cho phép chúng ta tách riêng các tập tin này vào 1 cây thư mục lên đến 3 cấp.
  - Cú pháp: `client_body_temp_path path [level1] [level2] [level3];`
  - Giá trị mặc định:

```
client_body_temp_path /tmp/nginx_rbf;
client_body_temp_path temp 2; # Nginx sẽ tạo các thư mục có tên 2 chữ số để giữ các tập tin dữ liệu yêu cầu
client_body_temp_path temp 1 2 4; # Nginx sẽ tạo 3 cấp thư mục (cấp đầu tiên: tên có 1 chữ số, cấp thứ 2: t
```

- `client_body_timeout` :

- Sử dụng trong khối : `server` , `http` , `location`
- Định nghĩa thời gian chờ không hoạt động trong khi đọc 1 dữ liệu yêu cầu từ client. 1 kết nối trở nên không hoạt động khi client ngừng truyền dữ liệu. Nếu đạt đến độ chậm trễ (delay), Nginx trả về 1 lỗi HTTP 408 Request timeout.
- Cú pháp: `client_body_timeout [giá trị thời gian tính bằng giây]`
- Giá trị mặc định: 60

- `client_header_buffer_size` :
  - Sử dụng trong khối : `server` , `http` , `location` , `if` Các biến được chấp nhận.
  - Chỉ thị này cho phép chúng ta định nghĩa kích thước vùng đệm mà Nginx phân bổ cho các tiêu đề (header) của yêu cầu. Thông thường, 1 KB là đủ. Tuy nhiên, trong vài trường hợp, các tiêu đề chứa nhiều dữ liệu cookie hay độ dài của URI quá lớn. Nếu gặp những trường hợp đó, Nginx sẽ phân bổ 1 hoặc nhiều vùng đệm lớn hơn (kích thước của vùng đệm lớn hơn được định nghĩa trong chỉ thị large_client_header_buffers)..
  - Cú pháp: `client_header_buffer_size [giá trị dung lượng byte]`
  - Giá trị mặc định: 1k

- `client_header_timeout` :
  - Sử dụng trong khối : `server` , `http` , `location`
  - Định nghĩa thời gian chờ không hoạt động trong khi đọc 1 tiêu đề yêu cầu từ client. 1 kết nối trở nên không hoạt động khi client ngừng truyền dữ liệu. Nếu độ chậm trễ (delay) đạt đến, Nginx trả về 1 lỗi HTTP 408 Request timeout.
  - Cú pháp: `client_header_timeout [giá trị thời gian tính bằng giây]`
  - Giá trị mặc định: 60

- `client_max_body_size` :
  - Sử dụng trong khối : `server` , `http` , `location`
  - Nó là kích thước tối đa của dữ liệu yêu cầu từ client. Nếu kích thước này bị vượt qua, Nginx trả về 1 lỗi HTTP 413 Request entity too large. Thiết lập này đặc biệt quan trọng nếu chúng ta cho phép người dùng tải các tập tin lên máy chủ qua HTTP.
  - Cú pháp: `client_max_body_size [Giá trị dung lượng bộ nhớ trong Nginx.]`
  - Giá trị mặc định: 1m (MB)

- `large_client_header_buffers` :
  - Sử dụng trong khối : `server` , `http` , `location`
  - Định nghĩa số lượng và kích thước của các vùng đệm lớn hơn được dùng cho việc lưu trữ các yêu cầu từ client, trong trường hợp vùng đệm mặc định (client_header_buffer_size) không hiệu quả. Mỗi dòng của tiêu đề phải phù hợp với kích thước của 1 vùng đệm. Nếu dòng URI lớn hơn kích thước của 1 vòng đệm, Nginx trả về 1 lỗi 414 Request URI too large. Nếu 1 dòng tiêu đề khác vượt qua kích thước của 1 vùng đệm, Nginx trả về lỗi 400 Bad request.
  - Cú pháp: `large_client_header_buffers [Giá trị dung lượng bộ nhớ .]`

