Nginx+php-cgi: настройка автоматического запуска при старте Windows

Введение

На одном из проектов мы используем windows server 2008 в качестве операционной системы, так как работа проекта связанна с конвертацией графических форматов, для чего мы используем Adobe Illustrator. К сожалению Inkskape/ImageMagick/GraphicsMagick, которые отлично работают на linux, нам не подходят из-за ряда проблем с конвертацией, так что приходится страдать. И вот появилась необходимость в использовании php. Вариантов как бы немного:

1. apache + mod_php/php-cgi
2. iis + php-cgi
3. nginx + php-cgi

К сожалению, официальных сборок apache под windows больше не выпускает, последняя официальная сборка 2.2.25 от 2013 года. Но правда есть т.н. 3rd party сборки:

1. apachelounge
2. XAMPP
3. apachehaus

С IIS я работал очень мало, и честно говоря, особого желания нет. А вот опыт работы с nginx + php-cgi/fpm более чем достаточный. Так что остановился на этом варианте. Особенно учитывая тот факт, что в ближайшем будущем скорее всего будем мигрировать с Windows/AI на Linux/Inkskape/ImageMagick/GraphicsMagick/etc.

Казалось бы простая задача, установить и настроить nginx + php-cgi, которая в linux занимает 10-15 минут, на windows платформе заняла у меня на порядок больше времени из-за своей особенности и специфики.

Установка и настройка сервера php-cgi

Идем на страницу php.net и выбираем нужную версию php. PHP поставляется в двух вариантах

1. Thread Safe version (TS)
2. Non Thread Safe version (NTS)

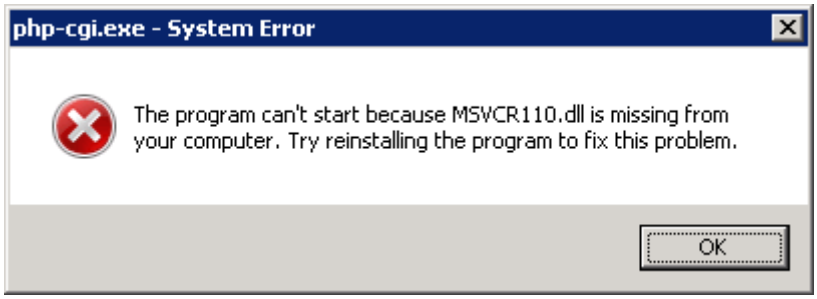
Если кратко, то TS версию следует использовать, если у вас php используется в качестве модуля apache. NTS версия должна быть использована, если вы используете php как cgi. Более подробно о различиях NTS от TS версии можно почитать на stackoverflow

Итак скачиваем архив, в моем случае это был файл php-5.6.21-nts-Win32-VC11-x86.zip и распаковываем в любое удобное для вас место. В рамках данной статьи я буду использовать следующий путь - **`C:\php-5.6.x-nts-vc11-x86\`**. Так же для избежания множества ошибок,

