Aplikacje WWW - konfiguracja serwera

Jan Wróblewski

Wersja z 5 maja 2015

1. Wybór serwera i systemu
2. Podstawowa konfiguracja serwera na KVM
3. Konfiguracja serwera www - apache2, mysql, php, phpmyadmin
4. Konfiguracja daemona DNS - bind9
5. Konfiguracja serwera mailowego Postfix
6. Konfiguracja firewalla, backupu, aktualizacji, monitorowanie

VPS vs serwer dedykowany

Na początku potrzebujemy wynająć maszynę na nasze aplikacje.

Mamy generalnie dwie opcje do wyboru:

- Virtual Private Server (VPS) - maszyna wirtualna współdzieląca zasoby (RAM, procesor, sieć, dysk) z innymi na mocnym serwerze;
- Dedicated server - fizyczna maszyna bez współdzielonych zasobów komputera (chociaż z współdzielonym łączem), zwykle słabsza niż host VPS.

Skoncentrujemy się na szukaniu maszyny mid-endowej, tj. dość szybkiej z kilka GB ramu, w granicach kilku-kilkudziesięciu dolarów miesięcznie.

VPS vs serwer dedykowany

Jest wiele różnic pomiędzy VPS i serwerem dedykowanym. Każdy nadaje się do innego zastosowania.

- VPS o podobnej mocy jest zwykle tańszy od serwera dedykowanego;
- VPS jest łatwiejszy w zarządzaniu - nie musimy się przejmować błędami sprzętowymi (np. bad sectory na dysku) i można robić snapshoty;
- Na VPS mamy zwykle mniej dostępnego RAMu, mniej dysku, ale mocniejszy procesor;
- Na VPS jesteśmy zwykle ograniczeni miesięcznym transferem danych, podczas gdy na serwerach dedykowanych mamy ograniczenie przepustowości łącza.

VPS vs serwer dedykowany

VPS jest dobry do aplikacji wymagających okazyjnie dużej mocy i małego transferu. Serwer dedykowany jest dobry do aplikacji wymagających stałego zużycia procesora lub dużo transferu. Jest dużo więcej ofert tanich VPS niż tanich serwerów dedykowanych.

Przykładowe parametry:

Serwer dedykowany	VPS
`http://online.net`	`http://ramnode.com`
Dedibox XC 2015	OpenVZ 2048MB SVZS
$15.99 / miesiąc	$14 / miesiąc
Intel C2750 (8c/8t @ 2.4GHz)	2 CPU (host Intel Xeon E3/E5)
8GB RAM	2GB RAM
1TB HDD / 120GB SSD	80GB SSD
1Gbps / guaranteed 200Mbps	3TB / miesiąc

VPS vs serwer dedykowany

Trudno jest znaleźć serwer dedykowany w granicach kilku-kilkunastu dolarów miesięcznie. Poniżej dwie opcje dla tanich serwerów dedykowanych:

- `http://www.online.net/en/dedicated-server/dedicated-server-overview-perso`;
- `http://kimsufi.com`.

Kilka opcji dla tanich VPS:

- `https://www.digitalocean.com`;
- `https://www.vultr.com`;
- `http://ramnode.com`.

VPS

W przypadku VPS mamy jeszcze do wyboru wirtualizację. Dwie najpopularniejsze to:

- OpenVZ - niepełna wirtualizacja jądra, szybsza, tańsza, zwykle z małym i nie konfigurowalnym swapem, tylko Linux;
- KVM - pełna wirtualizacja, bardziej konfigurowalna, dowolny swap, różne OS (np. Windows).

Istotne parametry firmy/serwera/data center

- Cena, wymagania naszych aplikacji;
- Lokalizacja (przynajmniej na tym samym kontynencie co użytkownicy);
- Opinie (`http://lowendtalk.com`, overselling, problemy z łączem, szybkość supportu);
- Benchmarki (`http://serverbear.com`, CPU, I/O, UnixBench).

System operacyjny

O ile nie potrzebujemy Windowsa z powodu jakiś konkretnych aplikacji, najlepiej użyć Linuxa - ma mniejszy odsetek błędów.

Dobre dystrubucje dla serwerów:

- RedHat - komercyjny linux z bardzo długim supportem, bardzo stabilny, komercyjny support dla biznesu;
- CentOS - bazowany na RedHat, darmowy, bardzo stabilny;
- Debian (stable distro) - darmowy, stabilny, łatwy w obsłudze, krótszy support.

Wybór na ten wykład

Cała konfiguracja w kolejnych rozdziałach będzie na najnowszym Debianie stable (wheezy) na VPS na KVM. Jest to dobra opcja dla początkujących.

Podstawowe narzędzia

Po zamówieniu serwera (niech ma IP `1.2.3.4`) i zainstalowaniu jego systemu operacyjnego możemy się na niego sshować.

- Instalujemy (`apt-get install`) nasz ulubiony konsolowy edytor;
- Appendujemy klucze ssh na serwerze i zmieniamy konfigurację ssh u siebie dla szybszego połączenia;
- Ustawiamy `.bashrc` na nasze ulubione środowisko (np. `PATH` do naszych skryptów).

Podstawowe narzędzia

Wybór przydatnych aplikacji

```
$ apt-get install vim less rcconf sudo rsync \
    nmap dnsutils mailutils links2 telnet \
    p7zip-full openjdk-7-jre build-essential \
    cmake git subversion python
```

Podstawowe narzędzia

.ssh/config

```
IdentityFile ~/.ssh/id_rsa

Host serwer
    Hostname 1.2.3.4
    User root
```

ssh

```
mojpc$ scp ~/.ssh/id_rsa.pub serwer:.
mojpc$ ssh serwer
serwer$ cat id_rsa.pub >> .ssh/authorized_keys
```

Hostname

Każdy serwer ma swój **hostname** (`foo`), **domenę** (`example.com`) i **FQHN** (fully qualified hostname, `foo.example.com`).

Uwaga

Ustawienie hostname, domeny i FQHN na początku jest **bardzo ważne**. Hostname/FQHN będzie się często pojawiał w konfiguracjach i bardzo ciężko jest go później zmienić.

Hostname

Chcemy doprowadzić do takiej sytuacji:

Sprawdzenie hostname, domeny i FQHN

```
$ hostname
foo
$ hostname -d
example.com
$ hostname -f
foo.example.com
```

Hostname

/etc/hostname

```
foo
```

/etc/hosts

```
127.0.0.1 localhost
1.2.3.4 foo.example.com foo

::1     localhost ip6-localhost ip6-loopback
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
```

Aktualizacja bez restartu: `hostname -F /etc/hostname`.

Swap

Na serwerze KVM nie mamy domyślnie swapa, ale możemy sobie go ustawić dowolnie dużego. Nie mamy zwykle całej partycji na swapa, więc tworzymy go w pliku.

4GB swap (4K * 1M)

```
$ dd if=/dev/zero of=/swapfile count=4K bs=1M
$ chmod 600 /swapfile
$ mkswap /swapfile
$ echo "/swapfile none swap sw 0 0" >> /etc/fstab
```

Restartujemy teraz system aby FQHN i swap się załadowały (komenda `reboot`).

Przydatne informacje

- Konfiguracje znajdują się w `/etc`, logi w `/var/log`;
- `ifconfig` przydaje się do przypomnienia sobie adresu IP i parametrów sieci;
- Za pomocą `service nazwa restart` resetujemy większość aplikacji serwerowych (i odpowiednio `start`, `stop`, `status`);
- Instalujemy aplikacje za pomocą `apt-get install`, a szukamy za pomocą `apt-cache search`.