    - Giá trị mặc định: 32 KB

- `lingering_time` :
    - Sử dụng trong khối : `server` , `http` , `location`
    - Chỉ thị này áp dụng cho các yêu cầu từ client với dữ liệu của yêu cầu đó. Ngay khi lượng dữ liệu được tải lên vượt quá giá trị max_client_body_size, Nginx sẽ ngay lập tức gửi 1 phản hồi lỗi HTTP 413 Request entity too large. Tuy nhiên, phần lớn các trình duyệt tiếp tục tải dữ liệu lên bỏ qua thông báo này. Chỉ thị này định nghĩa lượng thời gian Nginx sẽ đợi sau khi gửi phản hồi lỗi trên trước khi đóng kết nối.
    - Cú pháp: `lingering_time [Giá trị thời gian tính bằng giây]`
    - Giá trị mặc định: 30

- `lingering_timeout` :
    - Sử dụng trong khối : `server` , `http` , `location`
    - Chỉ thị này định nghĩa lượng thời gian Nginx sẽ đợi giữa 2 tiến trình đọc trước khi đóng kết nối từ client.
    - Cú pháp: `lingering_timeout [Giá trị thời gian tính bằng giây]`
    - Giá trị mặc định: 5

- `lingering_close` :
    - Sử dụng trong khối : `server` , `http` , `location`
    - Điều khiển cách Nginx đóng các kết nối từ client. Giá trị off sẽ ngay tức khắc đóng các kết nối sau khi tất cả dữ liệu từ các yêu cầu được nhận. Giá trị mặc định (on) cho phép đợi và xử lý dữ liệu bổ sung nếu cần thiết. Nếu thiết lập là always, Nginx sẽ luôn luôn đợi để đóng kết nối. Lượng thời gian đợi được định nghĩa bởi chỉ thị lingering_timeout.
    - Cú pháp: `lingering_close [on/ off/ always]`
    - Giá trị mặc định: on

- `ignore_invalid_headers` :
    - Sử dụng trong khối : `server` , `http`
    - Nếu chỉ thị này bị vô hiệu hóa, Nginx sẽ trả về 1 lỗi HTTP 400 Bad Request trong trường hợp tiêu đề của yêu cầu bị lỗi.
    - Cú pháp: `ignore_invalid_headers [on/ off]`
    - Giá trị mặc định: on

- `chunked_transfer_encoding` :
    - Sử dụng trong khối : `server` , `http` , `location`
    - Cho phép hoặc vô hiệu hóa việc mã hóa truyền theo khối cho các yêu cầu HTTP 1.1.
    - Cú pháp: `chunked_transfer_encoding [on/ off]`
    - Giá trị mặc định: on

- `max_ranges` :

- Sử dụng trong khối : `server` , `http` , `location`
- Định nghĩa số lượng các vùng byte Nginx sẽ chấp nhận để phục vụ khi client yêu cầu 1 phần nội dung từ 1 tập tin. Nếu chúng ta không chỉ rõ giá trị, sẽ không có giới hạn cho giá trị của chỉ thị này. Nếu chúng ta thiết lập giá trị này là 0, chức năng này sẽ bị vô hiệu hóa.
- Cú pháp: `max_ranges [Giá trị dung lượng bộ nhớ trong Nginx.]`

2.4 : MIME TYPES

Multipurpose Internet Mail Extensions (MIME) là 1 chuẩn Internet mở rộng định dạng của thư điện tử (email) để hỗ trợ:

- Văn bản trong các tập ký tự khác ASCII.
- Các phần đính kèm không phải văn bản
- Dữ liệu của thông điệp với nhiều phần.
- Thông tin tiêu đề trong các tập ký tự khác ASCII.

Mặc dù MIME đươc thiết kế chủ yếu cho giao thức SMTP, ngày nay nó được sử dụng trong chỉ trong việc mô tả nội dung của email và mà còn bao gồm cả các mô tả của các loại nội dung nói chung.

Tất cả các email trên Internet được truyền qua SMTP trong định dạng MIME.

