O'REILLY®

PHP

Nowe możliwości, najlepsze praktyki

JĘZYK PHP POWRACA W WIELKIM STYLU!

Helion

Josh Lockhart

Tytuł oryginału: Modern PHP

Tłumaczenie: Łukasz Piwko

ISBN: 978-83-283-1402-3

Wydawnictwo HELION
ul. Kościuszki 1c, 44-100 GLIWICE
tel. 32 231 22 19, 32 230 98 63
e-mail: *helion@helion.pl*
WWW: *http://helion.pl* (księgarnia internetowa, katalog książek)

Drogi Czytelniku!
Jeżeli chcesz ocenić tę książkę, zajrzyj pod adres
http://helion.pl/user/opinie/phpnom
Możesz tam wpisać swoje uwagi, spostrzeżenia, recenzję.

Pliki z przykładami omawianymi w książce można znaleźć pod adresem:
ftp://ftp.helion.pl/przyklady/phpnom.zip

Printed in Poland.

Spis treści

Część II Najlepsze praktyki

Część III Wdrażanie, testowanie i dostrajanie

ROZDZIAŁ 7.

Wdrażanie serwera

Po wybraniu hostingu dla aplikacji należy skonfigurować serwer i przygotować go do działania. Szczerze mówiąc, wdrażanie serwera to bardziej sztuka niż nauka. Wszystko zależy od wymagań aplikacji, której chcesz na danym serwerze używać.

Jeśli korzystasz z usługi typu PaaS, to zarządzaniem serwerem zajmuje się dostawca. Twoja rola sprowadza się do wykonywania usługi dostawcy, aby przenieść aplikację na jego platformę.

Jeśli nie korzystasz z usługi typu PaaS, to musisz przygotować do działania swój serwer VPS lub dedykowany. „Wdrażanie serwera" brzmi strasznie, ale w rzeczywistości nie jest takie trudne (nie śmiej się), choć wymaga podstawowej umiejętności posługiwania się wierszem poleceń. Jeżeli konsola to dla Ciebie czarna magia, to lepiej skorzystaj z usług PaaS takich firm jak Engine Yard albo Heroku.

Nie uważam się za administratora systemów, ale znajomość podstaw administracji systemami jest niezwykle przydatna dla każdego programisty, ponieważ ułatwia mu pracę i stworzenie elastycznego środowiska pracy. W tym rozdziale podzielę się z Tobą swoją wiedzą na ten temat, abyś mógł bez problemu uruchomić terminal i za jego pomocą wdrożyć serwer dla swojej aplikacji. Na koniec podsunę kilka dodatkowych źródeł, w których można znaleźć więcej informacji.

W tym rozdziale zakładam, że umiesz edytować pliki tekstowe za pomocą działającego w wierszu poleceń edytora typu nano (*http://www.nano-editor.org*) lub vim (*http://www.vim.org*) (są dostępne w większości linuksów). Jeśli nie, musisz znaleźć inny sposób na otwieranie i edytowanie plików na swoim serwerze.

Cel

Najpierw musisz wydzierżawić serwer wirtualny lub dedykowany. Następnie trzeba zainstalować na nim serwer sieciowy, który będzie obsługiwał żądania HTTP. Na koniec należy skonfigurować grupę procesów PHP do obsługi żądań PHP i zarządzać nimi. Procesy te muszą współpracować z serwerem.

Jeszcze kilka lat temu standardowo instalowało się serwer Apache i jego moduł `mod_php`. Serwer ten tworzy proces potomny do obsługi *każdego* żądania HTTP. Moduł `mod_php` osadza interpreter PHP w każdym takim procesie — nawet w tych, które serwują tylko statyczne zasoby, takie jak JavaScript, obrazy czy arkusze stylów. Jest to nieefektywne zarządzanie zasobami. Dlatego serwera Apache używa coraz mniej programistów, którzy przerzucają się na bardziej wydajne rozwiązania.

Dziś używa się serwera nginx (*http://nginx.org/*), który rezyduje przed kolekcją procesów PHP-FPM (i przekazuje do niej żądania PHP). To właśnie rozwiązanie przedstawiam w tym rozdziale.

Konfiguracja serwera

Zaczniemy od konfiguracji serwera VPS. Jestem absolutnym wielbicielem firmy Linode (*http://linode.com/*). Może nie jest to najtańszy dostawca usług, ale jeden z najbardziej niezawodnych. Wejdź na stronę Linode (albo swojego ulubionego dostawcy usług hostingowych) i zamów serwer VPS. Zostaniesz poproszony o wybranie dystrybucji Linuksa i zdefiniowanie hasła użytkownika *root* swojego nowego serwera.

Wielu dostawców serwerów VPS, np. Linode (*http://linode.com/*) i Digital Ocean (*https://www.digitalocean.com*), pobiera opłaty na godziny. To znaczy, że można uruchomić serwer VPS i pobawić się nim praktycznie za darmo.