Instalacja

Instalacja LAMP (standardowy serwer Linux-Apache-MySQL-PHP) z domyślną konfiguracją dobrze działa.

Instalacja

```
$ apt-get install apache2 mysql-server php5 \
    php5-mysql phpmyadmin
```

Instalacja

Podczas instalacji MySQL zostaniemy poproszeni o wpisanie hasła dla administratora bazy danych (`root`). Należy koniecznie go ustawić.

Przy instalacji phpMyAdmin wybieramy by się od razu skonfigurował dla `apache2`. Pozwalamy mu też skonfigurować bazę danych przez `dbconfig-common`. Podczas konfiguracji spyta się o hasło `roota` z MySQLa podane wcześniej. Na koniec poprosi o hasło dla użytkownika `phpmyadmin` - pozwalamy mu wygenerować losowe. Użytkownik MySQL `phpmyadmin` służy tylko do wewnętrznych celów phpMyAdmin.

Instalacja

Po instalacji powinniśmy być w stanie zobaczyć testową stronę (`http://1.2.3.4` w naszym przykładzie).

Możemy się połączyć z zainstalowaną bazą danych MySQL (domyślnie tylko lokalnie) przez `mysql -p` (opcjonalnie `-u user`).

Powinniśmy też móc zarządzać naszą danych przez phpMyAdmina (`http://1.2.3.4/phpmyadmin/`). Możemy się zalogować jako `root` z wpisanym wcześniej hasłem administratora MySQL.

Apache

Konfiguracja apache2 znajduje się w `/etc/apache2` w plikach `.conf` i innych o tej samej składni.

Konfiguracje apache2 potrafią includować całe foldery z konfiguracjami. W apache2 używa się techniki konfiguracji, w której w jednych folderach definiuje się wszystkie możliwe konfiguracje, a w innych tworzy się do nich symlinki (`ln -s`) i to je się includuje.

Aby wprowadzić w życie zmienioną konfigurację restartujemy serwer za pomocą `service apache2 restart`.

Apache

Ważniejsze pliki konfiguracyjne apache2 są następujące:

- `apache2.conf` - główny plik konfiguracyjny apache2;
- `ports.conf` - ustawienie na których portach ma serwer nasłuchiwać;
- `sites-available` - tu wstawiamy konfiguracje poszczególnych stron internetowych;
- `sites-enabled` - symlinki do `sites-available`;
- `mods-available` - definicje wszystkich zainstalowanych modów (nic nie zmieniamy);
- `mods-enabled` - symlinki do `mods-available`;
- `conf.d` - pozostałe pliki konfiguracyjne apache2 i symlinki do zewnętrznych konfiguracji (np. phpMyAdmina).

Apache

Konfiguracja całego serwera apache2 to tak naprawdę efekt wywołania jednego wielkiego skryptu, rozpoczynając w `apache2.conf` i includując po kolei mody z katalogu `mods-enabled`, konfigurację portów w `ports.conf`, dodatkowe ustawienia z katalogu `conf.d` i ustawienia poszczególnych stron z `sites-enabled`. Pliki mają tę samą składnię i podział jest wyłączenie dla wygody programisty.

Wiele dyrektyw jest zamkniętych w grupie poprzez `<grupa atrybuty>...</grupa>`. Takie grupy zwykle ograniczają zasięg konfiguracji do konkretnych przypadków.

Domyślne ustawienia `apache2.conf` i `ports.conf` są dobre. Skupimy się na tworzeniu konfiguracji stron.

Apache

Gdy apache2 otrzymuje żądanie `adresSerwera/sciezka` stara się dopasować `adresSerwera` do jednej z konfiguracji w `sites-enabled`, a następnie wyświetla plik w ścieżce `sciezka` folderu znalezionej podstrony.

Uwaga

Jeżeli `adresSerwera` nie dopasuje się do konkretnej strony poprzez dyrektywę `ServerName` czy `ServerAlias` to i tak może zostać użyta jedna z dyrektyw. Szczegóły wyboru `VirtualHosta`: `http://httpd.apache.org/docs/2.2/vhosts/details.html`.

Apache

Jeżeli `sciezka` jest folderem to szuka plików `index` z rozszerzeniami `html`, `cgi`, `pl`, `php` i innymi. Jeżeli wyświetlany plik jest skryptem, to jest on wywoływany. W przeciwnym razie jest on zwracany bezpośrednio.

To zachowanie może być zmodyfikowane przez mody i konfigurację serwera. Dalej będziemy pracować na domyślnych ustawieniach.

Apache

Domyślnie wyświetlana strona jest skonfigurowana w `sites-available/default` i wyświetla zawartość katalogu `/var/www`.

Stron w `sites-enabled` może być wiele, bo ten sam serwer może obsługiwać wiele subdomen i wiele IP, każde z osobną konfiguracją. Opiszemy w jaki sposób skonfigurować regułę, która konfiguruje stronę `example.com` na serwerze o IP `1.2.3.4`, nie dopasowuje się do subdomen `example.com` i serwuje stronę z katalogu `/var/www/example`.

Apache

/etc/apache2/sites-available/example.com

```
# konfiguracja strony na porcie 80
<VirtualHost *:80>
    # glowny adresSerwera to example.com
    ServerName example.com
    # inne obslugiwane adresy
    ServerAlias www.example.com
    ServerAlias 108.61.170.68
    # mail webmastera strony
    ServerAdmin foo@example.com

    # gdzie sa pliki strony
    DocumentRoot /var/www/example
```

Apache

/etc/apache2/sites-available/example.com c.d.

```
    # opcje dla naszych plikow
    <Directory /var/www/example>
        # dodatki, np. Indexes - listowanie plikow
        # w katalogu gdy nie ma pliku index.*
        Options Indexes FollowSymLinks MultiViews
        # pozwalamy na wszystko w .htaccess
        AllowOverride All
        # domyslnie wszystkie pliki sa serwowane
        Order allow,deny
        allow from all
    </Directory>
</VirtualHost>
```

Apache

Konfigurację stron trzymamy w `sites-available`. Aktywujemy je poprzez tworzenie symlinków w `sites-enabled` do plików z tego folderu.

Linkowanie jest uproszczone oraz otoczone dodatkowymi sprawdzeniemi przez programy `a2ensite` i `a2dissite`.

Aktywacja `example.com`

```
$ a2ensite example.com
```

Apache

Dodatkowe opcje do logów (we wnętrzu `<VirtualHost>`):

- `LogLevel` - jak bardzo poważne błędy logujemy;
- `ErrorLog`, `CustomLog`, ... - ścieżki logów.

Warto sprawdzić co jest naszym domyślnym `VirtualHostem`. Zwykle jest kilka subdomen nie służących do HTTP, ale wskazujących na serwer.

Apache

Można zrobić domyślne przekierowanie za pomocą `000-default` (pierwsze leksykograficznie w `sites-enabled`) ręcznie symlinkując poniższy kod (wymaga `mod_rewrite`).

/etc/apache2/sites-available/redirect

```
<VirtualHost _default_:80>
    # sztuczna nazwa by sie normalnie nie dopasowac
    ServerName default
    RewriteEngine on
    Redirect / http://example.com
</VirtualHost>
```

Apache

Mody aktywujemy i deaktywujemy za pomocą odpowiednio `a2enmod` i `a2dismod` (tworzy to odpowiednio symlinki).

