

# 77

## zadań dla adminów

Autor: Jakub 'unknow' Mrugalski  
[mrugalski.pl](http://mrugalski.pl)

## Wstęp

Mam na imię Kuba i jestem twórcą platformy [MIKR.US](https://mkr.us), zrzeszającej pasjonatów systemu Linux. Na co dzień mam styczność z początkującymi użytkownikami Linuksa, którzy stawiają swoje pierwsze kroki z serwerami.

Wielokrotnie byłem pytany o to, co zrobić, aby łatwiej wejść w linuksowy świat i lepiej zrozumieć jak działają podstawowe mechanizmy tego systemu. Na potrzeby kilku moich znajomych przygotowywałem mini zbiory zadań z Dockera, Ansible, zarządzania plikami, konfiguracji Nginxa, czy Apache. Widząc jak radzili sobie oni z otrzymanymi zadaniami i obserwując ich rozwój, postanowiłem przygotować ebooka, którego właśnie czytasz, licząc na to, że i dla Ciebie takie zbiory zadań będą użyteczne i zaowocują zwiększeniem wiedzy.

Ebook “77 zadań dla adminów” to jak nazwa wskazuje, kompilacja raczej niepowiązanych ze sobą zadań, które mają nakłonić Cię do rzucenia okiem na zadania na pierwszy rzut banalnie łatwe, ale dające przy okazji ich wykonywania dodatkową wiedzę.

Pamiętaj, że od samego czytania nie zdobywa się praktycznych umiejętności. Załatw sobie jakiegoś VPSa, czy uruchom Linuksa na VirtualBoxie, czy innym typie wirtualizacji jaki lubisz i wykonaj zebrane tutaj zadania w praktyce. Nawet jeśli przez myśl przejdzie Ci stwierdzenie “za łatwe!” to i tak to zrób. Albo utrwalisz w ten sposób swoją wiedzę, albo wykonując zadanie dojdiesz do wniosku, że tylko wydawało Ci się, że wiesz, jak to zrobić.

Pamiętaj, że Ebook przeznaczony jest dla osób początkujących, więc zebrane tutaj podpowiedzi niekoniecznie są najbardziej efektywne - są po prostu najłatwiejsze.

Życzę miłej lektury  
[Jakub 'unknow' Mrugalski](#)

## Zarządzanie użytkownikami

Oto lista zadań, które sprawią, że lepiej zrozumiesz ideę zarządzania użytkownikami w systemie Linux.

1. Załóż jednym poleceniem (nie interaktywnie!) użytkownika o nazwie 'Marian'. Spraw, aby jego główną grupą użytkownika była 'zarzad', a grupy pomocnicze to 'adm' oraz 'audio'. Marian ma mieć powłokę ustawioną na /bin/sh.

**Podpowiedź:** `man useradd`

2. Spraw, aby każdy nowo założony użytkownik w systemie miał w swoim katalogu plik 'regulamin.txt' z jakimś krótkim tekstem w środku.

**Podpowiedź:** odpowiedni parametr do `useradd` + katalog tzw. szkieletu (`skel`)

3. Zmień użytkownikowi Marian powłokę na /usr/bin/nano (to nie jest poprawny shell! Musisz to jakoś obejść). Gdy Ci się uda, zmień mu powłokę na /bin/bash, aby dało się na tym użytkowniku nadal pracować.

**Podpowiedź:** narzędzie do zmiany shella (jedno polecenie) + lista dozwolonych powłok w `/etc/`

4. Ustaw użytkownikowi 'Marian' hasło 'banany69', ale w sposób nieinteraktywny, za pomocą jednego polecenia.

**Podpowiedź:** `echo` + polecenie do zmiany hasła (ale nie `passwd`, bo ono jest interaktywne!).

5. Wykonaj polecenie 'whoami' jako Marian, ale będąc zalogowanym jako root.

**Podpowiedź:** `sudo`

6. Zmień (jednym poleceniem) podstawową grupę użytkownika 'Marian' na 'pracownicy'. Jeśli jest taka potrzeba, to przy okazji utwórz taką grupę.

**Podpowiedź:** `usermod`

7. Wymuś na użytkowniku 'Marian' regularną zmianę hasła co 7 dni (jednym poleceniem).

**Podpowiedź:** chage

8. Zablokuj konto 'Marian' tak, aby nie dało się na niego zalogować (nie chodzi o zmianę hasła, ani usunięcie konta), a później ponownie odblokuj tego użytkownika.