Các loại nội dung định nghĩa trong các chuẩn MIME cũng quan trọng bên ngoài phạm vi của thư điện tử, như trong các giao thức giao tiếp như HTTP. HTTP đòi hỏi rằng dữ liệu được truyền trong bối cảnh của các thông điệp giống như email, mặc dù dữ liệu thường không phải là những email thực sự

- `types` :
  - Sử dụng trong khối : `server` , `http` , `location`
  - Chỉ thị này cho phép chúng ta tạo các mối tương quan giữa các loại MIME và các phần mở rộng tập tin. Nó thực sự là 1 khối chấp nhận 1 cú pháp đặc biệt:

```
types {
  mimetype1 extension1;
  mimetype2 extension2 [extension3…];
  […]
}
```

Khi Nginx phục vụ 1 tập tin, nó kiểm tra phần mở rộng của tập tin để quyết định loại MIME. Loại MIME này sau đó được gửi như giá trị của trường Content-Type trong tiêu đề HTTP trong phản hồi của máy chủ với yêu cầu từ client. Tiêu đề này có thể ảnh hưởng đến cách các trình duyệt xử lý các tập tin

Ví dụ:

> Nếu loại MIME của tập tin là chúng ta yêu cầu là application/pdf, trình duyệt của chúng ta sẽ cố gắng đọc tập tin đó bằng việc sử dụng 1 plugin tương ứng với loại MIME đó thay cho việc tải tập tin đó về. Nói cách khác, chúng ta sẽ có thể đọc nội dung của tập tin pdf trên trình duyệt, thay cho việc trình duyệt sẽ tải tập tin pdf đó về máy của chúng ta.

Nginx bao gồm 1 tập cơ bản các loại MIME trong 1 tập tin riêng biệt (mime.types), chúng ta có thể bao hàm nội dung của tập tin này vào cấu hình của Nginx bằng việc sử dụng chỉ thị include.

```
include mime.types;
```

Tập tin này đã bao gồm các phần mở rộng tập tin quan trong nhất, vì vậy chúng ta có lẽ không cần phải điều chỉnh nó. Nếu phần mở rộng của tập tin được phục vụ không được tìm thấy trong các loại đã được liệt kê, loại mặc định được sử dụng, định nghĩa trong chỉ thỉ default_type.

Giá trị mặc định, nếu tập tin mime.types không được đính kèm, là:

```
types {
    text/html html;
    image/gif gif;
    image/jpeg jpg;
}
```

- `default_type` :
  - Sử dụng trong khối : `server` , `http` , `location`
  - Định nghĩa loại MIME mặc định. Khi Nginx phục vụ 1 tập tin, phần mở rộng của tập tin này được đối chiếu với các loại đã biết được khai báo bên trong các khối types để trả về loại MIME chính xác như giá trị của trường Content-Type trong tiêu đề HTTP. Nếu phần mở rộng không khớp với bất kỳ loại nào, giá trị của chỉ thị default_type được dùng.
  - Cú pháp: `default_type [loại MIME];`
  - Giá trị mặc định: text/plain

- `types_hash_max_size` :
  - Sử dụng trong khối : `server` , `http` , `location`
  - Định nghĩa kích thước tối đa của 1 entry trong bảng băm các loại MIME.
  - Cú pháp: `types_hash_max_size [giá trị số ];`
  - Giá trị mặc định: 4k hoặc 8k (1 dòng của CPU cache)

2.5 : Các chỉ thị về giới hạn

- `limit_except` :
  - Sử dụng trong khối : `location`
  - Chỉ thị này cho phép chúng ta ngăn chặn việc sử dụng của tất cả các phương thức HTTP, ngoại trừ những phương thức mà chúng ta cho phép 1 cách tường minh. Bên trong 1 khối location, chúng ta có thể giới hạn việc sử dụng 1 vài phương thức HTTP, như cấm client gửi các yêu cầu POST. Chúng ta cần định nghĩa 2 thành phần – đầu tiên, những phương thức không bị cấm, và thứ 2, những client được ảnh hưởng bởi việc giới hạn này:

```
location /admin/ {
  limit_except GET {
        allow 192.168.1.0/24;
        deny all;
  }
}
```