Aktywacja mod_rewrite i SSL

```
$ a2enmod rewrite
$ a2enmod ssl
```

`mod_rewrite` to bardzo przydatny mod pozwalający na redefiniowanie ustawień strony dla konkretnych podfolderów za pomocą plików `.htaccess` o tej samej składni co konfiguracja apache2.

Apache

SSL w apache2 jest prosty do załączenia. Wystarczy `a2enmod ssl` i w `VirtualHost` wprowadzić dwie zmiany:

- Zmieniając port na 443 (`<VirtualHost *:443>`);
- W grupie `VirtualHost` dodając ustawienia certyfikatu.

Zwykłe ustawienia SSL

```
SSLEngine on
SSLCertificateFile /etc/apache2/ssl/example.com.crt
SSLCertificateKeyFile /etc/apache2/ssl/example.com.key
```

Możemy też wcześniej zduplikować `VirtualHost` aby strona była dostępna również bez SSL.

Apache

Tworzenie certyfikatu self-signed ważnego rok

```
$ mkdir /etc/apache2/ssl
$ cd /etc/apache2/ssl
$ openssl req -x509 -nodes -days 365 \
    -newkey rsa:2048 -keyout example.com.key \
    -out example.com.crt
$ chgrp ssl-cert example.com.key
$ chmod 640 example.com.key
```

Common Name (CN) musi być równe adresowi serwera, z którego będzie otwierana strona (tutaj `example.com`, a nie `foo.example.com`). Reszta może być pusta.

MySQL

Domyślna konfiguracja MySQL jest bardzo dobra. Możemy skonfigurować MySQL edytując pliki konfiguracyjne w `/etc/mysql`, a zmiany wprowadzić w życie przez `service mysql restart`.

Domyślnie baza danych pozwala tylko na połączenia z `localhost`. Jeżeli chcemy to zmienić, to należy zmienić `bind-address` w `/etc/mysql/my.cnf` (np. na `0.0.0.0` by pozwolić na połączenie z dowolnej sieci). Jest to często jednak niepotrzebna luka w bezpieczeństwie.

Ze względów bezpieczeństwa tworzymy nowego użytkownika MySQL z ograniczonymi uprawnieniami dla każdej aplikacji.

MySQL

Tworzenie użytkowników MySQL: pełne uprawnienia dla jednej DB i read-only wszystkich DB

```
mysql> CREATE USER 'aplikacja'@'localhost'
    IDENTIFIED BY 'haslo1';
mysql> GRANT ALL PRIVILEGES ON db_aplikacji.* TO
    'aplikacja'@'localhost';
mysql> CREATE USER 'backup'@'localhost'
    IDENTIFIED BY 'haslo2';
mysql> GRANT SELECT, RELOAD, FILE, SUPER,
    LOCK TABLES, SHOW VIEW ON *.* TO
    'backup'@'localhost';
```

Wordpress

Istnieje wiele gotowych aplikacji, które można zainstalować pod LAMP. Jedną z nich jest blog Wordpress. Instalacja takich aplikacji przebiega podobnie.

Na początek musimy ściągnąć naszą aplikację.

Pobranie Wordpressa

```
$ wget http://wordpress.org/latest.tar.gz
$ tar -xf latest.tar.gz
```

Wordpress

Rozpakowaną aplikację należy gdzieś umieścić (np. jako podfolder `/var/www`) i skonfigurować apache2 dla niej.

Wordpress i wiele innych aplikacji potrzebuje mieć możliwość zapisywania plików w folderze aplikacji. Apache domyślnie działa na uprawnieniach użytkownika `www-data` (konfigurowalne w `/etc/apache2/envvars`).

Przekazanie plików użytkownikowi `www-data`

```
$ chown -R www-data wordpress
$ chgrp -R www-data wordpress
```

Wordpress

Aplikacje często posiadają skrypt instalacyjny uruchamiający się przy wejściu na stronę lub jakąś specjalną podstronę (np. `install.php`). Skrypt instalacyjny zwykle potrzebuje:

- Adres serwera bazy danych (u nas `localhost`);
- Nazwę bazy danych (`CREATE DATABASE nazwa`);
- Użytkownika bazy danych i jego hasło (najlepiej utworzyć nowego użytkownika z uprawnieniami RW do jego bazy danych);
- Czasem prefiks nazw tabel w bazie danych - przydatne, gdy aplikacje współdzielą bazę danych.

Generuje on zwykle plik `*config.php` z tymi danymi w postaci **plaintextu**, więc należy zadbać o odpowiednie uprawnienia dostępu (600).

Django

apache2 jest bardzo konfigurowalny i potrafi obsłużyć wiele frameworków do aplikacji www, w tym Django.

Tutorial do konfiguracji: `https://docs.djangoproject.com/en/1.7/howto/deployment/wsgi/`.

Używany do tego jest `mod_wsgi`. Można go pobrać z repo w paczce `libapache2-mod-wsgi` dla Pythona 2 i `libapache2-mod-wsgi-py3` dla Pythona 3.
Przyda się też `pip`. Jest on częścią Pythona od 2.7.9 i od 3.4. Gdy mamy starszą wersję instalujemy z paczki `python-pip` lub `python3-pip`.
Instalujemy też `pip install virtualenv`.

Django

Stworzenie projektu pod virtualenv (VENV, PROJ do podmiany)

```
$ cd /var/www; virtualenv VENV
$ cd VENV; django-admin startproject PROJ
```

Kawałek /etc/apache2/sites-available/PROJ dla Pythona 2

```
WSGIDaemonProcess example.com python-path= \
    /var/www/VENV/PROJ:/var/www/VENV/lib/python2.7/site-packages
WSGIProcessGroup example.com
WSGIScriptAlias / /var/www/VENV/PROJ/PROJ/wsgi.py
```

`Require all granted` jest potrzebne tylko dla apache $\geqslant$ 2.4.

Nameserver

Po kupnie domeny potrzeba powiedzieć na co ma wskazywać. Często można ją skonfigurować z serwera DNS dostawcy. Jeżeli jednak chcemy mieć nad nią pełną kontrolę to możemy postawić własny serwer DNS.

Aby został użyty nasz serwer DNS należy u dostawcy gdzieś w panelu administracyjnym wskazać jakie są nasze **nameservery** (może się to nazywać rekordem GLUE). Należy mieć conajmniej dwa, ale mogą mieć to samo IP. Zaleca się, by IP było różne - drugi nameserver będzie używany gdy pierwszy padnie. Zwyczajowo nazywa się je w taki sposób: `ns1.example.com`, `ns2.example.com`.

Nameserver

Ze względu na cacheowanie proces zmiany serwera DNS (zwany DNS propagation) może potrwać od paru godzin do dwóch dni.

Istnieje wiele narzędzi online pomagających w sprawdzaniu ustawień serwerów, w tym do śledzenia propagacji DNS. Jest to między innymi `https://www.whatsmydns.net`.

Konfiguracja bind9

Standardowym daemonem DNS jest bind9 (czasem nazywany również named).

Instalacja bind9

```
$ apt-get install bind9
```

Jego konfiguracja znajduje się w `/etc/bind`, z czego część jest wygenerowana dobrze podczas instalacji.

Więcej informacji o konfiguracji bind9 można znaleźć w internecie, m. in. w tutorialu `https://help.ubuntu.com/community/BIND9ServerHowto`.