Pierwsze logowanie

Gdy serwer jest już gotowy, należy się zalogować. Uruchom terminal w swoim lokalnym komputerze i połącz się z serwerem za pomocą polecenia `ssh`. Nie zapomnij wpisać adresu IP swojego VPS-a:

```
ssh root@123.456.78.90
```

Możesz zostać poproszony o potwierdzenie autentyczności swojego nowego serwera. Wpisz **`yes`** i naciśnij klawisz *Enter*:

```
The authenticity of host '123.456.78.90 (123.456.78.90)' can't be established.
RSA key fingerprint is 21:eb:37:f3:a5:d3:c0:77:47:c4:15:3d:3c:dc:3c:d1.
Are you sure you want to continue connecting (yes/no)?
```

Następnie zostaniesz poproszony o podanie hasła użytkownika *root*. Wpisz je i naciśnij klawisz *Enter*:

```
root@123.456.78.90's password:
```

Jesteś zalogowany w swoim nowym serwerze!

Aktualizacja oprogramowania

Kolejną czynnością powinno być zaktualizowanie systemu operacyjnego za pomocą poniższych poleceń:

```
# Ubuntu
apt-get update;
apt-get upgrade;

# CentOS
yum update
```

W trakcie ich wykonywania w konsoli będzie wyświetlanych wiele informacji dotyczących pobierania i instalowania aktualizacji różnych składników systemu. Jest to bardzo ważna pierwsza czynność, ponieważ zapewnia najnowsze aktualizacje i łatki zabezpieczeń serwera.

Inny użytkownik niż root

Nowy serwer nie jest zabezpieczony. Oto kilka porad, jak zwiększyć jego bezpieczeństwo.

Utwórz konto *innego użytkownika* niż *root* i w przyszłości do logowania się używaj tego nowego konta. Użytkownik *root* ma nieograniczone uprawnienia, jest bogiem. Może wykonać każde polecenie bez pytania o zgodę. *Dostęp do serwera przy użyciu konta tego użytkownika powinien być maksymalnie utrudniony.*

Ubuntu

Za pomocą polecenia przedstawionego na listingu 7.1 utwórz nowego użytkownika o nazwie `deploy`. Gdy zostaniesz o to poproszony, wpisz hasło i wykonaj pozostałe wyświetlane na ekranie instrukcje.

Listing 7.1. Tworzenie użytkownika innego niż root w systemie Ubuntu

```
adduser deploy
```

Następnie przypisz użytkownika `deploy` do grupy `sudo` za pomocą poniższego polecenia:

```
usermod -a -G sudo deploy
```

W ten sposób nadasz temu użytkownikowi uprawnienia `sudo`, czyli umożliwisz mu wykonywanie zadań wymagających zwiększonych uprawnień po podaniu hasła.

CentOS

Utwórz nowego użytkownika o nazwie `deploy` za pomocą polecenia przedstawionego poniżej. Gdy zostaniesz o to poproszony, wpisz i potwierdź hasło:

```
adduser deploy
```

Następnie przypisz użytkownika `deploy` do grupy `wheel` za pomocą poniższego polecenia:

```
usermod -a -G wheel deploy
```

W ten sposób nadasz temu użytkownikowi uprawnienia `sudo`, czyli umożliwisz mu wykonywanie zadań wymagających zwiększonych uprawnień po podaniu hasła.

Uwierzytelnianie poprzez SSH przy użyciu pary kluczy

Aby zalogować się na serwerze jako użytkownik `deploy` ze swojego komputera, możesz skorzystać z poniższego polecenia:

```
ssh deploy@123.456.78.90
```

Zostaniesz poproszony o podanie hasła tego użytkownika i gdy spełnisz tę prośbę, zostaniesz zalogowany. Proces logowania można lepiej zabezpieczyć przez wyłączenie uwierzytelniania za pomocą hasła. Wadą haseł jest to, że są podatne na ataki typu brute force polegające na wielokrotnym zgadywaniu hasła w krótkich sesjach. Zamiast tego skorzystamy z **uwierzytelniania przez SSH przy użyciu pary kluczy** (ang. *SSH key-pair authentication*).

Uwierzytelnianie tą metodą to skomplikowany temat. Najprościej mówiąc, polega na utworzeniu pary „kluczy" na komputerze lokalnym. Jeden z nich jest prywatny (pozostaje na maszynie lokalnej), a drugi jest publiczny (zostaje wysłany na serwer). Nazwa **para kluczy** bierze się stąd, że wiadomości zaszyfrowane przy użyciu klucza publicznego można rozszyfrować tylko przy użyciu odpowiedniego klucza prywatnego.