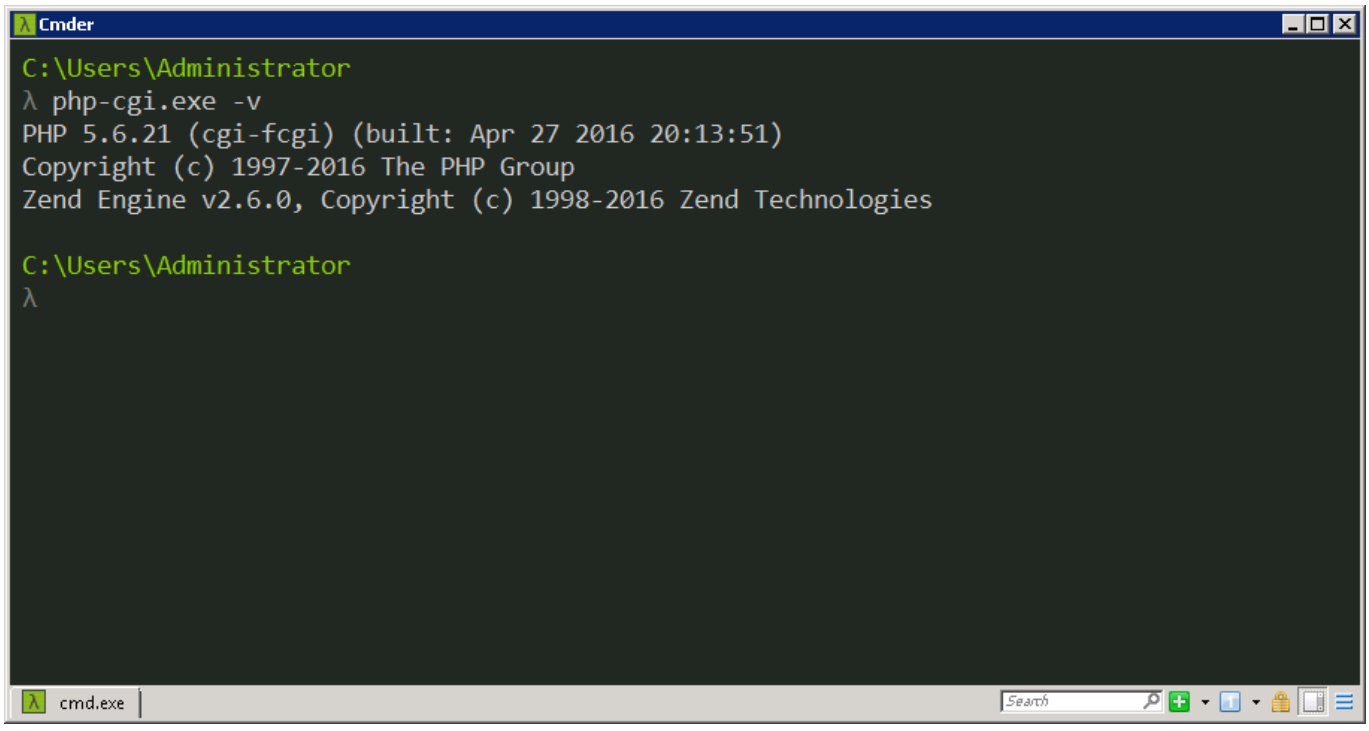
советую добавить данный путь в переменную окружения PATH.

Пользователям Windows стоить обратить внимание на то, что сборка x86_64 на момент написания статьи, а это май 2016 г, является экспериментальной, и не предоставляет работу с 64-bit целыми, а так же не поддерживает работу с большими файлами (Large File Support).

Так же не забываем о том, что для корректной работы php необходимо установить пакет Visual C++ Redistributable for Visual Studio 2012, если у вас такой не стоит в системе, иначе при запуске будем получать ошибку вида



Проверяем что переменная PATH применилась и работает



После распаковки копируем файл php.ini-production в php.ini и правим под свои нужды. Ниже я приведу лишь те строки, которые я менял

```
[PHP]

; Отключаем поддержку пользовательских ini файлов (некоторый аналог .htaccess),
; который позволяет пользователю переопределять некоторые переменные. Например,
memory_limit
user_ini.filename =
```

```
; Задаем путь к файлу с ошибками, для поиска проблем. Папку logs необходимо создать вручную
; до запуска cgi сервера, она не создается автоматически!!!
error_log = C:/php-5.6.x-nts-vc11-x86/logs/php-error.log

; А вот эта опция является обязательной. Здесь мы указываем путь к папке с расширениями - php_*.dll
; Как правило расширения располагаются в папке ext, которая находится в корне архива. Причем, лучше
; задать абсолютный путь, а не относительный, который используется по умолчанию.
extension_dir = "C:/php-5.6.x-nts-vc11-x86/ext"

; В целях безопасности и производительности отключаем данную опцию. При значении 1,
; которое является значением по умолчанию, если файл /image/test.gif/test.php не найден,
; то будет осуществлена попытка доступа к "ближайшему" файлу, в данном случае это /image/test.gif
; и данный файл будет интерпретироваться как php, со всеми вытекающими последствиями.
cgi.fix_pathinfo=0

; Включаем, нужны нам расширения
extension=php_bz2.dll
extension=php_curl.dll
extension=php_gd2.dll
extension=php_mbstring.dll
extension=php_openssl.dll
extension=php_sockets.dll
zend_extension=php_opcache.dll

; Задаем временную зону
date.timezone ='Etc/UTC'

; Базовые настройки для модуля opcache
[opcache]
opcache.error_log=C:/php-5.6.x-nts-vc11-x86/logs/opcache-error.log
opcache.memory_consumption=128
opcache.interned_strings_buffer=8
opcache.max_accelerated_files=4000
opcache.revalidate_freq=60
opcache.fast_shutdown=1
opcache.enable_cli=1
```

