

【Nginx】config 詳解

基本配置區塊

```
#config 區塊 基本配置
...      # 全域性區塊

event{      # events 區塊
    ...
}
http{      # http 區塊
    server{      # server 區塊
        location{      # location 區塊
            ...
        }
    }
}
```

server 區塊的配置

```
server{
    listen 80;
    listen [::]80;
    server_name example.com example2.com;
    location /{
        root /usr/share/nginx/html;
        index index.html index.htm;
    }
}
```

#listen 80; 代表監聽所有 ipv4 的位址
#listen [::]80; 代表監聽所有 ipv6 的位址
#server_name 是你的 Domain 名稱。可以使用空白listen 多個domain
#location 中則是指定對不同路徑要怎麼處理

```
# 如果沒有下default_server; 設定，第一個server 區塊會變成預設所以有位被匹配到的設定
# 避免除錯上的困難，可設定一個預設區塊，在未匹配到server 時，顯示403錯誤
server {
    listen 80 default_server;
    server_name _;

    set $cache_status -;
    set $parameter -;
    set $response_body_size 0;
    location / {
        default_type 'text/html';
        echo "default server" ;
        return 403; # 403 forbidden
    }
}
```

【反向代理】reverse proxy

```
server {

    listen 80;
    server_name www.example.com;

    location \ {
        proxy_pass http://192.168.1.1:8080;

        proxy_set_header Host    $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
```

```
    proxy_set_header X-Forwarded-Proto $scheme;
  }
}
```

【快取】實作圖片快取機制

```
server {

    listen 80;
    server_name www.example.com;

    location ~*\.(jpg|jpeg|png|css|js)$ {
        expires 365d; # 快取時間
        proxy_pass http://loaclhost:8080;
    }
}
```

【virtural host】實作 virtural host

```
# .web1
server {
    listen    80;
    server_name web1.example.com ;
    root /usr/local/nginx/web1;
    index index.html;
}

# .web2
server {
    listen    80;
    server_name web2.example.com ;
    root /usr/local/nginx/web2;
    index index.html;
}
```

【負載平衡】 load balance

#Nginx 提供了以下三種 load balancing 方法：

#round-robin：預設值，會將請留輪流平均分配到每台伺服器上

#lest-connected：會將請求分配到目前連接數最少的伺服器上

#ip-hash：利用 hash-function 來決定使用者要被分配到的伺服器，此方法可以達到同一個使用者 (IP address) 每次連結的伺服器都是相同的

#如果要從默認值 round-robin 方法改成 lest-connected 或 ip-hash 的方法，只需要在第一行加上lest_conn; 或 ip_hash; 即可

```
upstream myapp {
    ip_hash;
    server srv1.example.com;
    server srv2.example.com;
    server srv3.example.com;
}
server {
    listen 80;
    location / {
        proxy_pass http://myapp;
    }
}
```

weight 默認值為 1，

以下的配置代表如果有 5 次新的請求，則會有 3 次被分配到 srv1 和分配各 1 次到 srv2 srv3 上

```
upstream myapp1 {
    server srv1.example.com weight=3;
    server srv2.example.com;
    server srv3.example.com;
    server bak.example.com; #backup 代表，所有伺服器都掛掉之後，此伺服器才會生效
}
server {
    listen 80;
```

```
location / {
    proxy_pass http://myapp1;
}
}
```

限制來源

```
location / {
    deny 192.168.1.1;
    allow 192.168.1.0/24;
    allow 10.1.1.0/16;
    allow 2001:0db8::/32;
    deny all;
}
```