- Cú pháp:

```
limit_except METHOD1 [METHOD2…] {
  allow | deny | auth_basic | auth_basic_user_file | proxy_pass | perl;
}
```

- `limit_rate` :
  - Sử dụng trong khối : `server` , `http` , `location` , `if`
  - Cho phép chúng ta giới hạn tỉ lệ truyền của các kết nối khách hàng cá nhân. Tỉ lệ này được thể hiện ở dạng số byte trên mỗi giây (bytes per second):

```
limit_rate 500k;
```

- Điều này sẽ giới hạn tỉ lệ truyền kết nối là 500 KB/second. Nếu 1 khách hàng mở 2 kết nối, khách hàng sẽ được cho phép 2*500 KB

- Cú pháp: `limit_rate [Giá trị dung lượng trong Nginx.]`

- Giá trị mặc định: Không có

- `limit_rate_after` :
  - Sử dụng trong khối : `server` , `http` , `location` , `if`
  - Định nghĩa số lượng dữ liệu được truyền trước khi chỉ thị limit_rate có tác dụng.:

```
limit_rate_after 10m;
```

- Nginx sẽ gửi 10 MB đầu tiên với tốc độ tối đa. Qua kích thước này, tỉ lệ truyền được giới hạn bởi giá trị được chỉ rõ trong khai báo limit_rate. Tương tự với chỉ thị limit_rate, thiết lập này chỉ áp dụng cho 1 kết nối.

- Cú pháp: `limit_rate_after [Giá trị dung lượng trong Nginx.]`

- Giá trị mặc định: Không có

- `satisfy` :
  - Sử dụng trong khối : `location`
  - Chỉ thị satisfy định nghĩa việc khách hàng được yêu cầu đáp ứng tất cả các điều kiện truy cập (satisfy all) hoặc ít nhất 1 điều kiện (satisfy any) để là khách hàng hợp lệ.:

```
location /admin/ {
  allow 192.168.1.0/24;
  deny all;
  auth_basic “Authentication required”;
  auth_basic_user_file conf/htpasswd;
}
```

- Trong ví dụ trên, có 2 điều kiện cho khách hàng để có thể truy cập:
  - Qua chỉ thị allow và deny, chúng ta chỉ cho phép các khách hàng có địa chỉ IP cục bộ, tất cả các khách hàng khác sẽ bị từ chối.

    - Qua chỉ thị auth_basic và auth_basic_user_file, chúng ta chỉ cho phép các khách hàng cung cấp 1 username và password hợp lệ

- Với satisfy all, khách hàng phải thoả mãn cả 2 điều kiện trên.
- Với satisfy any, khách hàng chỉ cần thoả mãn 1 trong 2 điều kiện trên
- Cú pháp: `satisfy any | all;`
- Giá trị mặc định: all
- `internal` :
    - Sử dụng trong khối : `location`
    - Chỉ thị này chỉ rõ rằng khối location này là nội bộ. Nói cách khác, các tài nguyên được chỉ rõ không thể được truy cập từ các yêu cầu bên ngoài.

```
server {
    location /admin/ {
        internal;
    }
}
```

- Với cấu hình trên, khách hàng sẽ không thể truy cập vào đường dẫn http://abc.com/admin/. Những yêu cầu như vậy sẽ gặp lỗi HTTP 404 Not Found. Cách duy nhất để truy cập vào các tài nguyên này là qua các chuyển hướng nội bộ.

