Существуют различные схемы построения веб-серверов для передачи данных по протоколу HTTP. Среди них достойное место по производительности занимают схемы с использованием «Nginx» в качестве внешнего (кэширующего, front-end) сервера. «Nginx» разработан для отдачи статических данных, при этом, он показывает высокое быстродействие и нагрузочную способность (см. Nginx vs Cherokee vs Apache vs Lighttpd), генерировать же динамическое содержимое он не способен. Поэтому, он часто применяется в связке с внутренним (back-end) сервером для обработки динамических данных которые потом отдаются «Nginx» как статические без участия внутреннего сервера. В качестве внутреннего сервера может применяться «Apache2» или, что и рассматривается в данной статье, «PHP-FPM».
В данной статье рассматривается установка и настройка связки Nginx и PHP-FPM на локальной ЭВМ, если требуется работа на выделенном сервере, то следует обратится к более серьезным инструкциям или/и непосредственной помощи специалистов
Данная статья написана любителем, никто из профессионалов её не проверял, она может содержать ошибки и вредные советы и потому нацелена на тех, кто хочет поиграться

Установка

Сервер «Nginx» поставляется в одноименном пакете «nginx» и его установка производится, например, командой в терминале

```
sudo apt-get install nginx nginx-extras
```

Установку же «PHP-FPM» можно произвести, например, командой

```
sudo apt-get install php5-cli php5-common php5-mysql php5-gd php5-fpm php5-cgi php5-fpm php-pear php5-mcrypt
```

Настройка

Настройка состоит из двух этапов — настройки «Nginx» и «PHP-FPM». Для начала необходимо остановить процессы (демоны) «Nginx» и «PHP-FPM», например, командами

```
sudo service nginx stop
sudo service php5-fpm stop
```

Настройка PHP-FPM

Прежде всего, следует открыть файл «/etc/php5/fpm/php.ini» для редактирования, например, командой

```
sudo nano /etc/php5/fpm/php.ini
```

после чего, найти строчку содержащую «cgi.fix_pathinfo», которая по-умолчанию выглядит так (закомментирована)

```
;cgi.fix_pathinfo = 1
```

и привести её к виду

```
cgi.fix_pathinfo = 0
```

Это призвано устранить опасность неправильно трактования (и возникающей уязвимости) запросов вида «/image.gif/foo.php» (см. Don’t trust the tutorials: check your configuration!, Nginx 0day exploit for nginx + fastcgi PHP).
Если планируется загрузка больших файлов (важно для ownCloud версий < 8, в новой версии 8 и выше имеется отдельный файл для этих настроек), то можно увеличить максимальный объем загружаемых данных, например, до 200 МБ

```
post_max_size = 200M
```

и ниже

```
upload_max_filesize = 200M
```

Затем сохранить изменения в файле.
Далее, необходимо отрыть для редактирования файл «/etc/php5/fpm/pool.d/www.conf», например, командой

```
sudo nano /etc/php5/fpm/pool.d/www.conf
```

найти строчку с параметров «security.limit_extensions» и привести её к виду

```
security.limit_extensions = .php .php3 .php4 .php5
```

Эта настройка ограничит выполнение файлов по расширению имени. В этом же файле найти строчку с параметром «listen» и привести её к виду

```
listen = /var/run/php5-fpm.sock
```

Это определит файл для связи «Nginx» с «PHP-FPM» (сокет). В целях безопасности запрещаем какой-попало программе писать в сокет путём указания прав доступа к сокету. Находим строчки с описанием параметров «listen.owner», «listen.group» и «listen.mode» (по умолчанию они закомментированы) и приводим их к виду

```
listen.owner = www-data
listen.group = www-data
listen.mode = 0660
```

Следует сохранить изменения в файле и перезапустить «PHP-FPM», например, командой
```
sudo service php5-fpm restart
```
Можно убедится в том, что права доступа к сокету установлены верно:
```
ls -la /var/run/php5-fpm.sock
```
Права доступа должны быть «srw-rw—-», владелец «www-data» (группа «www-data»), например,
```
srw-rw---- 1 www-data www-data 0 May  2 16:36 /var/run/php5-fpm.sock
```