В Windows-версии PHP многие расширения уже встроены (builtin) в интерпретатор. Т.е. для загрузки данных расширений директивы extension не используются. Ниже приведен список встроенных расширений: BCMath, Caledar, COM, Ctype, DOM, FTP, LibXML, Iconv, ODBC, PCRE, Session, SimpleXML, SPL, SQLite, WDDX, XML и Zlib.

Если вы все правильно настроили, то команда `php-cgi.exe -m` должна выдать примерно такой список

```
cmder> php-cgi.exe -m
[PHP Modules]
bcmath
bz2
calendar
cgi-fcgi
Core
ctype
curl
date
dom
ereg
filter
ftp
gd
hash
iconv
json
libxml
mbstring
mcrypt
mhash
mysqlnd
odbc
openssl
pcre
PDO
Phar
Reflection
session
SimpleXML
sockets
SPL
standard
tokenizer
wddx
xml
xmlreader
xmlwriter
Zend OPcache
zip
zlib

[Zend Modules]
Zend OPcache
```

Если вы где то допустили ошибку, то в файле php-error.log появятся соответствующие предупреждения/ошибки. Например, ошибка вида

```
[11-May-2016 19:14:11 Etc/UTC] PHP Warning:  PHP Startup: Unable to load
```

```
dynamic library 'C:/php-5.6.x-nts-vc11-x86/ext/php_gd21.dll' - The specified
module could not be found.
 in Unknown on line 0
```

Если ошибок нет, то запускаем наш cgi сервер из командной строки

```
cmder> c:\php-5.6.x-nts-vc11-x86\php-cgi.exe -b127.0.0.1:9090
```

Теперь переходим к настройке nginx.

Установка и настройка nginx

Установка аналогична php, скачиваем архив с официального сайта, распаковываем в нужную нам папку, правим конфигурационный файл под свои нужды и запускаем. В рамках данной статьи я буду использовать следующий путь - **C:\nginx-1.8.x**

Ниже привожу минимально необходимые настройки, которые необходимо внести в файл nginx.conf

```
worker_processes  1;

error_log  logs/error.log;
pid        logs/nginx.pid;



events {
    worker_connections  1024;
}



http {
    include       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"';

    sendfile        on;
    #tcp_nopush     on;

    keepalive_timeout  65;

    gzip  on;
```

```
    server {
        listen       80;
        server_name  _default;

        charset utf-8;
    ; Небольшой плюс, на windows nginx по умолчанию собран с поддержкой debug, т.е.
нет необходимости
    ; запускать отдельный бинарный файл, например на linux это nginx-debug, в
случае отладки.
        access_log  logs/www.example.net_access.log main;
    error_log    logs/www.example.net_error.log;
    root C:/nginx-1.8.x/vhosts/default/;

        location / {
            index  index.html index.htm;
        }
    location = /favicon.ico {
        access_log off;
        }
        location = /robots.txt {
        access_log off;
        }

        error_page  404              /404.html;
        error_page  500 502 503 504  /50x.html;

        location ~ \.php$ {
            fastcgi_pass   127.0.0.1:9090;
            fastcgi_index  index.php;
            fastcgi_param  SCRIPT_FILENAME
$document_root$fastcgi_script_name;
            fastcgi_intercept_errors on;
            include        fastcgi_params;
        }

    }

}
```

После этого можно запустить nginx, для этого выполнив следующую команду

```
cmder> c:\nginx-1.8.x\nginx.exe -p c:\nginx-1.8.x -c conf\nginx.conf
```