2.6 : Các chỉ thị về xử lý tệp tin và bộ nhớ đệm

- `disable_symlinks` :
    - Chỉ thị này cho phép chúng ta điều khiển cách Nginx xử lý các liên kết tượng trưng khi chúng được phục vụ. Mặc định (giá trị mặc định là off) các liên kết này được cho phép và Nginx sẽ xử lý chúng. Chúng ta có thể quyết định vô hiệu hóa việc cho phép các liên kết tượng trưng dưới những điều kiện khác nhau bằng việc chỉ rõ 1 trong những giá trị sau:
        - `on` : Nếu bất kỳ phần nào của URI là 1 liên kết tượng trưng, truy cập đến nó sẽ bị từ chối và Nginx trả về 1 trang lỗi HTTP 403.
        - `if_not_owner` : Tương tự với giá trị on ở trên, nhưng truy cập bị từ chối chỉ khi liên kết và đối tượng nó trỏ đến có owner khác nhau.
        - Thông số tùy chọn từ `from` = cho phép chúng ta chỉ rõ 1 phần của URL mà sẽ không được kiểm tra liên kết tượng trưng. Ví dụ, nếu ta khai báo là `disable_symlinks from=$document_root;` sẽ yêu cầu Nginx theo các liên kết tượng trưng trong URI đến thư mục $document_root. Nếu 1 liên kết tượng trưng được tìm thấy trong các phần URI sau phần $document_root đó, truy cập đến tập tin được yêu cầu sẽ bị từ chối.
    - Note : `Symbolic Link` – liên kết tượng trưng (hoặc `symlink` hay `soft link`) là 1 loại tập tin đặc biệt chứa thông tin tham chiếu đến 1 tập tin hoặc thư mục khác (targer) trong hệ thống dưới dạng 1 đường dẫn tuyệt đối hoặc tương đối.

- `directio` :

- Sử dụng trong khối : `server` , `http` , `location`
- Nếu chỉ thị này được cho phép, các tập tin với kích thước lớn hơn giá trị được chỉ rõ sẽ được đọc với cơ chế hệ thống Direct I/O. Điều này cho phép Nginx đọc dữ liệu từ các thiết bị lưu trữ và đặt chúng trực tiếp trong bộ nhớ mà không cần tiến trình đưa vào bộ nhớ đệm trung gian.
- Cú pháp: `directio [Giá trị dung lượng trong Nginx, hoặc là off];`
- Giá trị mặc định: off

- `directio_alignment` :
  - Sử dụng trong khối : `server` , `http` , `location`
  - Thiết lập liên kết byte khi sử dụng directio. Thiết lập giá trị này là 4k nếu sử dụng XFS trên Linux.
  - Cú pháp: `directio [Giá trị dung lượng trong Nginx ];`
  - Giá trị mặc định: 512

- `open_file_cache` :
  - Sử dụng trong khối : `server` , `http` , `location`
  - Chỉ thị này cho phép chúng ta cho phép chúng ta sử dụng bộ nhớ cache lưu trữ thông tin về các tập tin được mở. Nó thực sự không lưu trữ nội dung của chính tập tin đó mà chỉ lưu trữ thông tin như:
    - Mô tả tập tin (kích thước, thời gian điều chỉnh, ...).
    - Sự tồn tại của tập tin và thư mục.
    - Các lỗi của tập tin, như là `Permission denied` , `File not found` , ... Lưu ý rằng chỉ thị này có thể bị vô hiệu hóa với chỉ thị `open_file_cache_errors` .
  - Chỉ thị này chấp nhận 2 tham số:
    - `max=X` , với X là số mục mà bộ nhớ cache có thể lưu trữ. Nếu số lượng này bị vượt qua, các mục cũ hơn sẽ được xóa để lấy không gian cho các mục mới hơn.
    - `inactive=Y` (tùy chọn, không bắt buộc) với Y là số giây mà 1 mục trong bộ nhớ cache sẽ được lưu trữ. Mặc định, Nginx sẽ đợi 60 giây trước khi xóa 1 mục trong bộ nhớ cache. Nếu mục trong bộ nhớ cache được truy cập, bộ đếm thời gian được khởi động lại. Nếu mục trong bộ nhớ cache được truy cập nhiều hơn giá trị định nghĩa bởi chỉ thị open_file_cache_min_users, mục đó sẽ không bị xóa (đến khi Nginx hết bộ nhớ và quyết định xóa các mục cũ hơn).
  - Cú pháp: `open_file_cache max=X [inactive=Y] | off`

```
open_file_cache max=5000 inactive=180;
```