Konfiguracja bind9

Interesują nas przede wszystkim trzy pliki:

- `named.conf.local` - deklaracje dla domen i reverse DNS;
- Folder `zones` z defnicjami subdomen;
- `named.conf.options` - DNS do obsługi żądań spoza naszej domeny. Domyślnie jest dobrze - najlepiej zostawić bliski serwerowi DNS, ewentualnie dodać zapasowy.

Konfiguracja bind9

W `named.conf.local` tworzymy dwa rekordy.

Zone dla naszej domeny

```
zone "example.org" {
    type master;
    file "/etc/bind/zones/example.com.db";
};
```

Zone dla reverse DNS

```
zone "3.2.1.in-addr.arpa" {
    type master;
    file "/etc/bind/zones/3.2.1.in-addr.arpa";
};
```

Konfiguracja bind9

/etc/bind/zones/3.2.1.in-addr.arpa

```
@ IN SOA ns1.example.com. root.example.com. (
    2006081501 ; serial
    86400 ; refresh
    86400 ; retry
    604800 ; expire
    86400 ; negative cache TTL
)

    IN    NS    ns1.example.com.
    IN    NS    ns2.example.com.
4   IN    PTR   example.com.
```

Konfiguracja bind9

`3.2.1` w poprzedniej konfiguracji wzięło się z pierwszych trzech segmentów IP w odwrotnej kolejności. W tej są nasze dwa nameservery i "4" - ostatni segment naszego IP. Kropki na końcu domen są ważne.
Nagłówek zawiera informacje o oświeżaniu danych, a każda linijka poniżej jest jednym rekordem w DNS.

Więcej informacji: `https://wiki.debian.org/Bind9` i Google.

Plikiem, który będzie nas najbardziej interesował, będzie zawierał subdomeny i będzie aktualizowany w przyszłości jest `zones/example.com.db`.

Konfiguracja bind9

/etc/bind/zones/example.com.db bez nagłówka

```
example.com.  IN  NS     ns1.example.com.
example.com.  IN  NS     ns2.example.com.
example.com.  IN  MX  0  example.com.
fooalias IN  CNAME foo.example.com.
ns1      IN  A     1.2.3.4
ns2      IN  A     1.2.3.4
foo      IN  A     1.2.3.4
@        IN  A     1.2.3.4
```

W nagłówku serial jak poprzednio, a czasy np. 10800, 3600, 604800, 10800.

Konfiguracja bind9

W pierwszych dwóch linijkach są rekordy NS - nameserverów.

W trzeciej linijce jest rekord MX o priorytecie 0. Oznacza to adres serwera mailowego o priorytecie zero (mniejszy - najpierw próbowany).

Czwarta linijka zawiera rekord CNAME - alias domeny. CNAME jest wolniejszy od A, bo powoduje kolejny lookup.

Kolejne linijki zawierają rekordy A - przyporządkowanie subdomenom adresu IP. Może to być adres zewnętrzny. Trzeba tu też podać adresy IP nameserverów.

Konfiguracja bind9

`@` jest specjalnym makrem dla nazwy tego zone, tj. `example.com.` w tym przypadku.

Po lewej stronie `IN` może stać zarówno nazwa subdomeny bez domeny (`foo`) jak i pełna nazwa subdomeny zakończona kropką (`foo.example.com.`).

Konfiguracja bind9

Warto sprawdzić konfigurację bind9 pod względem poprawności.

Sprawdzanie składni konfiguracji bind9

```
$ named-checkconf
```

Sprawdzanie zone `example.com` (zone zdefiniowane w `/etc/bind/named.conf.local`)

```
$ named-checkzone example.com \
    /etc/bind/zones/example.com.db
```

bind9 możemy zrestartować z nową konfiguracją za pomocą standardowego `service bind9 restart`.

Konfiguracja bind9

Do testowania serwera DNS przyda się narzędzie `dig` z paczki `dnsutils`.

Zapytanie DNS o rekordy A i MX (wynik w ANSWER SECTION)

```
$ dig a foo.example.com
$ dig mx example.com
```

Istnieją też narzędzia online sprawdzające ustawienia serwera DNS i nie tylko. Jednym z takich narzędzi sprawdzających wiele różnych aspektów serwera jest `http://mxtoolbox.com/domain/`.

Konfiguracja delegowania DNS i slave

Czasem chcemy stworzyć subdomenę wskazującą na zewnętrzny serwer z własnym DNS. Niech `sub.example.com` o IP `2.3.4.5` będzie serwerem dla konfigurowanej subdomeny.

W celu skonfigurowania tego DNS wystarczy dodać dwa nameservery `sub.example.com` do zone głównego serwera, a serwer DNS subdomeny skonfigurować niezależnie.

Dodawanie nameserverów subdomeny do zone głównego serwera

```
sub      IN NS  ns1.sub
sub      IN NS  ns2.sub
ns1.sub IN A 2.3.4.5 ; rekord GLUE
ns2.sub IN A 2.3.4.5 ; rekord GLUE
```

Konfiguracja delegowania DNS i slave

Więcej informacji o delegowaniu DNS:
`http://www.zytrax.com/books/dns/ch9/delegate.html`.
Jak wspomniane w powyższym linku, dobrą praktyką jest, by główny serwer od razu stał się slave DNS dla serwera subdomeny.

Generalnie jeżeli ma więcej niż jeden serwer to warto jeden z nich zrobić slavem DNS (`ns2`, ...). Podana dotychczas konfiguracja zakłada, że oba nameservery będą tą samą maszyną.
Skonfigurowanie kolejnych nameserverów na prawdziwe slavey pozwala uniknąć downtime DNS w razie awarii.

Informacje jak skonfigurować slave DNS znajdują się w internecie, m. in. `http://www.microhowto.info/howto/configure_bind_as_a_slave_dns_server.html`.

Serwer mailowy

Jest kilka serwerów mailowych do wyboru. Skupimy się na zestawie:

- Postfix - serwer mailowy (MTA, obsługa SMTP);
- Dovecot - do obsługi IMAP/POP3;
- Cyrus SASL - do uwierzytelniania.

Porównania serwerów mailowych:

- `http://shearer.org/MTA_Comparison`;
- `https://wiki.debian.org/Debate/DefaultMTA`.

Serwer mailowy

Jest dużo do skonfigurowania, by postawić serwer mailowy.

- Składa się on z kilku niezależnych aplikacji, z których każdą trzeba skonfigurować;
- Brak szyfrowania jest niebezpieczny, więc musimy je dodać;
- Kilka ustawień, by maile z serwera nie zostały uznane za spam;
- Musimy uważać, skąd i dokąd akceptujemy maile.

Postfix

Na początku skonfigurujmy sam Postfix.

Instalacja Postfix

```
$ apt-get install postfix
```

Podczas instalacji wybieramy typ konfiguracji "Internet Site". W "System mail name" wpisujemy domenę, na której chcemy pracować.

Przy domyślnych ustawieniach i właściwym ustawieniu FQHN możemy już wysłać maila do użytkowników lokalnych zarówno z lokalnej maszyny jak i z zewnątrz. Domyślnie będą też odrzucane maile nie skierowane do naszego FQHN.

Postfix

Konfiguracja Postfix znajduje się w katalogu `/etc/postfix`. Prawie wszystkie interesujące nas opcje są w `main.cf`. Zmienne z tego pliku są opisane udokumentowane na stronie `http://www.postfix.org/postconf.5.html`.
Możemy sprawdzić wartości parametrów za pomocą polecenia `postconf`.

postconf

```
$ postconf alias_maps # obecna wartosc
alias_maps = hash:/etc/aliases
$ postconf -d alias_maps # domyslna wartosc
alias_maps = hash:/etc/aliases, nis:mail.aliases
$ postconf # wszystko
...
```

Postfix

Przeładowujemy konfigurację Postfix za pomocą `postfix reload`. Jest to szybsze niż restart, tj. `service postfix restart`.

