https://debian.pro/1799

Про Debian

Делаем файловый сервер из голого nginx. Webdav.

Иногда возникает необходимость заливать файлики на второй сервер и отдавать их оттуда же по http(s). Ну навскидку — отдельные серверы с картинками для вашего проекта. Или с архивами. Да с чем угодно, мне-то какая разница =)

Поднимать ради этого ftp и городить работы с ftp в коде сайта? гм… SFTP… можно, но где взять разработчика, который умеет sftp из php? Джангисты и рельсовики все умеют, ладно. А если nodejs?
А вот curl/libcurl есть везде, PUT/POST запросы можно сделать из любого ЯП/фреймворка. Поэтому приходят 2 логичных решения — либо написать php-код на принимающей стороне, который будет принимать файлы, либо сделать всё это на голом nginx-е. Окей, решили вторым способом, пришли на тостер, спросили… И получили какой-то бред в ответе — https://toster.ru/q/46578
upload-module? big-upload-module? clientbodyonthefly ещё куда ни шло, это всё же штатная фича..

Да к черту. Открываем доку (окей, в моём случае — задаём вопрос в воздух и получаем ответ от соседа-админа) по nginx и листаем до пункта «http webdav module». Всё, можно заливать файлы (ладно-ладно, конфиг я вам покажу). И решаем сразу все проблемы: нужен только дефолтный nginx из репозиториев debian/ubuntu, размер файла ограничен только стабильностью соединения (13 гиговый файл? да легко!), конфиг писать легко (копипастинг работает, да), авторизация рулится через привычный htpasswd.. В общем — быть. Погнали

Допустим nginx у нас уже есть и что-то по http куда-то отвечает. Нам нужно на images.example.com настроить upload файлов через webdav (и научиться заливать файлы курлом) с авторизацией и научиться показывать эти файлы по http без авторизации. Создаём каталог, где будем всё это хранить, например:

```
root@server:~# mkdir /home/user/data/www/images.example.com
```

Nginx из коробки работает от пользователя www-data (я про деб, в других дистрах это может быть www), от него же он будет и файлы писать, так что пофиксим права:

```
root@server:~# chown -R www-data:www-data
/home/user/data/www/images.example.com
```

Теперь пишем конфиг (например, в /etc/nginx/sites-available/00-images.example.com):

```
server {
    listen 80;
    server_name images.example.com;
    # описываем upload-секцию:
    location /upload{
        # максимальный размер файла, который можно залить.
        client_max_body_size 15g;
        # каталог, куда заливать
        root /home/user/data/www/images.example.com;
        # chmod для залитых файлов - здесь 777, чтобы user тоже мог удалять
файлы
        dav_access user:rw group:rw all:rw;
        # разрешаем методы webdav-а. Для примера я перечислил все, для
аплоада файлов хватит PUT и MKCOL
        dav_methods PUT DELETE MKCOL COPY MOVE;
        # nginx будет создавать весь путь при аплоаде файлов (можно будет не
```

```
создавать предварительно вложенные каталоги)
        create_full_put_path on;
        # включаем autoindex в каталоге upload (чтобы на самом
images.example.com листинг не включать).
        autoindex on;
        autoindex_exact_size off;
        autoindex_localtime on;
        charset utf-8;
        # включаем авторизацию в /upload:
        auth_basic "Upload directory";
        auth_basic_user_file /etc/nginx/htpasswd;
        }
    # теперь описываем раздачу файлов по http из "корня" сайта:
    location / {
        root /home/user/data/www/images.example.com;
    }
}
```

Создадим htpasswd-файл:

```
root@server:~# touch /etc/nginx/htpasswd; chmod 600 /etc/nginx/htpasswd
```

Сгенерим пользователя (если у вас нет утилиты htpasswd, то поставьте пакет apache2-utils — apache2 этот пакет не ставит, если что =)) — команду повторите нужное количество раз, заменяя username, при запуске она запросит пароль для пользователя:

```
root@server:~# htpasswd -nm username >> /etc/nginx/htpasswd
```

Рестартим nginx, если ещё нет и приступаем к заливанию файлов. Я покажу это на примере консольного curl, с libcurl, думаю, разберетесь сами, если уже дочитали до сюда.
Например, у нас есть файл /tmp/bigfile. Нам его нужно будет показывать по адресу http://images.example.com/archives/bigfile.zip
Запускаем curl с хитрыми параметрами:

```
user@laptop:~$ curl -T /tmp/bigfile
http://username:password@images.example.com/upload/archives/bigfile.zip
```

Имейте в виду, что пока файл льётся, он льётся в каталог /var/lib/nginx/body/. Когда upload закончится, файл будет перемещен в docroot через mv — так что если у вас /var и каталог для upload на разных файловых системах, то операция будет не атомарной. Если же на одной файловой системе — то файл появится в docroot мнгновенно и целиком (то есть не будет такого, что вы заливаете файл, а в это время его кусок могут скачать другие).

В некоторых местах вы можете услышать мнение, что передавать логин/пароль так, как это сделал я выше небезопасно и нужно обязательно передавать basic-авторизацию через http-заголовок Authorization, потому что так безопаснее. Так вот — не слушайте потом больше никогда этого человека. Заголовок Authorization это всего-лишь base64 (хоть и немного нестандартный) и декодится по первой же ссылке в гугле — https://webnet77.net/cgi-bin/helpers/base-64.pl
Намного лучше будет настроить https (https://debian.pro/581) и настроить ограничение по ip (https://debian.pro/726). Впрочем, если вам удобнее передавать через заголовок — welcome, но не тешьте себя надеждой, что это позволит избежать раскрытие пароля, если к исходникам заливалки получат доступ.

Да, ещё с этим модулем nginx’а не будут работать webdav-клиенты — уж очень он простой (в нём даже нет вебдавного листинга файлов/каталогов). Если вам нужен настоящий webdav — то советую посмотреть в сторону установки OwnCloud.

30.04.2015 byinkvizitor68sl|Администрирование