- Giá trị mặc định: off

- `open_file_cache_min_uses` :
  - Sử dụng trong khối : `server` , `http` , `location`
  - Mặc định: các mục trong chỉ thị `open_file_cache` được xóa sau 1 khoảng thời gian không được sử dụng (mặc định là 60 giây). Chúng ta có thể ngăn Nginx xóa các mục này. Chỉ thị này định nghĩa số lần 1 mục trong bộ nhớ cache phải được truy cập để có đủ điều để bảo vệ.

```
        open_file_cache_min_uses 3;
```

Nếu 1 mục trong bộ nhớ cache được truy cập nhiều hơn 3 lần, nó chuyển trạng thái thành hoạt động vĩnh viễn và sẽ không bị xóa đến khi Nginx quyết định xóa các mục cũ hơn để giải phóng vùng nhớ.

Cú pháp: `open_file_cache_min_uses [giá trị số];`

Giá trị mặc định: 1

- `open_file_cache_valid` :
  - Sử dụng trong khối : `server` , `http` , `location`
  - Cơ chế lưu trong bộ nhớ cache các tập tin mở là quan trọng, nhưng thông tin được lưu trong bộ nhớ cache nhanh chóng trở thành lỗi thời. Về phương diện này, thông tin cần được tái kiểm tra sau 1 khoảng thời gian ngắn. Chỉ thị này chỉ rõ số giây mà Nginx sẽ đợi trước khi tái xác nhận 1 mục trong bộ nhớ cache.
  - Cú pháp: `open_file_cache_valid [giá trị thời gian tính bằng giây]`
  - Giá trị mặc định: 60s
- `read_ahead` :
  - Sử dụng trong khối : `server` , `http` , `location`
  - Định nghĩa số byte sẽ được đọc trước từ các tập tin. Dưới các hệ điều hành dựa trên Linux, việc thiết lập chỉ thị này này 1 giá trị lớn hơn 0 sẽ cho phép việc đọc trước, nhưng giá trị thực sự chúng ta chị định không có tác dụng. Thiết lập chỉ thị này là 0 để vô hiệu hóa việc đọc trước.
  - Cú pháp: `read_ahead [Giá trị dung lượng trong Nginx.]`
  - Giá trị mặc định: 0

2.5 : Các chỉ thị khác

- `log_not_found` :
  - Sử dụng trong khối : `server` , `http` , `location`
  - Cho phép hoặc vô hiệu hóa việc ghi nhận các lỗi HTTP 404 Not Found.
  - Cú pháp: `log_not_found [on | off]`
  - Giá trị mặc định: on
- `log_subrequest` :
  - Sử dụng trong khối : `server` , `http` , `location`
  - Cho phép hoặc vô hiệu hóa việc ghi nhận các yêu cầu con được kích hoạt bởi các chuyển hướng nội bộ hoặc các yêu cầu SSI.
  - Cú pháp: `log_subrequest [on | off]`
  - Giá trị mặc định: on
- `merge_slashes`

- Sử dụng trong khối : `server` , `http` , `location`
- Cho phép chỉ thị này sẽ có ảnh hưởng đến việc sát nhập nhiều ký tự "/" liên tiếp trong URI.

```
server {
[…]
server_name abc.com;
    location /documents/ {
        type {}
        default_type text/plain;
     }
}
```