Gdy zalogujesz się do serwera, korzystając z uwierzytelniania poprzez SSH parą kluczy, na serwerze zostanie utworzona losowa wiadomość, która zostaje zaszyfrowana przy użyciu klucza publicznego i wysłana do Twojego komputera lokalnego. W komputerze lokalnym następuje rozszyfrowanie wiadomości przy użyciu klucza prywatnego i zwrócenie jej do serwera. Wtedy serwer sprawdza tę rozszyfrowaną wiadomość i przydziela dostęp do serwera. Bardzo wszystko uprościłem, ale myślę, że wiesz, o co chodzi.

Jeśli logujesz się na serwerze z wielu różnych komputerów, to lepiej nie korzystaj z metody uwierzytelniania przy użyciu pary kluczy, ponieważ musiałbyś wygenerować osobną parę dla każdego z tych komputerów oraz wysłać na serwer klucz publiczny z każdej z tych par. W takich przypadkach lepszym rozwiązaniem jest pozostanie przy uwierzytelnianiu za pomocą hasła. Ale jeśli do logowania się na serwerze używasz tylko jednego komputera (jak robi to wielu programistów), to powinieneś wybrać metodę z parą kluczy. Klucze można wygenerować na komputerze lokalnym za pomocą poniższego polecenia:

```
ssh-keygen
```

Wykonaj pojawiające się na ekranie instrukcje i podaj potrzebne informacje, gdy zostaniesz o to poproszony. Polecenie to tworzy dwa pliki w komputerze lokalnym: *~/.ssh/id_rsa.pub* (klucz publiczny) i *~/.ssh/id_rsa* (klucz prywatny). Klucz prywatny powinien pozostać na komputerze lokalnym i nie należy go nikomu ujawniać. Klucz publiczny natomiast wysyła się na serwer. Można to zrobić za pomocą polecenia `scp` (ang. *secure copy* — bezpieczne kopiowanie):

```
scp ~/.ssh/id_rsa.pub deploy@123.456.78.90:
```

Nie zapomnij dodać dwukropka na końcu! Polecenie to wysyła klucz publiczny do katalogu głównego użytkownika `deploy` na serwerze. Następnie zaloguj się na serwerze jako użytkownik `deploy`. Po zalogowaniu sprawdź, czy istnieje katalog *~/.ssh*. Jeśli nie, utwórz go za pomocą poniższego polecenia:

```
mkdir ~/.ssh
```

W dalszej kolejności utwórz plik *~/.ssh/authorized_keys* za pomocą poniższego polecenia:

```
touch ~/.ssh/authorized_keys
```

W pliku tym będzie zapisana lista kluczy publicznych, które mogą być używane do logowania się na tym serwerze. Wykonaj poniższe polecenie, aby dodać swój przed chwilą wysłany na serwer klucz do pliku *~/.ssh/authorized_keys*:

```
cat ~/id_rsa.pub >> ~/.ssh/authorized_keys
```

Na koniec należy jeszcze zmodyfikować uprawnienia dostępu do niektórych katalogów i plików, aby tylko użytkownik `deploy` miał dostęp do swojego katalogu *~/.ssh* oraz mógł odczytać zawartość swojego pliku *~/.ssh/authorized_keys*. Wykonaj poniższe polecenia:

```
chown -R deploy:deploy ~/.ssh;
chmod 700 ~/.ssh;
chmod 600 ~/.ssh/authorized_keys;
```

Zrobione! Od tej pory powinieneś móc połączyć się ze swojego komputera z serwerem bez podawania hasła.

Połączenie przez SSH z serwerem bez hasła jest możliwe tylko z komputera, na którym znajduje się klucz prywatny!

Wyłączanie haseł i możliwości logowania się użytkownika root

Serwer można jeszcze lepiej zabezpieczyć. W tym celu wyłączymy uwierzytelnianie przy użyciu hasła wszystkim użytkownikom oraz uniemożliwimy logowanie się użytkownika *root* — i koniec. Przypomnę, że użytkownik *root* może zrobić wszystko i dlatego staramy się maksymalnie utrudnić dostęp do serwera przy użyciu jego konta.

Zaloguj się na serwerze jako użytkownik `deploy` i otwórz plik */etc/ssh/sshd_config* w dowolnym edytorze tekstu. Jest to plik konfiguracyjny serwera SSH. Znajdź w nim ustawienie `PasswordAuthenitication` i zmień jego wartość na `no`. Jeśli jest wyłączone przez komentarz `;`, usuń go, aby je włączyć. Następnie znajdź ustawienie `PermitRootLogin` i jego wartość również zmień na `no`. Jeśli jest wyłączone przez komentarz `;`, usuń go, by je włączyć. Zapisz zmiany i aby je zastosować, ponownie uruchom serwer SSH za pomocą poniższego polecenia:

```
# Ubuntu
sudo service ssh restart

# CentOS
sudo systemctl restart sshd.service
```

Gotowe. Zabezpieczyłeś serwer, możesz więc przejść do instalowania programów potrzebnych do działania aplikacji PHP. Wszystkie dalsze instrukcje należy wykonywać na serwerze jako użytkownik `deploy`.