Настройка Nginx

Основные настройки «Nginx» хранятся в файле «/etc/nginx/nginx.conf». Настройки базового сайта хранятся в файле «/etc/nginx/sites-available/default». Базовый конфигурационный файл сайта принято помещать в папку «/etc/nginx/sites-available/» и затем включить его путём добавления символической ссылки на этот файл в папке «/etc/nginx/sites-enabled/». Например, создадим конфигурационный файл для сайта с доменным именем (для примера выбрано «example.com») в названии для удобства
```
sudo touch /etc/nginx/sites-available/example.com
sudo ln -s /etc/nginx/sites-available/example.com /etc/nginx/sites-enabled/
```
и откроем его для редактирования
```
sudo nano /etc/nginx/sites-available/example.com
```

При редактировании данного файла необходимо учитывать синтаксис конфигурации «Nginx», готовые советы по настройке связки «Nginx + PHP-FPM» и следовать некоторым правилам/рекомендация для достижения эффективной работы сервера.

Описывать конфигурацию сайта в одном файле не очень удобно, для увеличения читабельности конфигурационного файла и гибкости настройки можно воспользоваться директивой «include» позволяющей указать «Nginx», что следует загрузить другой конфигурационный файл и затем продолжить чтение текущего.

Создадим в папке «/etc/nginx/» каталог «common», где будут хранится общие настройки для сайта, которые затем будут подгружаться из основного конфигурационного файла «/etc/nginx/sites-available/example.com» с помощью директивы «include»
```
sudo mkdir /etc/nginx/common
```

Некоторые запросы «Nginx» будет перенаправлять к «PHP-FPM», который в данном случае называется сервером выгрузки данных (upstream). Укажем как следует это делать. Создадим файл конфигурации с описанием серверов выгрузки данных
```
sudo touch /etc/nginx/common/upstream
```
и откроем его для редактирования
```
sudo nano /etc/nginx/common/upstream
```
и добавим в него строчки
```
upstream php-fpm
{
        # PHP5-FPM сервер
        server unix:/var/run/php5-fpm.sock;
}
```
где «php-fpm» – название для сервера выгрузки данных, для удобства.
Редактируем файл «/etc/nginx/sites-available/example.com». Добавляем строчку
```
include common/upstream;
```
для загрузки созданого выше конфигурационного файла. Как можно видеть – допускается указание относительного пути к файлу.
Далее описываем перенаправление от HTTP к HTTPS, если, конечно, это планируется. В таком случае необходимо наличие сертификатов для HTTPS
```
server
{
        listen 80;
        server_name example.com www.example.com;
        return 301  https://$server_name$request_uri;
}
```
иначе, можно опустить эти строчки.
Начинаем описывать конфигурацию сайта
```
server
{
```
Сетевой порт для приема соединений: 80 — обычный HTTP; 443 — HTTPS (см. выше)
```
        # Порты
        listen  80;
        listen  443     ssl;    # использовать шифрование для этого порта
```
Корневая директория сайта работающего на данном сервере

```
        root                    /var/www;
```
Возможные имена индексных файлов (их «Nginx» пытается открыть если он получил запрос вида «example.com/», вместо явного «example.com/index.html»)
```
        index                   index.php index.html index.htm;
```
Имя сервера – обычно доменное имя Вашего сервера
```
        server_name             example.com www.example.com;
```

Безопасность

Опишем настройки безопасности в отдельном файле
```
sudo touch /etc/nginx/common/security
sudo nano /etc/nginx/common/security
```
И укажем в нём
```
add_header              X-Frame-Options        "SAMEORIGIN";
add_header              X-Content-Type-Options "nosniff";
```
Сохраним и закроем файл, а затем подключим его строкой
```
        include common/security;
```

