Wstęp

Ostatnio przeszedłem z Apache na Nginx'a i postanowiłem napisać małe „how to” jak to wykonać - może komuś się przyda. Czemu się zdecydowałem na Nginx'a? Apache zajmuje dużo pamięci oraz jest mniej wydajny.

Instalacja

```
poldek:/all-avail> install nginx-light
```

Konfiguracja

Opis pliku konfiguracji **/etc/nginx/nginx-light.conf** (opisuje to co możemy zmieniać):

```
user                    nginx nginx;
worker_processes        5;
error_log               /var/log/nginx/nginx-light_error.log;
pid                     /var/run/nginx-light.pid;
```

Kolejno:

- pierwsza linia definiuje na jakim użytkowniku ma być uruchomiona usługa,
- worker_process - ile ma być uruchomionych wątków Nginx'a,
- error_log definiuje plik, gdzie będą zapisywane błędy,
- pid - gdzie ma być zapisywany process ID - nie zmieniamy.

```
events {
        worker_connections      2048;
        use epoll;
}
```

Worker_connections - ilość requestów na wątek.

```
http {
        include         /etc/nginx/mime.types;
        default_type    application/octet-stream;

        log_format      main    '$remote_addr - $remote_user [$time_local]
$request '
                                '"$status" $body_bytes_sent "$http_referer"
'
                                '"$http_user_agent"
"$http_x_forwarded_for"';
        access_log      /var/log/nginx/nginx-light_access.log   main;
        sendfile        on;
        tcp_nopush      on;
```

```
        tcp_nodelay     on;
        server_names_hash_bucket_size   128;
        types_hash_max_size     2048;
        types_hash_bucket_size  64;
        #keepalive_timeout      0;
        keepalive_timeout       65;
        limit_zone      test-limit      $binary_remote_addr     10m;
        #gzip   on;
        server {
                listen          80;
                server_name     localhost;
                access_log      /var/log/nginx/nginx-light_access.log  main;
                client_max_body_size    10M;

                location / {
                        autoindex       on;
                        root    /home/services/nginx/html;
                        index   index.html index.htm index.php;
                        limit_conn      test-limit      15;
                }


#               location /nginx_status {
#                       stub_status     on;
#                       access_log      off;
#                       allow   127.0.0.1;
#                       deny    all;
#               }

#               error_page      404     /404.html;
#               error_page      500 502 503 504 /50x.html;
#               location = /50x.html {
#                       root    /home/services/http/error-pages;
#               }
#               location = /404.html {
#                       root    /home/services/http/error-pages;
#               }

#               location ~ \.php$ {
#                       include         /etc/nginx/fastcgi.params;
#                       fastcgi_pass    127.0.0.1:1026;
#                       fastcgi_index   index.php;
#                       fastcgi_param   SCRIPT_FILENAME
/home/services/nginx/html$fastcgi_script_name;
#               }

        }

}
```

Sekcja http:

- log_format - definiujemy jak ma wyglądać wpis w logach - listę zmiennych znajdziemy tu: http://wiki.nginx.org/HttpCoreModule#Variables,
- access_log - plik gdzie zapisujemy logi,
- gzip - kompresja (opis niżej),

Sekcja server:

- listen - na jakim porcie ma nasłuchiwać vhost, pierwszy vhost jest domyslnym vhostem, możliwe kombinacje co do portu i adres IP to: listen IP:port; gdzie * = 0.0.0.0 (np *:80),
- server_name - definiujemy vhost'a, a po spacji definiujemy aliasy, aczkolwiek przy domyślnym vhoście nie ma to znaczenia,
- location / - tu ustawiamy opcje dotyczącego katalogu root w vhoście,
- autoindex - on włącza tworzenie się autoindeksów,
- root - definiowania katalogu root dla danego vhost'a,
- index - tu definiujemy domyslne pliki, które będą otwierane po wejściu do katalogu,
- location /nginx_status - włącza status Nginx'a pod danym virtualnym katalogiem vhost'a - przydatne do statystyk (opis niżej),
- error_page - definiowanie plików w przypadku błędów - 404, 500, itp,
- location ~ \.php$ - definiowanie obsługi plików PHP przez demona FastCGI (opis niżej).