Bezpieczeństwo serwerów to dynamiczna dziedzina, w której postępy cały czas należy śledzić. Oprócz zastosowania wcześniej opisanych zabiegów zalecam też zainstalowanie zapory ogniowej. Użytkownicy systemu Ubuntu mogą skorzystać z UFW (*https://help.ubuntu.com/community/UFW*), a właściciele CentOS-a mogą używać iptables (*http://wiki.centos.org/HowTos/Network/IPTables*).

PHP-FPM

PHP-FPM (*http://php.net/manual/en/install.fpm.php*) (ang. *PHP FastCGI Process Manager*) to oprogramowanie zarządzające pulą powiązanych procesów PHP, która odbiera i obsługuje żądania od serwera sieciowego, takiego jak np. nginx. PHP-FPM tworzy jeden nadrzędny proces (zazwyczaj wykonywany przez użytkownika *root* systemu operacyjnego), który steruje przekazywaniem żądań HTTP do procesów potomnych. Ponadto proces nadrzędny sprawuje kontrolę nad tworzeniem procesów potomnych PHP (do obsługi dodatkowego ruchu sieciowego w aplikacji) i ich kasowaniem (jeśli są zbyt stare i niepotrzebne). Każdy proces puli PHP-FPM żyje dłużej niż pojedyncze żądanie HTTP i może obsługiwać 10, 50, 100, 500, a nawet więcej żądań HTTP.

Instalacja

PHP-FPM najłatwiej zainstalować przy użyciu systemowego menedżera pakietów, za pomocą poleceń pokazanych poniżej.

Szczegółowy poradnik na temat instalacji PHP-FPM znajduje się w dodatku A.

```
# Ubuntu
sudo apt-get install python-software-properties;
sudo add-apt-repository ppa:ondrej/php5-5.6;
sudo apt-get update;
sudo apt-get install php5-fpm php5-cli php5-curl \
     php5-gd php5-json php5-mcrypt php5-mysqlnd;

# CentOS
sudo rpm -Uvh \
  http://dl.fedoraproject.org/pub/epel/7/x86_64/e/epel-release-7-5.noarch.rpm;
sudo rpm -Uvh \
  http://rpms.famillecollet.com/enterprise/remi-release-7.rpm;
sudo yum -y --enablerepo=epel,remi,remi-php56 install php-fpm php-cli php-gd \
  php-mbstring php-mcrypt php-mysqlnd php-opcache php-pdo php-devel;
```

Jeśli instalacja za pomocą polecenia EPEL `rpm` nie uda się, uruchom przeglądarkę internetową i wejdź na stronę *http://dl.fedoraproject.org/pub/epel/7/x86_64/e/*. Poszukaj na niej aktualnej wersji pakietu EPEL i użyj jej.

Konfiguracja globalna

W Ubuntu główny plik konfiguracji PHP-FPM to */etc/php5/fpm/php-fpm.conf*. W systemie CentOS jest to z kolei plik */etc/php-fpm.conf*. Otwórz go w wybranym edytorze tekstu.

Pliki konfiguracyjne PHP-FPM są w formacie INI. Więcej informacji na jego temat można znaleźć w Wikipedii (*https://pl.wikipedia.org/wiki/INI*).

Poniżej opisuję dwa najważniejsze ustawienia, które zalecam zmienić. Domyślnie mogą one być wyłączone za pomocą komentarza, więc w razie potrzeby włącz je, usuwając ten komentarz. Ustawienia te nakazują głównemu procesowi PHP-FPM ponownie rozpocząć działanie, jeśli określona liczba jego procesów potomnych ulegnie awarii w zdefiniowanym czasie. Jest to podstawowe zabezpieczenie procesów PHP-FPM, które może pomóc rozwiązać proste problemy. Nie pomoże jednak w przypadku takich usterek jak źle napisany kod PHP.

`emergency_restart_threshold = 10`
: Maksymalna liczba procesów potomnych PHP-FPM, które mogą ulec awarii w określonym czasie, zanim główny proces PHP-FPM dokona ponownego uruchomienia.

`emergency_restart_interval = 1m`
: Ilość czasu dotycząca ustawienia `emergency_restart_threshold`.

Więcej informacji na temat globalnych ustawień konfiguracyjnych PHP-FPM można znaleźć na stronie *http://php.net/manual/en/install.fpm.configuration.php*.

Konfiguracja puli

W innym miejscu pliku konfiguracyjnego PHP-FPM znajduje się sekcja o nazwie `Pool Definitions` (definicje pul). W tym podrozdziale opisuję ustawienia dla każdej z nich. Pula PHP-FPM to zbiór powiązanych ze sobą procesów potomnych PHP. Zazwyczaj każda aplikacja ma własną osobną pulę.