Załóżmy, że mamy serwer mailowy `foo.example.com`. Musimy podjąć decyzję, czy będziemy obsługiwać maile `@example.com`, `@foo.example.com` czy może jeszcze inne. Załóżmy, że chcemy obsługiwać obie, ale by serwer przedstawiał się jako `example.com`.

Definiujemy naszą domenę w kilku miejscach (zmieniając/odkomentowując/dodając konfigurację):

- `myhostname` i `mydomain` ustawiamy na `example.com`;
- `myorigin` ustawiamy na `/etc/mailname` i modyfikujemy ten plik na `example.com`;
- Dodajemy rekord MX `example.com`.

Postfix

Rekordy MX ustawiamy w naszym serwerze DNS. Rekordy MX wyglądają jak rekordy CNAME, ale mają jeszcze dodatkowy numeryczny argument - priorytet. Przy wysyłaniu maila następują próby wysłania od rekordów z najmniejszym priorytetem.

W przypadku serwera DNS bind9 składnia jest jak poniżej.

Rekord MX w bind9

```
example.com. IN MX 0 example.com.
example.com. IN MX 10 backup.example.com.
```

Postfix

Musimy jeszcze ustalić które maile mają dochodzić do skrzynek lokalnych użytkowników.

mydestination

```
mydestination = localhost.$mydomain localhost
    example.com foo.example.com
```

Dzięki temu maile nadane z naszego serwera na te domeny będą dochodzić bezpośrednio do użytkowników.

relay_domains

```
relay_domains = example.com foo.example.com
```

Określa to jakie domeny użytkowników są akceptowane dla maili przychodzących z zewnątrz.

Postfix

Musimy jeszcze skonfigurować użytkowników. Domyślnie wszystkie maile do użytkownika `x@example.com` będą dochodzić do użytkownika linuxa `x` (i będą gromadzić się w pliku `/var/mail/x`. Jeżeli użytkownik `x` nie istnieje to nastąpi błąd.

Możemy stworzyć aliasy dla użytkowników. Robimy to za pomocą `alias_maps` i `alias_database`. Domyślnie jest to skonfigurowane na `hash:/etc/aliases`, czyli na plik `/etc/aliases` w formacie każdej linijki `alias: user/alias`.

Przykładowe /etc/aliases

```
mailer-daemon: postmaster
postmaster: root
root: user
```

Postfix

Zgodnie z RFC 5321 (SMTP) powinniśmy stworzyć alias/użytkownika `postmaster`. Inne RFC wymagają np. `webmaster`. Zamiast tworzyć użytkowników warto dodać tutaj aliasy.
Warto również przekierować maile użytkownika root do jakiegoś normalnego użytkownika. Różne serwery IMAP i POP albo zabraniają albo przynajmniej nie polecają możliwości logowania jako root.

Po modyfikacji pliku `/etc/aliases` należy wykonać polecenie `newaliases`, które utworzy dla niego plik `/etc/aliases.db`. Dopiero po jego modyfikacji (i restarcie Postfixa) zmiany wejdą w życie.

Postfix

Możemy zwiększyć kontrolę nad aliasami za pomocą wirtualnych aliasów.

virtual_alias_maps

```
virtual_alias_maps = hash:/etc/postfix/virtual
```

Plik podany w tej ścieżce należy utworzyć. Podobnie jak dla alias_maps, aby weszło to w życie musimy stworzyć plik .db.

Tworzenie /etc/postfix/virtual.db

```
$ postmap /etc/postfix/virtual
```

Postfix

Format pliku `virtual` pozwala na mapowanie nie tylko użytkowników, ale całych domen/subdomen. Pozwala również na przekierowywanie maili na zewnątrz. `virtual_alias_maps` może istnieć obok `alias_maps`.

Przykładowy `/etc/postfix/virtual`

```
# przekierowanie dla konkretnej pary user-domena
info@example.com user
# przekierowanie na zewnetrznego maila
webmaster@foo.example.com joe@example.org
# catch-all dla jednej z domen
@example.com root
```

Postfix

]
Można również stworzyć wirtualne skrzynki pocztowe nie powiązane z konkretnymi linuxowymi użytkownikami. Więcej informacji o tym oraz o `virtual_alias_maps` na stronie `http://www.postfix.org/VIRTUAL_README.html`.

Na tym etapie mamy w pełni działające SMTP - możemy wysyłać i pobierać maile będąc na serwerze.

Postfix

W razie problemów z Postfixem logi serwera mailowego są w `/var/log/mail.log`.

Jeżeli chcemy, by `postmaster` był informowany o maksymalnej ilości błędów możemy zmienić `notify_classes`.

`notify_classes`

```
notify_classes = bounce data delay policy protocol
    resource software
```

Wysyłanie maila

Wygodnym narzędziem do wysyłania testowych maili jest polecenie `mail` i `telnet` z paczek `mailutils` i `telnet`.

Polecenie `mail` służy do sprawdzania lokalnej poczty użytkownika (z `/var/mail`) i wysyłania maili z lokalnej maszyny.

Polecenie `telnet` jest generycznym programem do ręcznej obsługi protokołów tekstowych, m. in. SMTP i POP3. Jest bardzo dobre do wysyłania sfabrykowanych maili w celu przetestowania zabezpieczeń.

Wysyłanie maila

Ważniejsze komendy `mail`:

- `next/n` - przeczytanie kolejnego maila;
- `delete/d/delete <numer>` - usunięcie maila;
- `<numer>` - wyświetlenie maila o danym numerze;
- `hold/hold <numer>` - nie przeniesienie przeczytanego maila do `~/mbox`.

Domyślnie maile po przeczytaniu są przenoszone do `~/mbox`.
Polecenia bez numerów odnoszą się do ostatnio czytanego maila.

Wysyłanie maila

Używanie mail

```
$ echo "tresc zewnetrznego maila" | mail -s \
    "test" jakis@zewnetrzny.adres.email
$ echo "tresc lokalnego maila" | mail -s "temat" root
$ mail
"/var/mail/root": 1 message 1 new
>N   1 root                Tue Apr 14 02:19 13/458  temat
? 1
...
Subject: temat
...
tresc lokalnego maila
? delete
? quit
Held 0 messages in /var/mail/root
```

Wysyłanie maila

SMTP przez `telnet` (z lokalnej maszyny)

```
$ telnet localhost 25
Trying ::1...
Connected to localhost.
Escape character is '^]'.
220 example.com ESMTP Postfix (Debian/GNU)
HELO my-fqhn
250 example.com
MAIL FROM:<my-email@example.com>
250 2.1.0 Ok
RCPT TO:<some-email@example.net>
250 2.1.5 Ok
DATA
354 End data with <CR><LF>.<CR><LF>
Subject: topic
Content.
.
250 2.0.0 Ok: queued as 7A016A4D1
QUIT
221 2.0.0 Bye
Connection closed by foreign host.
```

Wysyłanie maila

SMTP przez `telnet` (z zewnętrznej maszyny)

```
$ telnet 1.2.3.4 25
Trying 1.2.3.4...
Connected to 1.2.3.4.
Escape character is '^]'.
220 example.com ESMTP Postfix (Debian/GNU)
HELO spammer
250 example.com
MAIL FROM:<fake@email.example.com>
250 2.1.0 Ok
RCPT TO:<spam-target@example.net>
554 5.7.1 <spam-target@example.net>: Relay access denied
```

Wysyłanie maila

Jeżeli mamy poprawnie skonfigurowany serwer to łącząc się do niego `telnetem` z zewnątrz (drugi przykład) zostaną odrzucone maile z `RCPT TO:` (recipent) nie należące do naszej domeny.