- Mặc định, nếu khách hàng cố gắng truy cập http://abc.com//documents/ (lưu ý // ở giữa URI), Nginx sẽ trả về lỗi HTTP 404 Not Found. Nếu cho phép chỉ thị này, 2 ký hiệu / sẽ được hợp thành 1 và mẫu location sẽ khớ
- Cú pháp: `merge_slashes [on | off]`
- Giá trị mặc định: on
- `msie_padding` :
  - Sử dụng trong khối : `server` , `http` , `location`
  - Chỉ thị này được dùng với các trình duyệt Microsoft Internet Explorer (MSIE) và Google Chrome. Trong trường hợp các trang lỗi (với mã lỗi là 400 hoặc cao hơn), nếu kích thước của dữ liệu phản hồi nhỏ hơn 512 Byte, những trình duyệt này sẽ hiển thị trang lỗi của chính những trình duyệt đó. Nếu cho phép chỉ thị này, phần thân của phản hồi với 1 mã trạng thái 400 hoặc cao hơn sẽ được đệm thêm đến 512 Byte.
  - Cú pháp: `msie_padding [on | off]`
  - Giá trị mặc định: off
- `resolver` :
  - Sử dụng trong khối : `server` , `http` , `location`
  - Chỉ rõ các máy chủ phân giải tên miền (DNS) được sử dụng bởi Nginx để phân giải hostname của các địa chỉ IP và ngược lại. Các kết quả truy vấn DNS được lưu trong bộ nhớ cache trong 1 thời gian, hoặc được chỉ rõ bởi giá trị TTL (Time-to-live) của máy chủ DNS, hoặc được chỉ rõ bởi giá trị thời gian cho đối số hợp lệ.
  - Cú pháp: `resolver [địa chỉ id], [valid=Time value]`
  - Ví dụ :

```
resolver 8.8.8.8 8.8.4.4 valid=1h; # use Google DNS and cache results for 1 hour
```

- `resolver_timeout` :
  - Sử dụng trong khối : `server` , `http` , `location`
  - Giới hạn thời gian cho 1 truy vấn phân giải hostname.
  - Cú pháp: `resolver_timeout [Giá trị thời gian trong Nginx (tính bằng giây)]`
  - Giá trị mặc định: 30s
- `server_tokens` :

    - Sử dụng trong khối : `server , http , location`
    - Chỉ thị này cho phép chúng ta định nghĩa việc cho phép/không cho phép Nginx thông báo cho khách hàng biết về phiên bản Nginx đang chạy tại máy chủ. Có 2 tình huống Nginx chỉ số phiên bản của nó là:
        - Trong phần server header của những phản hồi HTTP (như là nginx/1.2.9). Nếu chúng ta thiết lập server_tokens là off, phần server header sẽ chỉ có giá trị là Nginx.
        - Trên các trang lỗi, Nginx chỉ số phiên bản trong phần footer. Nếu chúng ta thiết lập server_tokens là off, phần footer của trang lỗi sẽ chỉ là Nginx.
    - Cú pháp: `server_tokens [on | off]`
    - Giá trị mặc định: on

- `underscores_in_headers` :
    - Sử dụng trong khối : `server , http , location`
    - Cho phép hoặc không cho phép các dấu gạch dưới (_) trong các tên header HTTP tùy chọn. Nếu chỉ thị này được để là on, phần header ví dụ sau sẽ được xem là hợp lệ bởi Nginx: `test_header: value`
    - Cú pháp: `underscores_in_headers [on | off]`
    - Giá trị mặc định: off

- `variables_hash_max_size` :
    - Sử dụng trong khối : `http`
    - Cho phép hoặc không cho phép các dấu gạch dưới (_) trong các tên header HTTP tùy chọn. Nếu chỉ thị này được để là on, phần header ví dụ sau sẽ được xem là hợp lệ bởi Nginx: `test_header: value`
    - Cú pháp: `variables_hash_max_size [Giá trị số]`
    - Giá trị mặc định: 512

- `variables_hash_bucket_size` :
    - Sử dụng trong khối : `http`
    - Chỉ thị này cho phép chúng ta thiết lập kích thước vùng chứa cho các bảng băm chứa các biến.
    - Cú pháp: `variables_hash_bucket_size [Giá trị số]`
    - Giá trị mặc định: 64 (hoặc 32, hoặc 128, phụ thuộc vào bộ nhớ cache của bộ xử lý).

- `post_action` :
    - Sử dụng trong khối : `http , server , location , if`