W Ubuntu sekcja `Pool Definitions` zawiera następujący tekst:

```
include=/etc/php5/fpm/pool.d/*.conf
```

W systemie CentOS pliki definicji pul są dołączane na początku głównego pliku konfiguracyjnego za pomocą poniższego kodu:

```
include=/etc/php-fpm.d/*.conf
```

Kod ten nakazuje PHP-FPM załadowanie poszczególnych plików z definicjami pul z katalogu */etc/php5/fpm/pool.d/* (Ubuntu) lub */etc/php-fpm.d/ directory* (CentOS). Przejdź do tego katalogu, w którym powinien znajdować się jeden plik o nazwie *www.conf*. Jest to plik konfiguracyjny dla domyślnej puli PHP-FPM o nazwie `www`. Otwórz go w wybranym edytorze tekstu.

Każda konfiguracja puli PHP-FPM zaczyna się od znaku [, po którym wpisuje się nazwę puli i znak]. Na przykład domyślna konfiguracja PHP-FPM zaczyna się od napisu [www].

Każda pula PHP-FPM działa jako użytkownik systemu operacyjnego i określona grupa. Osobiście zazwyczaj uruchamiam pule jako osobnych użytkowników bez uprawnień użytkownika *root*. Dzięki temu mam ułatwione zadanie, gdy chcę znaleźć procesy wybranej aplikacji w wierszu poleceń za pomocą poleceń `top` i `ps aux`. Jest to też po prostu dobry nawyk, ponieważ procesy każdej puli PHP-FPM są ograniczone uprawnieniami swojego użytkownika systemowego i grupy.

Teraz pokażę, jak skonfigurować domyślną pulę `www`, aby działała jako użytkownik i grupa `deploy`. Otwórz zatem plik konfiguracyjny puli `www` w dowolnym edytorze tekstu. Poniżej opisuję ustawienia, których domyślne wartości zalecam zmienić:

`user = deploy`
: Użytkownik systemowy będący właścicielem procesów potomnych tej puli PHP-FPM. Należy wpisać nazwę użytkownika systemowego niemającego uprawnień użytkownika *root*.

`group = deploy`
: Grupa systemowa będąca właścicielem procesów potomnych tej puli PHP-FPM. Należy wpisać nazwę użytkownika systemowego niemającego uprawnień użytkownika *root*.

`listen = 127.0.0.1:9000`
: Adres IP i numer portu, na którym pula PHP-FPM nasłuchuje i przyjmuje żądania od serwera nginx. Wartość `127.0.0.1:9000` oznacza, że pula nasłuchuje połączeń na lokalnym porcie `9000`. Ja używam portu o tym numerze, ale może to być dowolny inny nieuprzywilejowany port (każdy o numerze większym od `1024`), który nie jest jeszcze używany przez inny proces systemowy. Do tego ustawienia wracam jeszcze przy opisie konfiguracji hosta wirtualnego na serwerze nginx.

`listen.allowed_clients = 127.0.0.1`
: Adresy IP, które mogą wysyłać żądania do tej puli PHP-FPM. Ze względów bezpieczeństwa ustawiam tę opcję na `127.0.0.1`, co oznacza, że tylko bieżący komputer może przekazywać żądania do tej puli. Domyślnie ustawienie to może być wyłączone za pomocą komentarza, konieczne może być więc jego włączenie.

`pm.max_children = 51`
: Wartość określająca maksymalną liczbę istniejących jednocześnie procesów puli PHP-FPM. Nie da się powiedzieć, jaka wartość jest najlepsza w tym ustawieniu. Trzeba przetestować aplikację, sprawdzić, ile pamięci wykorzystuje każdy proces, i ustawić taką liczbę procesów, jaką komputer jest w stanie obsłużyć. Większość małych i średnich aplikacji PHP wykorzystuje od 5 do 15 MB pamięci na proces (choć to tylko przybliżone dane). Jeśli dysponujesz 512 MB pamięci dla puli PHP-FPM, to możesz tę opcję ustawić na wartość `512 pamięci / 10 MB na proces`, czyli 51 procesów.

`pm.start_servers = 3`
: Liczba procesów puli PHP-FPM, które są dostępne natychmiast po uruchomieniu PHP-FPM. W tym przypadku również nie da się podać jedynej poprawnej wartości. Dla większości małych i średnich aplikacji zalecam wartość 2 lub 3. Dzięki temu początkowe żądania aplikacji nie muszą czekać na inicjację procesów PHP-FPM, ponieważ dwa lub trzy procesy już działają i czekają.

`pm.min_spare_servers = 2`
: Najmniejsza liczba procesów puli PHP-FPM, jaka istnieje, gdy aplikacja PHP jest nieaktywna. Wartość ta powinna mieścić się w podobnych granicach co wartość ustawienia `pm.start_servers`. Ustawienie to gwarantuje, że nowe żądania HTTP nie będą musiały czekać na inicjację nowych procesów do obsługi żądań.