Serwery pozwalające na wysyłanie maili od każdego do każdego to tzw. open relay i są stosunkowo szybko wykrywane, wykorzystywane do spamu i dodawane do blacklist. Jest to poważna luka w bezpieczeństwie.

Więcej szczegółów o SMTP w internecie: `http://en.wikipedia.org/wiki/Simple_Mail_Transfer_Protocol`.

Dovecot - SMTP

Obecnie użytkownicy muszą zalogować się na serwer, by móc wysłać lub odczytać maile. Chcemy skonfigurować protokoły SMTP, IMAP i POP, by mogli robić to z zewnątrz za pomocą klientów email (np. Thunderbird, Outlook).

Protokół SMTP to protokół zarówno pomiędzy serwerami mailowymi jak i do wysyłania maila przez klienta email. Protokoły IMAP i POP służą do pobierania maili.

Aby umozliwić zewnętrznym użytkownikom na wysyłanie maila od nas należy ich najpierw autoryzować (inaczej będziemy open relay). Ponieważ w SMTP hasła są przesyłane plaintextem chcemy również włączyć szyfrowanie. Podobnie w IMAP i POP.

Dovecot - SMTP

SMTP i szyfrowanie jest obsługiwane przez Postfix, ale autoryzacja jest osobnym programem. Postfix obsługuje autoryzację przez Cyrus SASL lub Dovecot. Użyjemy Dovecota, który służy również do obsługi IMAP i POP.

Instalacja Dovecot

```
$ apt-get install dovecot-core dovecot-imapd \
    dovecot-pop3d
```

Dovecot - SMTP

Informacje w internecie jak poprawnie skonfigurować autoryzację Dovecot SMTP różnią się dla różnych dystrybucji i sposobów konfiguracji. Poniżej linki do kilku tutoriali:

- `http://mysql-apache-php.com/mailserver.htm` - cały zestaw, ale na CentOS/Red Hata;
- `http://wiki2.dovecot.org/HowTo/PostfixAndDovecotSASL` - krótka instalacja;
- `http://www.postfix.org/SASL_README.html` - dokumentacja po stronie Postfix;
- `https://www.digitalocean.com/community/tutorials/how-to-set-up-a-postfix-e-mail-server-with-dovecot` - dodatkowo o szyfrowaniu;
- `http://www.postfix.org/TLS_README.html` - samo szyfrowanie.

Dovecot - SMTP

Aktywujemy oba rodzaje autoryzacji plaintext, `plain` i `login`, odkomentowując i edytując odpowiednią opcję w `/etc/dovecot/conf.d/10-auth.conf`. Nie potrzebujemy więcej, ponieważ zabezpieczymy kanał szyfrowaniem.

Zarówno `plain` jak i `login` przekazują nazwę użytkownika i hasło plaintextem zakodowanym base64. To drugie jest/było używane przez klientów email Microsoft Outlook (`http://www.dovecot.org/list/dovecot/2006-March/012199.html`).

Kawałek `/etc/dovecot/conf.d/10-auth.conf`

```
auth_mechanisms = plain login
```

Dovecot - SMTP

Szukamy w `/etc/dovecot/conf.d/10-master.conf` stringa `unix_listener /var/spool/postfix/private/auth` i modyfikujemy tę opcję.

Kawałek `/etc/dovecot/conf.d/10-master.conf`

```
# Postfix smtp-auth
unix_listener /var/spool/postfix/private/auth {
    mode = 0666
    user = postfix
    group = postfix
}
```

Po modyfikacji restartujemy dovecot przez `service dovecot restart`.

Dovecot - SMTP

Na koniec musimy skonfigurować sam Postfix.

Konfiguracja `/etc/postfix/main.cf` do pracy z Dovecot

```
smtpd_sasl_auth_enable = yes
smtpd_sasl_type = dovecot
smtpd_sasl_path = private/auth
```

Autoryzacja plaintextem powinna już działać. Logi o błędach w autoryzacji znajdują się w `/var/log/auth.log`.

Dovecot - SMTP

Dobrze jest przetestować autoryzację SMTP telnetem. Opis SMTP AUTH pod `http://www.fehcom.de/qmail/smtpauth.html`.

Należy pamiętać, że Dovecot nie pozwoli na autentykację `roota`, nawet jeżeli podamy poprawne hasło. Jest tak zarówno dla SMTP jak i IMAP/POP.

Będzie potrzebne użycie kodowania base64. W konsoli do tego służy `echo` pipeowane do programu `base64`. Gdybyśmy chcieli zdekodować dane to jest `base64 --decode`.

Dovecot - SMTP

SMTP AUTH PLAIN

```
$ echo -en "user\0user\0pass" | base64
dXNlcgB1c2VyAHBhc3M=
$ telnet foo.example.com 25
Trying 1.2.3.4...
Connected to 1.2.3.4.
Escape character is '^]'.
220 example.com ESMTP Postfix (Debian/GNU)
HELO user
250 example.com
AUTH PLAIN
334
dXNlcgB1c2VyAHBhc3M=
235 2.7.0 Authentication successful
QUIT
221 2.0.0 Bye
Connection closed by foreign host.
```

Dovecot - SMTP

Kilka opcji Postfix wpływających na bezpieczeństwo:

- `smtpd_sasl_security_options = noanonymous` - ograniczenie metod autentykacji do nie-anonimowych;
- `smtpd_tls_security_level = encrypt` - wymuszenie szyfrowania do SMTP AUTH zamiast pozostawienia go opcjonalnym;
- `smtpd_client_connection_count_limit`, `smtpd_client_connection_rate_limit`, `smtpd_client_message_rate_limit` - zabezpieczenie przed floodem;
- `smtpd_tls_protocols = !SSLv2, !SSLv3` - zabronienie używania tych starych protokołów;
- `smtpd_recipient_restrictions`, `smtpd_helo_restrictions`, `smtpd_client_restrictions` - definiowanie listy zabezpieczeń do pobierania/wysyłania maili;
- `smtpd_tls_cert_file`, `smtpd_tls_key_file` - ustawienie certyfikatu zamiast domyślnego kiepskiego.

Dovecot - IMAP i POP3

Użyjemy zaintalowanego wcześniej Dovecota do obsługi protokołów IMAP i POP3 do pobierania poczty. IMAP i POP3 to różne protokoły, gdzie POP3 pozostawia maile na serwerze a IMAP nie.

Konfiguracja IMAP i POP3 dla Dovecot znajduje się również w `/etc/dovecot`. Większość ustawień jest zakomentowanych i udokumentowanych w różnych plikach w katalogu `conf.d`.

Dovecot - IMAP i POP3

Domyślne ustawienia działają by nawiązać szyfrowane połączenie. Ciekawsze opcje:

- `ssl = required` w `conf.d/10-ssl.conf` aby wymusić szyfrowanie;
- `protocols = imap imaps pop3 pop3s` zamiast domyślnie includowanych w `dovecot.conf`;
- `disable_plaintext_auth = no` do przetestowania IMAP lub POP3.