nginx.conf 詳細配置

```
#定义Nginx运行的用户和用户组
user www www;

#nginx進程數，建議設置為等於CPU總核心數。
worker_processes 8;

#全局錯誤日誌定義類型，[ debug | info | notice | warn | error | crit ]
error_log /var/log/nginx/error.log info;

#進程文件
pid /var/run/nginx.pid;

#一個nginx進程打開的最多文件描述符數目，理論值應該是最多打開文件數（系統的值ulimit -n）與nginx進程數相除，但是nginx分配請求並不均勻，所以建議與ulimit -n的值保持一致。
worker_rlimit_nofile 65535;

#工作模式與連接數上限
events
{
    #參考事件模型，use [ kqueue | rtsig | epoll | /dev/poll | select | poll ]; epoll模型是Linux 2.6以上版本內核中的高性能網絡I/O模型，如果跑在FreeBSD上面，就用kqueue模型。
    use epoll;
    #單個進程最大連接數（最大連接數=連接數*進程數）
    worker_connections 65535;
}

#設定http服務器
http
{
    include mime.types; #文件擴展名與文件類型映射表
    default_type application/octet-stream; #默認文件類型
    #charset utf-8; #默認編碼
    server_names_hash_bucket_size 128; #服務器名字的hash表大小
    large_client_header_buffers 4 64k; #設定請求緩
    autoindex on; #開啟目錄列表訪問，合適下載服務器，默認關閉。

    #此指令設置 Nginx 使用的 DNS 解析器的配置。在這個例子中，使用了兩個 DNS 解析器的 IP 地址：8.8.8.8 和 168.95.1.1。
    #ipv6=off 表示禁用 IPv6 解析。valid=300s 表示 DNS 解析的結果將在 300 秒（5 分鐘）內被緩存。
    resolver 8.8.8.8 168.95.1.1 ipv6=off valid=300s;

    #此指令設置 DNS 解析的超時時間。如果在指定的時間內無法解析域名，Nginx 將中斷解析並返回錯誤。
    resolver_timeout 3s;

    client_body_buffer_size 256K; #此指令指定客戶端請求主體的緩衝區大小，這是用於存儲客戶端發送的請求主體數據的內存區域。
    client_body_timeout 3s; #此指令設置接收客戶端請求主體的超時時間。如果在指定的時間內沒有接收到完整的請求主體，Nginx 將結束該連接。
```

client_header_buffer_size 64k; #此指令指定用於存儲客戶端請求標頭的緩衝區大小。(上傳文件大小限制?)

client_header_timeout 3s; #此指令設置接收客戶端請求標頭的超時時間。如果在指定的時間內沒有接收到完整的請求標頭，Nginx 將結束該連接。

client_max_body_size 1m; #此指令設置允許客戶端發送的最大請求主體大小。如果客戶端嘗試發送超過此大小的請求主體，Nginx 將返回 413 Request Entity Too Large 錯誤。

large_client_header_buffers 8 64k; #此指令設置用於存儲大型客戶端請求標頭的緩衝區大小。如果客戶端請求的標頭超過指定的大小，Nginx 將使用這些緩衝區來處理它。

keepalive_timeout 5s; #此指令設置持久連接的超時時間。如果在指定的時間內沒有新的請求到達，Nginx 將關閉持久連接。

open_file_cache max=5000 inactive=20s; #此指令啟用了文件打開時的緩存機制，以加快文件的讀取速度。max 參數指定緩存的最大數量，inactive 參數指定文件在緩存中保持不活躍的時間。

open_file_cache_valid 30s; #此指令設置緩存中的文件信息的有效期限。在此期限內，Nginx 將重用緩存的文件信息而不需要再次訪問磁盤。

open_file_cache_min_uses 8; #此指令設置文件緩存機制中的最小使用次數。只有當某個文件被訪問的次數超過此設定值時，才會將該文件的信息緩存起來。

open_file_cache_errors on; #此指令設置是否緩存文件打開期間的錯誤。當設置為 "on" 時，如果在打開文件時發生錯誤，Nginx 會將該錯誤信息緩存起來，以避免重複出現相同的錯誤。

send_timeout 10; #此指令設置向客戶端發送數據的超時時間。如果在指定的時間內沒有將數據發送給客戶端，Nginx 將結束該連接。

#開啟高效文件傳輸模式，sendfile指令指定nginx是否調用sendfile函數來輸出文件，對於普通應用設為on，用於高效地將文件內容發送給客戶端，而無需將文件數據從磁盤讀入到用戶空間。

#如果用來進行下載等應用磁盤IO重負載應用，可設置為off，以平衡磁盤與網絡I/O處理速度，降低系統的負載。注意：如果圖片顯示不正常把這個改成off。

sendfile on;

sendfile_max_chunk 512k; #此指令設置單次 sendfile 操作的最大數據塊大小。如果需要發送的文件數據大於此值，Nginx 將分段進行 sendfile 操作。

server_tokens off; #此指令關閉 Nginx 的版本號顯示。在生產環境中，關閉版本號的顯示可以增加安全性，防止攻擊者利用已知漏洞進行攻擊。

tcp_nodelay on; #此指令啟用 TCP 即時發送功能。它禁用了 Nagle 算法，從而降低了數據傳輸的延遲，但可能會增加網絡流量。

tcp_nopush on; #此指令啟用 TCP 壓縮機制。當服務器將數據發送給客戶端時，它將盡可能地將多個小的數據包包含並為一個較大的數據包發送，從而減少了網絡傳輸的開銷。

types_hash_max_size 2048; #此指令設置 Nginx 在處理 MIME 類型時使用的哈希表的大小。增大此值可以提高性能

#FastCGI相關參數是為了改善網站的性能：減少資源佔用，提高訪問速度。下面參數看字面意思都能理解。

fastcgi_connect_timeout 300;

fastcgi_send_timeout 300;

fastcgi_read_timeout 300;

fastcgi_buffer_size 64k;

fastcgi_buffers 4 64k;

fastcgi_busy_buffers_size 128k;

fastcgi_temp_file_write_size 128k;