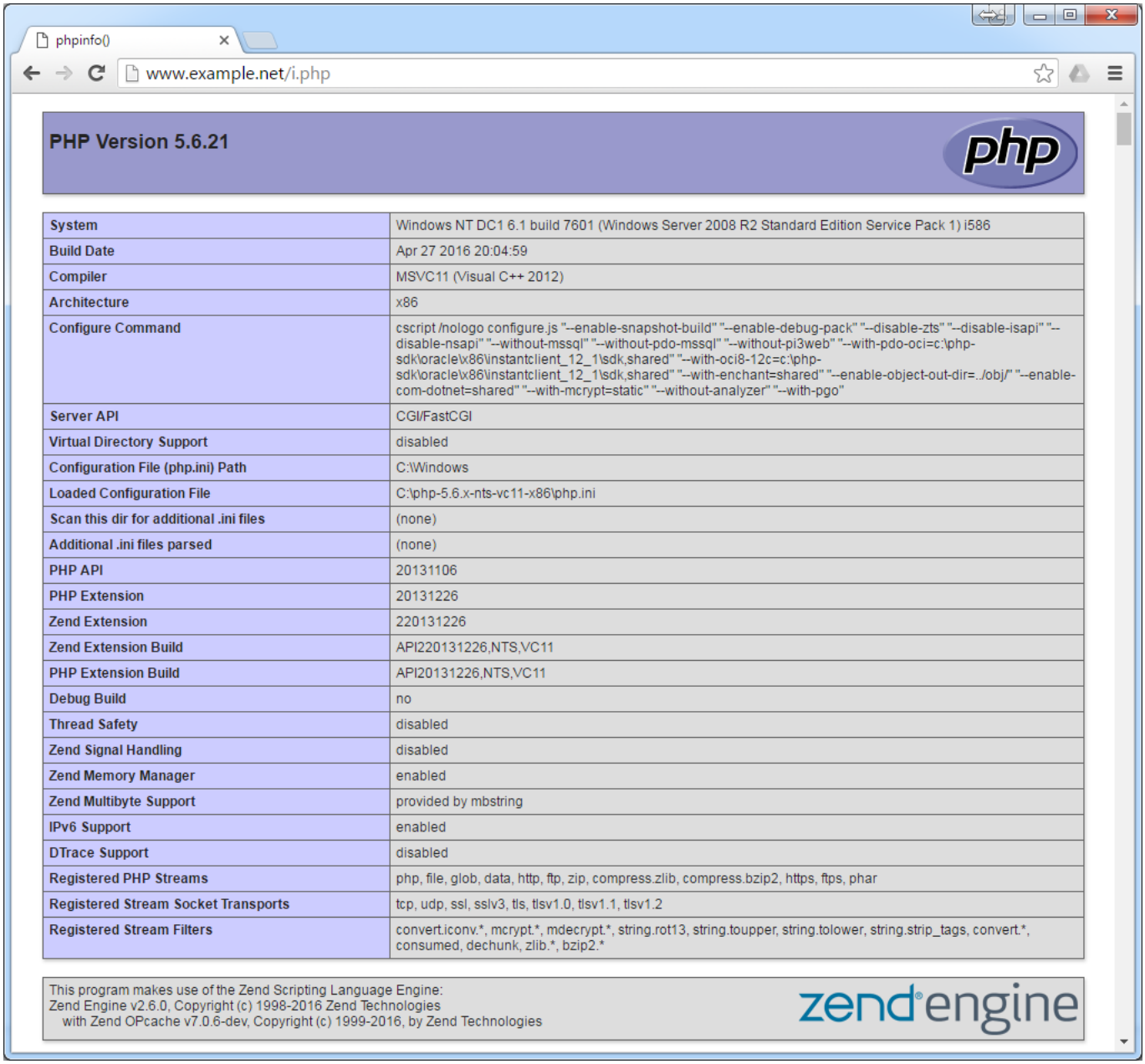
Не забываем открыть 80й порт

```
cmder> netsh advfirewall firewall add rule name='NGINX' dir=in action=allow
protocol=TCP localport=80
OK
```