Jeżeli pozwalamy na autentykację plaintextem bez SSL/TLS to możemy przetestować IMAP i POP3 ręcznie (przykład: `http://en.wikipedia.org/wiki/Post_Office_Protocol`). Dobrze po ostatecznym ustawieniu wszystkiego zrobić testy korzystając z klienta mailowego.

Dovecot - IMAP i POP3

Logowanie przez POP3

```
$ telnet example.com 110
Trying 1.2.3.4...
Connected to example.com.
Escape character is '^]'.
+OK Dovecot ready.
USER user
+OK
PASS pass
+OK Logged in.
LIST
+OK 1 messages:
1 441
.
QUIT
+OK Logging out.
Connection closed by foreign host.
```

SPAM

Jest wiele technik walki ze spamem. Blacklisty są najprostszą, ale skuteczne są techniki badające zawartość maili. Dosyć popularną i współdziałającą z Postfixem aplikacją jest SpamAssasin. Odpowiednie tutoriale są w internecie.

Skupimy się nie na walce z spamem, ale na walce z aplikacjami do walki z spamem. Musimy odpowiednio skonfigurować nasz serwer, by wiadomości przychodzące od niego nie zostały uznane za spam. Jest to skomplikowane, ponieważ technik i standardów jest wiele, a każdy serwer mailowy może używać swoją ich kombinację.

SPAM

Należy zwrócić uwagę między innymi na:

- Poprawne ustawienie rekordu MX w DNS;
- Przedstawianie się serwera z poprawnej domeny w SMTP;
- Rekord SPF TXT w DNS;
- Poprawnie działające reverse DNS.

Ta lista nie jest wyczerpująca, ale powinna wystarczyć w przypadku GMaila, który jest dosyć restrykcyjny pod względem spamu. Warto próbować wysyłać maile na różne posiadane przez siebie skrzynki pocztowe i sprawdzanie czy są one uznane za spam.

Wskazówki od GMaila:
`https://support.google.com/mail/answer/81126`.
Przydają się też bardzo narzędzia online testujące różne aspekty serwera mailowego. Jednym z nich jest kompleksowe narzędzie `http://mxtoolbox.com/domain/`.

SPAM

Jednym z prostszych narzędzi do walki z spamem są rekordy DNS SPF. Służą one do określenia jakiego rodzaju maili mozna oczekiwać od danej domeny.

Rekordy SPF to rekordy DNS typu TXT. W przypadku bind9 definiujemy je standardowo w naszym zone.

Rekordy SPF w bind9

```
; serwery podane w MX, A sa ok, reszta to spam
example.com. IN TXT "v=spf1 mx a -all"
; serwery podane w MX, A oraz z subdomen sa ok,
    reszta jest podejrzana
example.com. IN TXT "v=spf1 mx a ptr ~all"
```

SPAM

Istnieją dobre generatory online rekordów SPF. Jednym z nich jest `http://www.spfwizard.net`.
Dodatkowe informacje o SPF:
`http://www.zytrax.com/books/dns/ch9/spf.html`.

Tworząc serwer mailowy należy też dobrze sprawdzić czy serwer się poprawnie przedstawia w protokole SMTP (np. telnetem). Należy też sprawdzić, czy domena podana w certyfikacie (CN) zgadza się z domeną używaną w mailu. Jeżeli tak nie jest wiadomości mogą zostać uznane za spam i klienty mailowe mogą pokazywać ostrzeżenia.

Dobrze jest też ustawić reverse DNS na domenę maila. Tę usługę musi udostępnić dostawca serwera.

iptables

Podstawowym firewallem na Debianie jest iptables. Domyślnie przepuszcza cały ruch. Jeżeli zauważymy, że IP 4.3.2.1 robi nam coś niedobrego możemy po prostu odrzucać wszystkie pakiety pochodzące od niego.

Konfiguracja iptables do odrzucania pakietów od 4.3.2.1

```
$ iptables -I INPUT -s 4.3.2.1 -j DROP
```

Możemy również zabezpieczać się nie pozwalając na ruch na zewnątrz serwera. iptables ma dużo opcji, o których można poczytać w internecie. Więcej informacji o firewallu na Debianie: `https://wiki.debian.org/DebianFirewall`.

iptables

Problemem z iptables jest to, że nie zapisuje on automatycznie konfiguracji na dysk.

Ręczny zapis/odczyt iptables dla IPv4 na/z dysku

```
$ iptables-save > /etc/iptables/rules.v4
$ iptables-restore < /etc/iptables/rules.v4
```

Po każdej zmianie iptables trzeba je zapisać na dysk. Istnieje paczka, która ładuje te reguły przy starcie systemu.

Instalacja iptables-persistent i zapis iptables w dobre miejsce

```
$ apt-get install iptables-persistent
$ service iptables-persistent save
```

Backup

Musimy dobrze zabezpieczyć się przed utratą danych. Awarie następują właśnie wtedy, gdy się nie zabezpieczamy.

Aby backupy miały sens, muszą być regularne, automatyczne i działające. To znaczy w szczególności, że musimy koniecznie przetestować, czy jesteśmy w stanie odtworzyć backup.

Aby backupy były bezpieczne muszą istnieć w odrębnej lokalizacji od serwera. Posiadanie backupa na tej samej maszynie jest złym pomysłem.

Backup

Istnieje kilka rodzajów backupów, każdy mający inne własności:

- Snapshot - możemy zrobić snapshot naszego VPS. Skuteczne, ale mało przenośne i zajmujące bardzo dużo miejsca. Dobrzy dostawcy VPS często oferują takie usługi;
- Kopia plików - możemy wybrać podzbiór plików do backupu i regularnie robić kopie do bezpiecznej lokalizacji. Wymaga ostrożności w doborze plików, ale zajmuje mniej miejsca i jest bardziej przenośne;
- Inkrementalna kopia plików - wersja kopiowania plików, w której przechowywane są tylko różnice pomiędzy plikami. Bardzo dobre do oszczędzania miejsca przy regularnych backupach, ale mniej odporne na błędy.

Ostatnia opcja pozwala na przechowywanie najdłuższej historii backupów.

Backup

Skupimy się na inkrementalnej kopii wybranych plików. Istnieje wiele narzędzi do backupów, ale czego wybierzemy `rdiff-backup`. Dokumentacja na `http://www.nongnu.org/rdiff-backup/`.

Instalacja `rdiff-backup`

```
$ apt-get install rdiff-backup
```

`rdiff-backup` robi kopię wybranej części systemu plików do katalogu z backupem mającym kopię 1:1 najnowszych wersji plików z dodatkowym katalogiem z metadanymi (diffy). Dzięki temu nasz najnowszy backup można przeglądać bez żadnych narzędzi oraz jest bardziej odporny na problemy.

Backup

Backup powinien być dostosowany do konkretnego zastosowania serwera. Stwórzmy następujący:

- W `/backup/filelist.txt` będzie lista plików do backupu;
- Backup robimy do folderu `/backup`, który jest albo podmontowanym dyskiem sieciowym albo lokalizacją regularnie synchronizowaną z czymś;
- Robimy również backup naszej bazy danych MySQL.

Backup bazy danych robimy z użytkownika posiadającego read-only do wszystkich backupowanych baz danych. Przed przywróceniem takiego backupu należy stworzyć brakujących użytkowników w bazie danych.