Wszelkie zmiany powinny zostać przetestowane składniowo przez Nginx'a, służy to tego komenda:

```
$ nginx-light -t
```

i powinna ona zwrócic coś takiego:

```
$ the configuration file /etc/nginx/nginx-light.conf syntax is ok
$ configuration file /etc/nginx/nginx-light.conf test is successful
```

w przypadku błędu pojawi się informacja gdzie jest błąd.

Logi z błędami w krótkim czasie robią się bardzo duże - to głównie za sprawą różnego rodzaju botów. Przydatną opcją jest wyłączenie logownia próby pobrania plików/katalogów których nie ma na serwerze, a ślady w logach są. Wyłączyć możemy za pomocą przydatnej opcji:

```
http {
...
           log_not_found off;
...
}
```

IPv6

Najpierw trzeba sprawdzić czy Nginx został skompilowany z opcją –with-ipv6. Aby to sprawdzić wykonajmy:

```
$ nginx-light -V
```

Jeśli Nginx obsługuje IPv6 to w sekcji serwer danego vhosta modyfikujemy listen:

```
listen          [::]:80;
```

taki wpis oznacza, że serwer nasłuchuje na IPv4 oraz IPv6, jeśli chcemy, aby tylko i wyłącznie nasłuchiwał na IPv6 ustawiamy:

```
listen          [::]:80 ipv6only=on;
```

zamiast [::] możemy podać adres (np: [ffff:a:b:c::01]).

SSL

Tworzymy certyfikaty SSL. Aby to zrobić trzeba zainstalować narzędzia Openssl'a:

```
$ install openssl-tools
```

Generowanie certyfikatów:

```
$ openssl genrsa -out /etc/nginx/server.key 1024
$ openssl req -new -x509 -days 365 -key /etc/nginx/server.key -out
/etc/nginx/server.crt
$ chmod 600 /etc/nginx/server.*
```

Do sekcji http dopisujemy:

```
ssl_certificate      server.crt;
ssl_certificate_key  server.key;
```

w sekcji **server** głównego vhosta wpisujemy:

```
listen          443 default_server ssl;
```

w przypadku IPv6:

```
listen          [::]:443 default_server ssl;
```

w kolejnych vhostach wystarczy wpisać:

```
listen          443;
```

Oczywiście każdy vhost może mieć osobny certyfikat, np:

```
server {
...
 listen          443;
 ssl_certificate      server1.crt;
 ssl_certificate_key  server1.key;
...
}
```

```
...
server {
...
 listen          443;
 ssl_certificate      server2.crt;
 ssl_certificate_key  server2.key;
...
}
```

Vhosty

Domyślnie działa nam główny vhost - czyli domyślny. Aby utworzyć kolejnego trzeba dodać do konfiguracji kolejną sekcje **server** w sekcji **http**. Aby mieć porządek w pliku konfiguracji polecam dodanie na końcu sekcji **http** (przed **}**) opcji:

```
include         /etc/nginx/vhosts.d/*.conf;
```

która będzie dodawała konfiguracje z plików (z rozszerzeniem **conf**) z katalogu **/etc/nginx/vhosts.d/**. Tego katalogu nie ma więc musimy go stworzyć:

```
$ mkdir /etc/nginx/vhosts.d/
$ chmod 700 /etc/nginx/vhosts.d/
```

Przykładowy konfig vhosta:

```
server {
        listen          [::]:80;
        server_name     nasz.vhost.ltd;
        access_log      /home/users/kamil/www/nasz.vhost.ltd/access.log
main;
        error_log       /home/users/kamil/www/nasz.vhost.ltd/error.log
error;

        location / {
            autoindex on;
                        root    /home/users/kamil/www/nasz.vhost.ltd/htdocs;
                        index   index.html index.htm;
        }

}
```

Z czasem on się zmieni w zależności co będziemy chcieli osiągnąć w danym vhoście.