Проверяем в командной строке или в любом проводнике, если вы не используете cmder

```
cmder> curl -I -i http://www.example.net/
HTTP/1.1 200 OK
Server: nginx/1.8.1
Date: Sat, 14 May 2016 11:39:55 GMT
Content-Type: text/html; charset=utf-8
Content-Length: 612
Last-Modified: Tue, 26 Jan 2016 14:41:53 GMT
Connection: keep-alive
ETag: "56a785b1-264"
Accept-Ranges: bytes
```

Ну а для проверки php, создаем тестовый файл с выводом phpinfo. В результате мы должны увидеть примерно такую страницу



Настройка автоматического запуска при старте

Сама связка у нас настроена и работает, но проблема в том, что как только вы закроете сеанс или перезагрузите сам сервер, то придется заново запускать в ручном режиме, что очень не удобно, а иногда и не приемлемо. К сожалению сам php-cgi.exe и nginx.exe не поддерживают работу в виде службы, можно конечно использовать и сторонние утилиты, но большинство из них платные, или глючные. Немного поискав в google, нашел интересную утилиту от автора Jenkins - winsw (Windows Service Wrapper). Эта небольшая утилита ~58 Кбайт, которая представляет собой один исполняемый файл - winsw-1.18-bin.exe.

Итак, для настройки нам необходимо скопировать winsw-1.18-bin.exe в папку с nginx.exe и переименовать во что то осмысленное (на самом деле переименовывать не обязательно, просто так будет удобней). Я буду использовать **`C:\nginx-1.8.x\winsw-nginx.exe`** для запуска nginx и **`C:\php-5.6.x-nts-vc11-x86\winsw-php.exe`** для запуска php-cgi.exe соответственно. После копирование файла в папки nginx и php создаем там же xml файлы с точно таким же названием как и сам exe - winsw-nginx.xml и winsw-php.xml с таким содержимым

```
<!-- winsw-nginx.xml -->
<service>
  <id>nginx-1.8.x</id>
  <name>nginx-1.8.x</name>
  <description>nginx-1.8.x</description>
  <executable>c:/nginx-1.8.x/nginx.exe</executable>
  <logpath>c:/nginx-1.8.x/wrapper</logpath>
  <logmode>roll</logmode>
  <depend></depend>
  <startargument>-pc:/nginx-1.8.x</startargument>
  <startargument>-cconf/nginx.conf</startargument>
  <stopexecutable>c:/nginx-1.8.x/nginx.exe</stopexecutable>
  <stopargument>-sstop</stopargument>
</service>
```

```
<!-- winsw-php.xml -->
<service>
  <id>php-cgi-56</id>
  <name>php-cgi-56</name>
  <description>php-cgi-56</description>
  <executable>c:\php-5.6.x-nts-vc11-x86\php-cgi.exe</executable>
  <logpath>c:\php-5.6.x-nts-vc11-x86\wrapper</logpath>
  <logmode>roll</logmode>
  <startargument>-b127.0.0.1:9090</startargument>
  <stopexecutable>taskkill /F /T /IM php-cgi.exe</stopexecutable>
</service>
```

Думаю xml говорит сам за себя, по сути мы указываем как и с какими параметрами нам надо запускать и останавливать нашу службу. Так как php-cgi.exe не имеет ключа для остановки, то