Базовые ограничения

Выше была написана строчка для подключение файла «/etc/nginx/common/deny»
```
include common/deny;
```
рассмотрим его содержание. В нём идет запрет доступа к некоторым стандартным файлам. Создадим этот файл
```
sudo touch /etc/nginx/common/deny
sudo nano /etc/nginx/common/deny
```
с содержанием
```
# Запрет доступа к .htaccess и .htpasswd файлам
location ~* "/\.(htaccess|htpasswd)$"
{
        deny all;               # запретить все для всех
        return 404;             # вернуть код ошибки
}
```
Следует быть бдительным, неверно указанный шаблоны для запрета доступа (не только здесь но и в примерах выше) могут сильно навредить. Например, клиент ownCloud может начать удалять файлы которые не сможет загрузить на сервер из-за неправильного запрета где-то в конфигурационном файле
Следует переписать все файлы «.htaccess» в директивы «Nginx». Найти эти файлы среди файлов сайта можно, например, командой
```
sudo find /var/www/ -name .htaccess
```

Вызов PHP-FPM

В примерах выше использовался файл «/etc/nginx/common/php-fpm» — в нём идет перенаправление обработки php-скриптов внутреннему серверу «PHP-FPM»
В файле «php.ini» должно быть установлено «cgi.fix_pathinfo = 0;» Также, в файле «/etc/php5/fpm/pool.d/www.conf» должно присутствовать ограничение на разширение имени исполняемых скриптов - «security.limit_extensions = .php .php3 .php4 .php5»
Создаём этот файл
```
sudo touch /etc/nginx/common/php-fpm
sudo nano /etc/nginx/common/php-fpm
```
с содержанием
```
# Настройки порта или сокета PHP-FPM производятся в файле
"/etc/php5/fpm/pool.d/www.conf"
fastcgi_pass   php-fpm;
# Порядок важен - строчка "include fastcgi_params" должна быть первой
include fastcgi_params;
fastcgi_split_path_info                       ^(.+?\.php)(/.*)?$;
# Вместо переменной "$document_root" можно указать адрес к корневому каталогу
сервера и это желательно (см. http://wiki.nginx.org/Pitfalls)
fastcgi_param  SCRIPT_FILENAME        $document_root$fastcgi_script_name;
fastcgi_param  PATH_TRANSLATED        $document_root$fastcgi_script_name;
# См. http://trac.nginx.org/nginx/ticket/321
```

```
set             $path_info              $fastcgi_path_info;
fastcgi_param   PATH_INFO               $path_info;
# Additional variables
fastcgi_param   SERVER_ADMIN            email@example.com;
fastcgi_param   SERVER_SIGNATURE        nginx/$nginx_version;
fastcgi_index   index.php;
```

Кеширование

Выше, в примерах, был упомянут файл «/etc/nginx/common/cache»
Сайт работает значительно лучше когда часть контента сохранена на стороне клиента с прошлого посещения сайта. Не все файлы можно кешировать. Поэтому описание кеширования производится в самом конце (т.е. эти настройки будут иметь наименьший приоритет и есть шанс что это не повлияет на правильную работу сайта). Создадим файл с параметрами для кеширования

```
sudo touch /etc/nginx/common/cache
sudo nano /etc/nginx/common/cache
```

где укажем

```
location ~*
".+\.(?:ogg|ogv|svg|svgz|eot|otf|woff|mp4|ttf|rss|css|swf|js|atom|jpe?g|gif|p
ng|ico|zip|tgz|gz|rar|bz2|doc|xls|exe|ppt|tar|mid|midi|wav|bmp|rtf)$"
        {
                access_log      off;
                log_not_found   off;
                expires         max;
        }
```

Окончание

Закрываем фигурные скобки директивы «server» в «/etc/nginx/sites-available/example.com»

```
}
```

На этом правка файла «/etc/nginx/sites-available/example.com» завершена. Убедитесь в том, что все фигурные скобки «{ }» закрыты корректно и части файла верно вложены друг в друга («location» внутри «server» и т.п.).
Сохраняем все изменённые файлы.
Теперь можно перезапустить демоны

```
sudo service nginx restart
sudo service php5-fpm restart
```

Проверка

Проверить свой сайт можно создав файл «info.php» с содержанием

```
<?php
phpinfo();
?>
```

затем скопировав его, например, в «/var/www/wordpress/wp-content/uploads/», затем открыв адрес в браузере «http://example.com/wordpress/wp-content/uploads/info.php», если он выполнится вместо того чтобы просто сохранится — то что-то настроено неправильно, в этой директории php файлы выполняться не должны ни в коем случае (она доступна для загрузки)