CGI

CGI będzie nam potrzebne, aby uruchomić np Mailman'a. Ale żeby to zrobić musimy sobie skompilować program do obsługi CGI. A więc zatem instalujemy kompilator:

```
poldek:/all-avail> install gcc gcc-c++ gcc-objc++ gcc-objc libstdc++-
libstdc++-devel git-core autoconf automake fcgi fcgi-devel
```

Ściągamy i kompilujemy **FcgiWrap**:

```
$ mkdir ~/install
$ cd ~/install
$ git clone git://github.com/gnosek/fcgiwrap.git
$ cd fcgiwrap
$ ./configure
$ make
```

Jeśli podczas make będziecie mieć taki błąd:

```
fcgiwrap.c:30:24: fatal error: fcgi_stdio.h: No such file or directory
```

edytujemy plik **fcgiwrap.c** i zmieniamy w nim **linię nr 30** z:

```
#include <fcgi_stdio.h>
```

na:

```
#include <fastcgi/fcgi_stdio.h>
```

Po skompilowaniu kopiujemy binarkę do **/usr/local/bin/**

```
$ cp ~/install/fcgiwrap/fcgiwrap /usr/local/bin/
```

Tworzymy plik **/etc/init.d/fcgiwrap** i zapisujemy w nim:

```
#!/usr/bin/perl

use strict;
use warnings FATAL => qw( all );

use IO::Socket::UNIX;

my $bin_path = '/usr/local/bin/fcgiwrap';
my $socket_path = $ARGV[0] || '/tmp/cgi.sock';
my $num_children = $ARGV[1] || 1;

close STDIN;

unlink $socket_path;
my $socket = IO::Socket::UNIX->new(
    Local => $socket_path,
    Listen => 100,
);

die "Cannot create socket at $socket_path: $!\n" unless $socket;
```

```
for (1 .. $num_children) {
    my $pid = fork;
    die "Cannot fork: $!" unless defined $pid;
    next if $pid;

    exec $bin_path;
    die "Failed to exec $bin_path: $!\n";
}
```

Nadajemy odpowiednie uprawnienia:

```
$ chmod 750 /etc/init.d/fcgiwrap
$ chown root:nginx /etc/init.d/fcgiwrap
```

Uruchamiamy:

```
$ sudo -u nginx /etc/init.d/fcgiwrap
$ chmod 770 /tmp/cgi.sock
```

Dodpisujemy na koniec pliku **/etc/rc.d/rc.local** dwa powyższe polecenia.

Przykładowa konfiguracja vhosta Mailmana:

```
server {
        listen          [::]:80;
        listen          [::]:443;

        server_name     mailman.host.ltd lists.host.ltd;
        access_log      /var/log/httpd/mailman.orchia.pl_access.log main;
        error_log       /var/log/httpd/mailman.orchia.pl_error.log error;

        root    /usr/lib/mailman/cgi-bin;

        location = / {
                rewrite ^ /mailman/listinfo permanent;
        }

        location / {
                rewrite ^ /mailman$uri;
        }

        location ~ ^/mailman(/[^/]*)(/.*)?$ {
                fastcgi_split_path_info (^/mailman/[^/]*)(.*)$;
                include fastcgi.params;
                fastcgi_param GATEWAY_INTERFACE CGI/1.1;
                fastcgi_param SCRIPT_FILENAME $document_root$1;
                fastcgi_param PATH_INFO $fastcgi_path_info;
                fastcgi_param PATH_TRANSLATED $document_root$2;
                fastcgi_pass unix:/tmp/cgi.sock;
        }
```

```
        location /icons {
                alias /usr/lib/mailman/icons;
        }

        location /mailman/pipermail {
                alias /var/lib/mailman/archives/public;
                autoindex on;
        }

}
```

PHP

Instalujemy PHP'a:

```
poldek:/all-avail> install php-common php-cgi php-cli php-fpm
```

Ustawiamy zmienne w pliku **/etc/php/php-fpm.conf**

```
pid = run/php/fpm.pid
daemonize = yes
listen = 127.0.0.1:9000
listen.allowed_clients = 127.0.0.1
user = nginx
group = nginx
pm = dynamic
pm.min_spare_servers = 5
pm.max_spare_servers = 35
```

Tworzymy strukturę katalogów pod vhosta:

```
$ mkdir -p /home/users/user/www/moj.vhost.ltd/htdocs
$ mkdir -p /home/users/user/www/moj.vhost.ltd/tmp
$ chmod -R 750 /home/users/user/www/moj.vhost.ltd/htdocs
$ chmod -R 770 /home/users/user/www/moj.vhost.ltd/tmp
$ chown -R user:http /home/users/user/www/moj.vhost.ltd
```