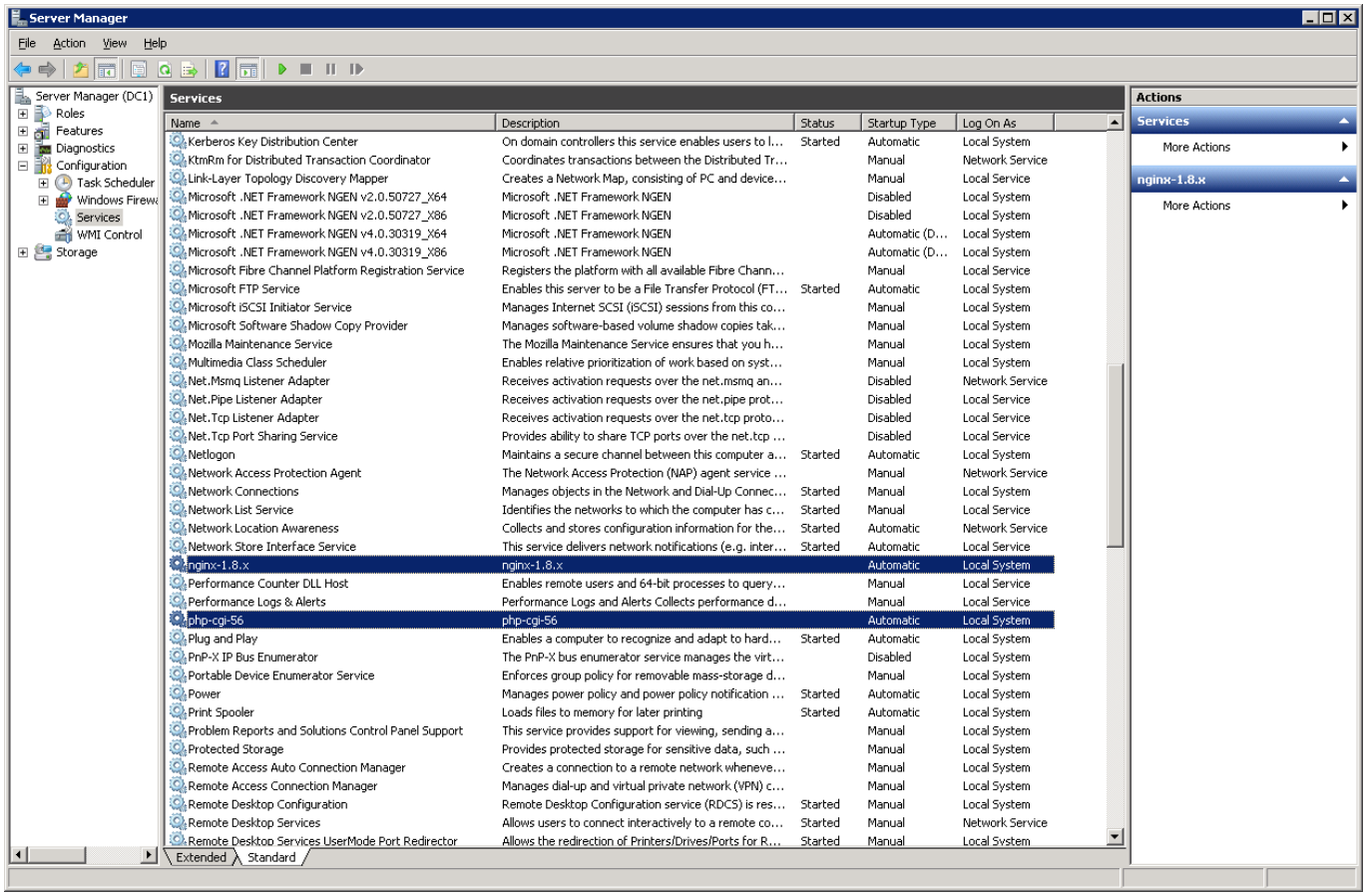
приходится его останавливать при помощи taskkill, вот такая небольшая хитрость. Обратите внимание на то, что папка, указанная в logpath должна быть создана до запуска службы.

После этого необходимо создать сами службы, для этого выполним следующие команды

```
cmder> cd /D c:\nginx-1.8.x\
cmder> winsw-nginx.exe install
```

```
cmder> cd /D c:\php-5.6.x-nts-vc11-x86\
cmder> winsw-php.exe install
```

После чего, в Service Manager на закладке Services вы должны увидеть две новые службы