Backup

Szkielet /backup/backup.sh

```
#!/bin/bash
mysqldump -ubackupdbuser -ppassword --databases \
    dbname1 dbname2 > /backup/db-backup.sql
rdiff-backup -v5 --include-globbing-filelist \
    /backup/filelist.txt --exclude / \
    / /backup/rdiff-backup/
rm /backup/db-backup.sql
```

Komenda `mysqldump` tworzy plik SQL do utworzenia kopii baz danych `dbname1`, `dbname2`.
Następnie tworzona jest kopia wszystkich plików i katalogów z listy w `/backup/filelist.txt`.
Lista ta zawiera właśnie utworzony `/backup/db-backup.sql`, więc jest on usuwany.

Backup

Szkielet /backup/restore.sh

```
#!/bin/bash
rdiff-backup -r now /backup/rdiff-backup \
    /restored-backup
cp -pr /restored-backup/* /
rm -rf /restored-backup
cat /backup/db-backup.sql | mysql -p
```

`rdiff-backup` ma problemy z nadpisaniem całego systemu plików, więc najpierw rozpakowujemy najnowszy backup do `/restored-backup`, potem kopiujemy wszystko z atrybutami do `/`, a na koniec usuwamy tymczasowy katalog. Aplikujemy także backup bazy danych przez użytkownika root. Musimy pamiętać o dołożeniu jakiegoś skryptu tworzącego brakujących użytkowników bazy danych.

Backup

Przykładowy /backup/filelist.txt

```
- **.swo
- **.swp
/etc/aliases
/etc/phpmyadmin/apache.conf
/etc/apache2/apache2.conf
/etc/apache2/sites-available/
/etc/apache2/ssl
/etc/bind/named.conf.local
/etc/bind/zones/
/etc/postfix/main.cf
/etc/postfix/virtual
/etc/apt/sources.list.d/
/etc/apt/apt.conf.d/50unattended-upgrades
/etc/apt/apt.conf.d/02periodic
/etc/dovecot/conf.d/10-auth.conf
/etc/dovecot/conf.d/10-master.conf
/etc/dovecot/dovecot.conf
/etc/iptables/
/var/www/
/root/scripts/
/root/.bashrc
/root/.ssh/
```

Tworzymy listę plików do backupu od dołu dodając i odejmując (minus) katalogi i konkretne pliki. Obsługiwane są wildcardy.

Automatyczne aktualizacje

Należy dbać o regularną aktualizację oprogramowania. W Debianie stable nie służy to do posiadania najnowszych funkcjonalności, ale do łatania dziur wpływających na bezpieczeństwo aplikacji.

Większość aktualizacji nie potrzebuje żadnej informacji od użytkownika, ale są też takie, które wymagają interakcji z użytkownikiem. Aby te nie wymagające interakcji instalowały się automatycznie, użyjemy paczki `unattended-upgrades`. Jest ona domyślnie zainstalowana, wymaga tylko konfiguracji w katalogu `/etc/apt/apt.conf.d`.

Więcej informacji o `unattended-upgrades`: `https://wiki.debian.org/UnattendedUpgrades`.

Automatyczne aktualizacje

Musimy utworzyć plik konfiguracyjny `02periodic` aktywujący aktualizacje.

/etc/apt/apt.conf.d/02periodic

```
// zalaczone
PT::Periodic::Enable "1";
// update paczek codziennie
APT::Periodic::Update-Package-Lists "1";
// apt-get upgrade --download-only codziennie
APT::Periodic::Download-Upgradeable-Packages "1";
// uruchamianie skryptu aktualizujacego codziennie
APT::Periodic::Unattended-Upgrade "1";
// apt-get autoclean co tydzien
APT::Periodic::AutocleanInterval "7";
// 0 w powyzszych oznacza wylaczenie
```

Automatyczne aktualizacje

Powinniśmy również skonfigurować wysyłanie maila (np. do roota) w razie problemów odkomentowując odpowiednie opcje.

Kawałek `/etc/apt/apt.conf.d/50unattended-upgrades`

```
// Mail do powiadamiania
Unattended-Upgrade::Mail "root";
// Mail tylko w razie bledow
Unattended-Upgrade::MailOnlyOnError "true";
```

Automatyczne aktualizacje

Logi z działania `unattended-upgrades` znajdują się w `/var/log/unattended-upgrades`.

Raz na jakiś czas możemy ręcznie zaaktualizować system dla paczek wymagających interakcji od nas.

Ręczna aktualizacja

```
$ apt-get update
$ apt-get upgrade
```

Możemy również użyć bardziej agresywnego `apt-get dist-upgrade`, który w przeciwieństwie do `apt-get upgrade` może czasem usunąć mniej ważne paczki, by zaaktualizować inne.

Monitorowanie

Warto regularnie monitorować logi. Wiemy, że powinniśmy kogoś zbanować, gdy zauważymy powtarzalną niebezpieczną akcję w logach wykonywaną z danego IP.

Na przykład w `/var/log/apache2` zauważymy gdy ktoś będzie bez przerwy kilka(dziesiąt) razy na sekundę odświeżał coś kosztownego obliczeniowo (o ile to nie jest oczekiwane). Albo w `/var/log/mail.log`, że ktoś próbuje się nieustannie niepoprawnie zalogować na nasz serwer mailowy.

Większość programów ma swoje opcje do ustawienia poziomu logowania (np. apache2 ma `LogLevel`), a czasem i ścieżki (`ErrorLog`). Warto je ustawić na bardziej debugowe wartości na początku.

Monitorowanie

Trzeba pamiętać, że mimo starań nasz serwer może mieć lukę w bezpieczeństwie, może stać się celem ataku DoS albo po prostu mogą wystąpić problemy u dostawcy. Warto zatem posiadać osobny serwer, który będzie sprawdzał czy inne serwery działają poprawnie.

Najprostszą wersją byłoby umieszczenie na nim skryptów pingujących inne serwery i wysyłającym maila w razie problemów (lub, gdy mamy odpowiedni sprzęt, nawet SMSa lub dzwoniące do nas). W zależności od konkretnego zastosowania możemy również napisać skrypty monitorujące aspekt stanu serwera.

Monitorowanie

Przykładowe skrypty:

- skrypt sprawdzający stronę www (czy pobiera się z HTTP 200 OK, a może nawet czy wyświetlaja się określona zawartość);
- skrypt sprawdzający czy da się zalogować z zewnątrz do bazy danych;
- skrypt wysyłający testowego maila z serwera mailowego i sprawdzającego czy doszedł po jakimś czasie;
- skrypt na samym serwerze przeglądający logi w poszukiwaniu błędów lub określonych wzorców;
- inne skrypty wykonujące proste czynności na monitorowanych usługach.

Monitorowanie

Istnieją darmowe serwisy online, które mogą pomóc nam w monitorowaniu niektórych aspektów naszego serwera. Jednym z takich serwisów jest `http://uptimerobot.com`. Pozwala on na posiadanie monitorów pingujących, sprawdzających określone porty, sprawdzających czy strony www się ładują i sprawdzających czy zawierają daną zawartość.

Przy bardziej zaawansowanych zastosowaniach potrzebny jest inny serwer z skryptami. Możemy je dość szybko napisać w bashu i podpiąć pod crona, by się regularnie wykonywały. Istotne jest, by serwer monitorujący był stabilny i w niezależnej lokalizacji. Jeżeli będzie on na tej samej fizycznej maszynie/datacenter co monitorowany serwer to nie otrzymamy maila gdy oba padną. Podobnie, jeżeli będzie on np. na naszym laptopie to możemy uzyskać wiele fałszywych alarmów gdy będziemy tracili/odzyskiwali łączność z internetem.