Tworzymy plik **/etc/nginx/vhosts.d/moj.vhost.ltd** i zapisujemy w nim konfigurację:

```
server {
        listen          [::]:80;
        server_name     cp.kamilm.net;
        access_log      /home/users/user/www/moj.vhost.ltd/access.log main;
        error_log       /home/users/user/www/moj.vhost.ltd/error.log error;

        location / {
                        root    /home/users/user/www/moj.vhost.ltd/htdocs;
                        index   index.html index.htm index.php;
        }
```

```
        location ~ \.php$ {
                include         /etc/nginx/fastcgi/moj.vhost.ltd.param;
                fastcgi_pass    127.0.0.1:9000;
                fastcgi_index   index.php;
                fastcgi_param   SCRIPT_FILENAME
/home/users/user/www/moj.vhost.ltd/htdocs$fastcgi_script_name;
        }

}
```

Tworzymy katalog na konfigurację paramterów PHP'a dla vhostów:

```
$ mkdir /etc/nginx/fastcgi
$ chmod 700 /etc/nginx/fastcgi
```

Tworzymy plik **/etc/nginx/fastcgi/moj.vhost.ltd.param** i zapisujemy w nim:

```
fastcgi_param  QUERY_STRING       $query_string;
fastcgi_param  REQUEST_METHOD     $request_method;
fastcgi_param  CONTENT_TYPE       $content_type;
fastcgi_param  CONTENT_LENGTH     $content_length;

fastcgi_param  SCRIPT_NAME        $fastcgi_script_name;
fastcgi_param  REQUEST_URI        $request_uri;
fastcgi_param  DOCUMENT_URI       $document_uri;
fastcgi_param  DOCUMENT_ROOT      $document_root;
fastcgi_param  SERVER_PROTOCOL    $server_protocol;

fastcgi_param  GATEWAY_INTERFACE  CGI/1.1;
fastcgi_param  SERVER_SOFTWARE    nginx/$nginx_version;

fastcgi_param  REMOTE_ADDR        $remote_addr;
fastcgi_param  REMOTE_PORT        $remote_port;
fastcgi_param  SERVER_ADDR        $server_addr;
fastcgi_param  SERVER_PORT        $server_port;
fastcgi_param  SERVER_NAME        $server_name;

# PHP only, required if PHP was built with --enable-force-cgi-redirect
fastcgi_param  REDIRECT_STATUS    200;

fastcgi_param  PHP_VALUE
"open_basedir=/home/users/user/www/moj.vhost.ltd/htdocs:/home/users/user/www
/moj.vhost.ltd/tmp:/usr/share/php/:/usr/share/pear/
upload_tmp_dir=/home/users/user/www/moj.vhost.ltd/tmp
session.save_path=/home/users/user/www/moj.vhost.ltd/tmp
sendmail_path='/usr/sbin/sendmail -f user -t -i'";
```

Robimy reload Nginx'a:

```
$ /etc/init.d/nginx-light reload
```

Gdy normalne strony działają, a w PHP nie chcą - pisze np: **Bad Gateway** to najpierw sprawdźmy czy konfiguracja PHP'a jest dobra. Sprawdzić możemy prostym przykładem:

```
$ echo "<?php echo \"dupa\"; " | php
```

PERL

czekam aż ktoś kopnie nowszą wersję perla do repo wtedy dokończe z wraperem perla.

Statystyki

W konfiguracji Nginx'a danego vhosta musimy mieć zapis:

```
               location /nginx_status {
                       stub_status     on;
                       access_log      off;
                       allow   127.0.0.1;
                       deny    all;
               }
```

Czasem bywa tak, że danego vhosta możemy odpytywać nie z localhost'a tylko z wew/zew IP - należy go dopisać do **allow**. Wystarczy przeładować Nginx'a i możemy się odwoływać do http://vhost.ltd/nginx_status - gdy to zadziała możemy pobierać dane i wciągać je do naszego systemu statystyk. Np w MRTG:

Tworzymy plik nginx_vhost.ltd.conf:

```
#
# Nginx stat
#
# connections

WorkDir: /home/users/vftp/www/stats.sun.orchia.pl/htdocs

Target[nginx_con]: `/etc/mrtg/conf.d/mrtg-nginx con`
Options[nginx_con]: growright, integer, nobanner, nopercent, transparent
Title[nginx_con]: Nginx Connections
MaxBytes[nginx_con]: 2048
YLegend[nginx_con]: Connections
ShortLegend[nginx_con]: con
LegendI[nginx_con]:
LegendO[nginx_con]: con:
Legend1[nginx_con]: Connections
Legend2[nginx_con]: Connections
PageTop[nginx_con]: <h1>Nginx Connections</h1>
# requests
Target[nginx_req]: `/etc/mrtg/conf.d/mrtg-nginx req`
Options[nginx_req]: gauge, growright, nobanner, nopercent, transparent
```

```
Title[nginx_req]: Nginx Requests
MaxBytes[nginx_req]: 2048
YLegend[nginx_req]: request/s
ShortLegend[nginx_req]: req/s
Legend1[nginx_req]: Requests per Second
Legend2[nginx_req]: Requests per Second
LegendI[nginx_req]:
LegendO[nginx_req]: Requests:
PageTop[nginx_req]: <h1>Nginx Requests</h1>
```

Do pliku **/etc/mrtg/conf.d/mrtg-nginx** zapisujemy:

```
#!/usr/bin/perl
# $Revision: 2 $
# $Date: 2008-09-12 15:11:40 +0300 (Fri, 12 Sep 2008) $

my %opt = (
# http link to nginx stub_status, be sure turn on stub_status in nginx conf
    nginx_status   => 'http://localhost:80/nginx_status',
# path for program what may dump web page, normaly lynx -dump
#    lynx            => 'lynx -dump',
    lynx            => 'wget -q -Y off -O -',
);

$opt{var} = $ARGV[0] if $ARGV[0];
$opt{nginx_status} = $ARGV[1] if $ARGV[1] and $ARGV[1]=~/^http:\/\/\w+/;
$opt{var} ||= '';

my $do = `$opt{lynx} $opt{nginx_status}`;

if ($opt{var} eq 'req') {
    $do=~/^Active connections:\s*(\d+)\s*$/ms or warn "Error! Can't find
data!\nIN :\n$do";
    $opt{d2} = $opt{d1} = $1;
}
elsif ($opt{var} eq 'con') {
    $do=~/^\s*(\d+)\s+(\d+)\s+(\d+)\s*$/ms or warn "Error! Can't find
data!\nIN :\n$do";
    $opt{d2} = $opt{d1} = $3;
}
#elsif { $do=~/^Reading:\s+(\d+).*Writing:\s+(\d+).*Waiting:\s+(\d+)/; }
else {
    $opt{var} = 'ERROR';
    $opt{d2} = $opt{d1} = 0;
    warn "Error! Please read the help and set (req|con)\n";
}

print "$opt{d1}\n";
print "$opt{d2}\n";
#print "$opt{up}\n" if $opt{up};
```

```
print "Nginx $opt{var}\n";
```

Inne

HTTP Auth:

```
server {
...
        location / {
...
                        auth_basic            "Restricted";
                        auth_basic_user_file
/home/users/user/www/moj.vhost.ltd/.htpasswd;
        }
    }
```

Plik **/home/users/user/www/moj.vhost.ltd/.htpasswd** możemy wygenerować za pomocą skryptu, który musimy pobrać:

```
$ wget http://trac.edgewall.org/browser/trunk/contrib/htpasswd.py?format=txt
$ mv ./htpasswd.py?format=txt ./htpasswd.py
$ chmod 755 ./htpasswd.py
```

Aby plik działał musimy mieć zainstalowany pakier **python-modules**:

```
poldek:/all-avail> install python-modules
```

Uruchamiamy wg składni:

```
$ ./htpasswd.py -c -b /home/users/user/www/moj.vhost.ltd/.htpasswd admin
tajne_haslo
```

HTTP → HTTPS:

```
server {
...
                listen 80;
                rewrite          ^(.*)          https://$server_name$1
permanent;
...
}

server {
...
                listen 443;
                fastcgi_param HTTPS on;
...
}
```

Zend:

```
server {
...
        location / {
...
                if (!-f $request_filename) {
                   rewrite  ^(.*)$  /index.php  last;
                   break;
                }
        }
```