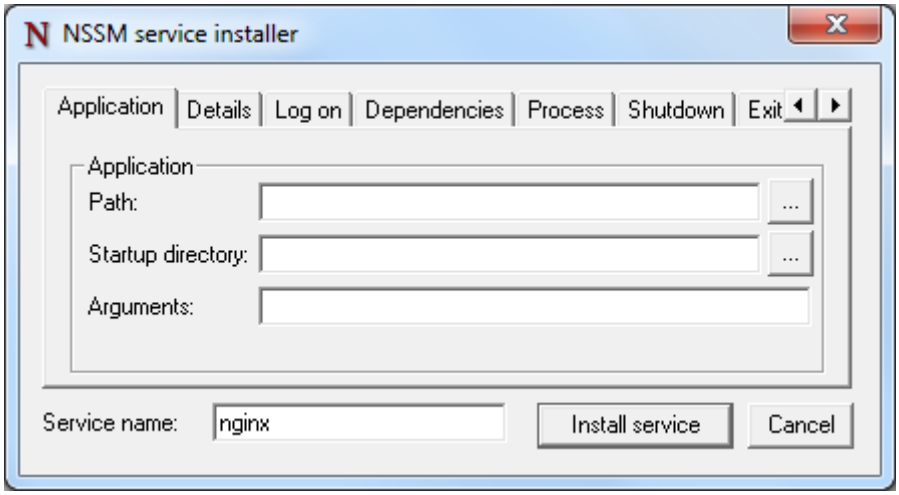
Запускаем соответствующие службы и проверяем работу. Так как Startup type у этих служб по умолчанию будет Automatic, то после перезагрузки сервера они запустятся автоматически.

Так же советую обратить внимание на утилиту nssm, которая имеет очень интересное описание

nssm is a service helper which doesn't suck. srvany and other service helper programs suck because they don't handle failure of the application running as a service. If you use such a program you may see a service listed as started when in fact the application has died. nssm monitors the running service and will restart it if it dies. With nssm you know that if a service says it's running, it really is. Alternatively, if your application is well-behaved you can configure nssm to absolve all responsibility for restarting it and let Windows take care of recovery actions.

Так же представляет из себя один exe файл, и даже имеет графический интерфейс, мелочь, а

приятно.



А нашел я ее случайно, когда начал поиск возможности запуска bitvise ssh клиента при старте системы. К сожалению с помощью winsw мне это сделать не получилось, а вот с помощью nssm все запустилось в течение 5 минут.

Немного деталей о cgi.fix_pathinfo

Я уже описал, поведение данной опции в php.ini, но лучше один раз увидеть, чем 100 раз услышать. Проверяемым работу при значении по умолчанию - cgi.fix_pathinfo=1

```
$ curl -i http://www.example.net/pathinfo.php
HTTP/1.1 200 OK
Server: nginx/1.8.1
Date: Sat, 14 May 2016 13:02:55 GMT
Content-Type: text/html; charset=UTF-8
Transfer-Encoding: chunked
Connection: keep-alive
X-Powered-By: PHP/5.6.21

cgi.fix_pathinfo
```

```
$ curl -i http://www.example.net/pathinfo.php/.php
HTTP/1.1 200 OK
Server: nginx/1.8.1
Date: Sat, 14 May 2016 13:03:21 GMT
Content-Type: text/html; charset=UTF-8
Transfer-Encoding: chunked
Connection: keep-alive
X-Powered-By: PHP/5.6.21

cgi.fix_pathinfo
```

Как видно во втором запросе на самом деле был обработан файл pathinfo.php, несмотря на то, что мы запросили /pathinfo.php/.php. Теперь ставим 0, перезапускаем службу php и проверяем

снова

```
$ curl -i http://www.example.net/pathinfo.php
HTTP/1.1 200 OK
Server: nginx/1.8.1
Date: Sat, 14 May 2016 13:04:14 GMT
Content-Type: text/html; charset=UTF-8
Transfer-Encoding: chunked
Connection: keep-alive
X-Powered-By: PHP/5.6.21

cgi.fix_pathinfo
```

```
$ curl -i http://www.example.net/pathinfo.php/.php
HTTP/1.1 404 Not Found
Server: nginx/1.8.1
Date: Sat, 14 May 2016 13:04:16 GMT
Content-Type: text/html; charset=UTF-8
Transfer-Encoding: chunked
Connection: keep-alive
X-Powered-By: PHP/5.6.21

No input file specified.
```

Теперь при втором запросе nginx как и положено возвращает 404ю ошибку. Потенциально злоумышленник может загрузить на сервер картинку и при не достаточной проверке поместить в png файл php код и затем выполнить его, обратившись по адресу `/static/uploads/user_avatar.png/.php`. Идея, думаю, понятна. Так что лучше убрать такую возможность.