`pm.max_spare_servers = 4`
: Największa liczba procesów puli PHP-FPM, jaka istnieje, gdy aplikacja PHP jest nieaktywna. Typowo wartość tego ustawienia powinna być nieco większa niż ustawienia `pm.start_servers`. Ustawienie to gwarantuje, że nowe żądania HTTP nie będą musiały czekać na inicjację nowych procesów do obsługi żądań.

`pm.max_requests = 1000`
: Maksymalna liczba żądań HTTP, jaką każdy proces z puli PHP-FPM obsługuje, zanim zostanie poddany recyklingowi. Ustawienie to pomaga uniknąć akumulacji wycieków pamięci powodowanych przez źle napisane rozszerzenia i biblioteki PHP. Zalecam wartość 1000, ale lepiej dostosować ją do konkretnych potrzeb aplikacji.

`slowlog = /ścieżka/do/slowlog.log`
: Bezwzględna ścieżka systemowa do pliku dziennika, gdzie rejestrowane są informacje o żądaniach HTTP, których przetwarzanie trwało dłużej niż *{n}* sekund. Informacje te są przydatne w znajdowaniu wąskich gardeł w aplikacjach PHP. Pamiętaj, że użytkownik i grupa puli muszą mieć uprawnienia zapisu w tym pliku. Wartość ***/ścieżka/do**/slowlog.log* to tylko przykład. Zamień ją na własną ścieżkę.

`request_slowlog_timeout = 5s`
: Ilość czasu, po jakiej następuje zrzut zawartości stosu bieżącego żądania HTTP do pliku dziennika określonego w ustawieniu `slowlog`. Wybór wartości zależy od tego, co uważamy za powolne żądanie. Na początek rozsądnym ustawieniem wydaje się `5s`.

Po wprowadzeniu i zapisaniu zmian w pliku konfiguracyjnym PHP-FPM ponownie uruchom proces nadrzędny za pomocą poniższego polecenia:

```
# Ubuntu
sudo service php5-fpm restart

# CentOS
sudo systemctl restart php-fpm.service
```

Więcej o konfiguracji pul PHP-FPM można przeczytać na stronie *http://php.net/manual/install.fpm.configuration.php*.

Serwer nginx

Serwer nginx (wym. in gen ex) to serwer sieciowy podobny do Apache, ale znacznie łatwiejszy w konfiguracji i często wykorzystujący mniej pamięci systemowej. Nie ma miejsca na szczegółowe opisywanie tego programu, ale przynajmniej pokażę, jak go zainstalować na swojej maszynie i przekazać odpowiednie żądania do puli PHP-FPM.

Instalacja

Najprostszą metodą instalacji serwera nginx jest użycie menedżera pakietów systemu operacyjnego.

Ubuntu

W Ubuntu serwer nginx można zainstalować przy użyciu PPA, czyli gotowego archiwum serwisowanego przez społeczność użytkowników:

```
sudo add-apt-repository ppa:nginx/stable;
sudo apt-get update;
sudo apt-get install nginx;
```

CentOS

W systemie CentOS serwer nginx instaluje się przy użyciu repozytorium oprogramowania EPEL, które dodaliśmy wcześniej. W domyślnych repozytoriach oprogramowania tego systemu może nie być najnowszej wersji serwera nginx:

```
sudo yum install nginx;
sudo systemctl enable nginx.service;
sudo systemctl start nginx.service;
```

Host wirtualny

Teraz pokażę, jak zdefiniować **wirtualnego hosta** na serwerze nginx dla aplikacji PHP. Host wirtualny to grupa ustawień określających nazwę domeny, położenie plików oraz sposób przekazywania żądań HTTP do puli PHP-FPM aplikacji.

Najpierw musimy wybrać miejsce do przechowywania plików aplikacji. Musi to być katalog, do którego odczytu i zapisu ma uprawnienia niebędący *rootem* użytkownik *deploy*. W tym przykładzie wybrałem katalog */home/deploy/apps/example.com/current*. Dodatkowo potrzebny jest katalog na dzienniki — ja wybrałem */home/deploy/apps/logs*. Poniższe polecenia tworzą te katalogi i nadają im odpowiednie zezwolenia:

```
mkdir -p /home/deploy/apps/example.com/current/public;
mkdir -p /home/deploy/apps/logs;
chmod -R +rx /home/deploy;
```

Swoją aplikację PHP umieść w katalogu */home/deploy/apps/example.com/current*. W konfiguracji hostów wirtualnych serwera nginx przyjęto, że aplikacja zawiera katalog *public/*, pełniący rolę katalogu głównego dokumentów wirtualnego hosta.

Każdy host wirtualny nginx ma własny plik konfiguracyjny. Jeśli używasz systemu Ubuntu, utwórz plik konfiguracyjny */etc/nginx/sites-available/example.conf*. Użytkownicy systemu CentOS powinni natomiast utworzyć plik */etc/nginx/conf.d/example.conf*. Otwórz plik *example.conf* w swoim ulubionym edytorze tekstu.