#gzip模塊設置

gzip on; #開啟gzip壓縮輸出

gzip_min_length 1k; #最小壓縮文件大小

gzip_buffers 4 16k; #壓縮緩衝區

gzip_http_version 1.0; #壓縮版本（默認1.1，前端如果是squid2.5請使用1.0）

gzip_comp_level 2; #壓縮等級

gzip_types text/plain application/x-javascript text/css application/xml;

#壓縮類型，默認就已經包含text/html，所以下面就不用再寫了，寫上去也不會有問題，但是會有一個warn。

gzip_vary on;

#limit_zone crawler \$binary_remote_addr 10m; #開啟限制IP連接數的時候需要使用

upstream blog.ha97.com {

#upstream的負載均衡，weight是權重，可以根據機器配置定義權重。weight參數表示權值，權值越高被分配到的機率越大。

server 192.168.80.121:80 weight=3;

server 192.168.80.122:80 weight=2;

server 192.168.80.123:80 weight=3;

}

#虛擬主機的配置

server

{

#監聽端口

listen 80;

#域名可以有多个，用空格隔開

server_name www.ha97.com ha97.com;

index index.html index.htm index.php;

root /data/www/ha97;

```

location ~ .*\.php|php5)?$
{
    fastcgi_pass 127.0.0.1:9000;
    fastcgi_index index.php;
    include fastcgi.conf;
}
#圖片緩存時間設置
location ~ .*\.gif|jpg|jpeg|png|bmp|swf)$
{
    expires 10d;
}
#JS和CSS緩存時間設置
location ~ .*\.js|css)?$
{
    expires 1h;
}
#日誌格式設定
log_format access '$remote_addr - $remote_user [$time_local] "$request" '
'$status $body_bytes_sent "$http_referer" '
'"$http_user_agent" $http_x_forwarded_for';
#定義本虛擬主機的訪問日誌
access_log /var/log/nginx/ha97access.log access;


#對"/" 啟用反向代理
location / {
    proxy_pass http://127.0.0.1:88;
    proxy_redirect off;
    proxy_set_header X-Real-IP $remote_addr;
    #後端的Web服務器可以通過X-Forwarded-For獲取用戶真實IP
    proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
    #以下是一些反向代理的配置，可選。
    proxy_set_header Host $host;
    client_max_body_size 10m; #允許客戶端請求的最大單文件字節數
    client_body_buffer_size 128k; #緩衝區代理緩衝用戶端請求的最大字節數，
    proxy_connect_timeout 90; #nginx跟後端服務器連接超時時間(代理連接超時)
    proxy_send_timeout 90; #後端服務器數據回傳時間(代理發送超時)
    proxy_read_timeout 90; #連接成功後，後端服務器響應時間(代理接收超時)
    proxy_buffer_size 4k; #設置代理服務器（nginx）保存用戶頭信息的緩衝區大小
    proxy_buffers 4 32k; #proxy_buffers緩衝區，網頁平均在32k以下的設置
    proxy_busy_buffers_size 64k; #高負荷下緩衝大小（proxy_buffers*2）
    proxy_temp_file_write_size 64k;
    #設定緩存文件夾大小，大於這個值，將從upstream服務器傳
}


#設定查看Nginx狀態的地址
location /NginxStatus {
    stub_status on;
    access_log on;
    auth_basic "NginxStatus";
    auth_basic_user_file conf/htpasswd;
    #htpasswd文件的內容可以用apache提供的htpasswd工具來產生。
}


#本地動靜分離反向代理配置
#所有jsp的頁面均交由tomcat或resin處理
location ~ .(jsp|jspx|do)?$ {
    proxy_set_header Host $host;
    proxy_set_header X-Real-IP $remote_addr;
    proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
    proxy_pass http://127.0.0.1:8080;
}
#所有靜文件由nginx直接取不經tomcat或resin
location ~ .*.(htm|html|gif|jpg|jpeg|png|bmp|swf|ioc|rar|zip|txt|flv|mid|doc|ppt|pdf|xls|mp3|wma)$
{ expires 15d; }

```

```
location ~ .*.(js|css)?$  
{ expires 1h; }  
}  
}
```

🕒修訂版本 #9

★由 treeman 建立於 25 🕒@🕒🕒🕒 2023 12:50:24

✍由 treeman 更新於 1 🕒🕒🕒🕒 2023 10:40:11