Ustawienia wirtualnego hosta nginx wpisuje się w bloku `server {}`. Poniżej znajduje się kompletna zawartość pliku konfiguracyjnego wirtualnego hosta:

```
server {
    listen 80;
    server_name example.com;
    index index.php;
    client_max_body_size 50M;
    error_log /home/deploy/apps/logs/example.error.log;
    access_log /home/deploy/apps/logs/example.access.log;
    root /home/deploy/apps/example.com/current/public;

    location / {
        try_files $uri $uri/ /index.php$is_args$args;
    }

    location ~ \.php {
        try_files $uri =404;
        fastcgi_split_path_info ^(.+\.php)(/.+)$;
        include fastcgi_params;
        fastcgi_param SCRIPT_FILENAME $document_root$fastcgi_script_name;
        fastcgi_param SCRIPT_NAME $fastcgi_script_name;
        fastcgi_index index.php;
        fastcgi_pass 127.0.0.1:9000;
    }
}
```

Skopiuj ten kod do pliku konfiguracyjnego *example.conf*. Nie zapomnij zmienić ustawienia `server_name` i ścieżek `error_log`, `access_log` i `root`. Oto zwięzłe objaśnienie każdego z zastosowanych ustawień:

`listen`
: Numer portu, na którym serwer nginx nasłuchuje przychodzących żądań HTTP. W większości przypadków dla ruchu HTTP jest to port 80, a dla HTTPS — 443.

`server_name`
: Nazwa domeny identyfikującej danego hosta wirtualnego. Należy wpisać własną domenę oraz ustawić ją na adres IP swojego serwera. Serwer nginx wysyła żądania HTTP do tego wirtualnego hosta, jeśli treść nagłówka `Host:` pasuje do wartości `server_name`.

`index`
: Pliki serwowane domyślnie, jeśli żaden nie zostanie określony w identyfikatorze URI żądania HTTP.

`client_max_body_size`
: Maksymalny rozmiar treści żądania HTTP akceptowany przez nginx dla tego hosta wirtualnego. Jeśli treść żądania będzie większa niż dozwolona wartość, nginx zwróci odpowiedź HTTP z rodziny *4xx*.

`error_log`
: Ścieżka do pliku dziennika błędów hosta wirtualnego.

`access_log`
: Ścieżka do dziennika dostępu hosta wirtualnego.

`root`
: Główny katalog dokumentów.

Dodatkowo w przedstawionej konfiguracji znajdują się jeszcze dwa bloki `location`. Zawierają one informacje na temat sposobu obsługi żądań HTTP pasujących do adresów URL zbudowanych wg określonych wzorców. Pierwszy blok `location / {}` zawiera dyrektywę `try_files`, która szuka prawdziwych plików pasujących do URI z żądania. Jeśli katalog nie zostanie znaleziony, przepisuje URI z żądania HTTP na */index.php* i dołącza łańcuch zapytania, jeśli jest dostępny. Przepisany adres URL lub każde żądanie kończące się przyrostkiem *.php* są obsługiwane przez blok `location ~ \.php {}`.

Blok `location ~ \.php {}` przekazuje żądania HTTP do puli PHP-FPM. Pamiętasz, że w jej konfiguracji został ustawiony port nasłuchu żądań o numerze 9000? Blok ten przekazuje właśnie żądania PHP do portu 9000, po czym pula PHP-FPM przejmuje kontrolę.

W bloku `location ~ \.php {}` znajduje się trochę więcej kodu. Uniemożliwia on przeprowadzanie ataków polegających na zdalnym wykonywaniu kodu (*http://bit.ly/remote-ex*).

W systemie Ubuntu konieczne jest utworzenie łącza symbolicznego między plikiem konfiguracyjnym hosta wirtualnego a katalogiem */etc/nginx/sites-enabled/* za pomocą poniższego polecenia:

```
sudo ln -s /etc/nginx/sites-available/example.conf \
     /etc/nginx/sites-enabled/example.conf;
```

Następnie należy ponownie uruchomić serwer nginx, korzystając z poniższego polecenia:

```
# Ubuntu
sudo service nginx restart

# CentOS
sudo systemctl restart nginx.service
```

Aplikacja działa i jest gotowa do użytku! Serwer nginx można skonfigurować na wiele sposobów. W tym rozdziale opisałem tylko najważniejsze ustawienia, ponieważ jest to książka o języku PHP, a nie o serwerze nginx. Więcej informacji znajdziesz w źródłach wymienionych poniżej:

- *http://nginx.org/*,
- *https://github.com/h5bp/server-configs-nginx*,
- *https://serversforhackers.com/editions/2014/03/25/nginx/*.

Automatyzacja wdrażania serwera

Wdrażanie serwera to czasochłonny i niezbyt przyjemny proces, zwłaszcza gdy trzeba ręcznie wdrożyć dużą liczbę serwerów. Na szczęście istnieją narzędzia pomagające w automatyzacji tych czynności. Oto niektóre z najpopularniejszych:

- Puppet (*http://puppetlabs.com/*),
- Chef (*https://www.getchef.com/chef/*),
- Ansible (*http://www.ansible.com/home*),
- SaltStack (*http://www.saltstack.com/*).

Każde narzędzie jest inne, ale wszystkie służą do tego samego celu — automatycznie wdrażają nowe serwery zgodnie z określoną specyfikacją. Jeśli więc zarządzasz wieloma serwerami, warto zainteresować się tymi programami, bo dzięki nim można oszczędzić mnóstwo czasu.

Zlecanie wdrażania serwerów

Istnieją też internetowe usługi wdrażania serwerów na życzenie klienta. Jedną z nich jest Forge (*https://forge.laravel.com/*) Taylora Otwella. Byłem beta testerem tej aplikacji i muszę powiedzieć, że jest naprawdę pomocna. W ofercie znajdują się wdrożenia serwerów u takich dostawców jak Linode, Digital Ocean i wielu innych popularnych firm oferujących serwery VPS.

Każdy serwer wdrażany przez Forge jest automatycznie zabezpieczany przy użyciu technik, które opisałem w tym rozdziale, oraz na każdym instalowane jest oprogramowanie nginx z PHP-FPM. Ponadto Forge ułatwia wdrażanie aplikacji PHP, instalację certyfikatów SSL, tworzenie zadań CRON oraz wykonywanie wielu innych nudnych lub trudnych zadań administracyjnych. Jeśli administracja systemami nie jest Twoją mocną stroną, to gorąco polecam Ci usługę Forge.

Dodatkowe źródła

Administracja systemami to dla mnie bardzo ciekawy temat. Nie chciałbym zajmować się tym na co dzień, ale lubię od czasu do czasu pogrzebać w wierszu poleceń. Moim zdaniem najlepszym źródłem wiedzy dla administratorów systemów jest książka *Servers for Hackers* (*https://book.serversforhackers.com/*) Chrisa Fidao.

Co dalej

W tym rozdziale opisałem metody wdrażania serwerów dla aplikacji PHP. W następnym pokażę Ci, jak wycisnąć z serwera siódme poty dla zwiększenia wydajności Twojej aplikacji.

Skorowidz

I

J

K

L

Ł

M

N

O

P

R

S

Ś

T

U

V

W

X

Z

O'REILLY®

Przekonaj się, jakie nowości kryje PHP!

PHP to dziś jeden z najbardziej niedocenianych języków programowania. W sieci można znaleźć mnóstwo artykułów wykluczających jego zastosowanie w zaawansowanych projektach. A przecież w obecnej wersji język ten jest pełnoprawnym kandydatem do zadań specjalnych! Przekonaj się, jak używać go we właściwy sposób!

Ta książka powinna obalić wszystkie mity o niedojrzałości języka PHP. Sięgnij po nią i przekonaj się, jakie fantastyczne możliwości daje Ci jego najnowsza wersja. Wśród nich znajdziesz domknięcia, wbudowany serwer HTTP, przestrzenie nazw oraz generatory. Ponadto poznasz standardy kodowania PSR oraz przekonasz się, jak korzystać z gotowych komponentów. Na kolejnych stronach znajdziesz informacje na temat najlepszych praktyk i dowiesz się, jak skonfigurować środowisko uruchomieniowe tak, żeby osiągnąć jego najwyższą wydajność. Ta książka jest obowiązkową lekturą dla wszystkich programistów języka PHP. Język PHP właśnie się odradza — lepszy i silniejszy!

Dzięki tej książce:

- poznasz nowości w języku PHP
- wykorzystasz wbudowany serwer HTTP do szybkiego uruchamiania projektów
- zaznajomisz się z najlepszymi praktykami tworzenia kodu
- dostroisz środowisko uruchomieniowe tak, żeby osiągnąć najwyższą wydajność
- przetestujesz swoje rozwiązanie
- odkryjesz język PHP na nowo

Josh Lockhart — twórca popularnego szkieletu Slim Framework. Pomysłodawca inicjatywy PHP The Right Way, promującej najlepsze praktyki kodowania w języku PHP. Programista w firmie New Media Campaigns, z siedzibą w Karolinie Północnej.

Helion
36528 numer katalogowy
księgarnia internetowa
http://helion.pl
zamówienia telefoniczne
0 801 339900
0 601 339900
Informatyka w najlepszym wydaniu

Sprawdź najnowsze promocje:
http://helion.pl/promocje
Książki najchętniej czytane:
http://helion.pl/bestsellery
Zamów informacje o nowościach:
http://helion.pl/nowosci

Helion SA
ul. Kościuszki 1c, 44-100 Gliwice
tel.: 32 230 98 63
e-mail: helion@helion.pl
http://helion.pl

sięgnij po WIĘCEJ
KOD KORZYŚCI

ISBN 978-83-283-1402-3
9 788328 314023
cena: 39,90 zł