**Podpowiedź:** passwd lub chage

9. Spraw, aby nowo zakładani użytkownicy mieli automatycznie przypisywane numery ID od 8000 w górę.

**Podpowiedź:** /etc/login.defs

10. Spraw, aby Marian po zalogowaniu się na swoje konto zobaczył aktualny uptime i load na serwerze (zakładamy, że Marian używa basha)

**Podpowiedź:** polecenie uptime + pliki ~/.bash\_profile lub ~/.bashrc (poczytaj o różnicach!).

11. Spraw, aby użytkownik 'Marian' mógł uruchomić maksymalnie 5 procesów jednocześnie.

**Podpowiedź:** /etc/security/limits.conf

Chcesz lepiej zrozumieć zarządzanie użytkownikami i nie tylko? Rzuć okiem na kurs [“Zarządzanie Linuksem w godzinę”](#)

## Zarządzanie plikami

Wykonanie poniższej listy zadań sprawi, że lepiej poznasz zasadę zarządzania plikami w Linuksie.

**Założenia wstępne:** w systemie posiadasz tylko trzech użytkowników:

- **Marian** - jest w grupie 'uzytkownicy'
- **Ania** - jest w grupie 'uzytkownicy' oraz 'kontraktorzy'
- **Stefan** - jest w grupie 'praktykanci'

1. Załóż pusty plik o nazwie 'raport.txt'. Ania i Marian mogą czytać jego zawartość. Marian może także pisać do niego, a Stefan nie ma do niego dostępu.

**Podpowiedź:** `chmod + chown`

2. Spraw, aby do pliku 'logi.txt' dało się tylko dopisywać nowe dane na końcu i aby mogła robić to tylko Ania z Marianem.

**Podpowiedź:** `chmod + chown + chattr`

3. Plik o nazwie 'config.yaml' ma być zdatny do odczytania przez wszystkich użytkowników, ale NIKT nie może go usunąć. Nawet jego właściciel.

**Podpowiedź:** `chattr`

4. Wszyscy użytkownicy mogą zakładać pliki w katalogu /tmp/dane/, ale pomimo tego, że każdy ma tam dostęp, to tylko właścicielom wolno usuwać swoje pliki i zmieniać ich nazwy (i to nawet gdyby plik miał `chmod 777!`).

**Podpowiedź:** sticky bit

5. Spraw, aby pliki "pierwszy.txt" oraz "drugi.txt" miały dokładnie taką samą zawartość, a zmiana tekstu w jednym automatycznie powodowała zmianę tekstu w drugim z plików. Zmiana praw dostępu (chmod) do jednego z plików automatycznie zmienia te same prawa na drugim pliku (widoczne przez ls -l).

**Podpowiedź:** komenda 'ln'

6. Stwórz plik, który ma dokładnie 100MB. Zrób to jednym poleceniem.

**Podpowiedź:** polecenie 'truncate' lub 'dd' (w ramach nauki wypróbuj oba)

7. Utwórz plik 'starocie.txt' w taki sposób, aby polecenie 'ls -l' pokazywało, że został on założony 1 stycznia 1999 roku.

**Podpowiedź:** touch

8. W katalogu masz kilkaset plików z rozszerzeniem TXT. Zmień każdemu z nich rozszerzenie na DOC. Zrób to jednym poleceniem, bez pętli.

**Podpowiedź:** rename

9. Jesteś w katalogu, w którym są tysiące plików. Usuń te, których rozmiar jest większy niż 1MB, a nazwa zaczyna się na literę 'b'. Zrób to jednym poleceniem.

**Podpowiedź:** find

10. W katalogu masz tysiące plików. Musisz usunąć wszystkie, które NIE zawierają litery 'x' w nazwie. Nie używaj do tego pętli (for/while).

**Podpowiedź:** find z negacją (prostsze) / ls + grep z xargs

11. W katalogu masz dziesiątki plików. Stwórz paczkę 'paczka.tgz' (tar+gzip) zawierającą jedynie pliki większe niż 1MB i do tego mające rozszerzenie txt.

**Podpowiedź:** tar + find (być może trzeba będzie użyć subshella lub pipe)

Jeśli masz problemy z poruszaniem się w linuksowym terminalu, to [rzuć okiem na mój film](#), w którym tłumaczę podstawy.

## Serwer Apache

Do testowania serwera webowego przyda Ci się polecenie 'CURL' (apt install curl). Chcąc odpytać lokalny serwer o domenę np. mrugalski.pl, zrobisz to w następujący sposób:

```
curl -H 'Host: mrugalski.pl' http://localhost
```

1. Stwórz dwa hosty wirtualne. Jeden o nazwie "pierwszy.pl", a drugi o nazwie "drugi.pl". Pod każdym umieść oddzielny plik index.html tak, aby dało się zorientować, który host został wywołany. Za każdym razem, gdy ktoś odpyta o nieistniejący host (np. 'trzeci.pl'), otwierać ma się zawartość 'drugi.pl'.

**Podpowiedź:** site-enabled + site-available + domyślny vhost (jak się go ustawia?)

2. Spraw, aby domyślnym plikiem startowym strony 'drugi.pl' był 'start.html' zamiast 'index.html'. Dla wszystkich pozostałych domen plikiem startowym ma być 'cokolwiek.html'.

**Podpowiedź:** dyrektywa 'DirectoryIndex' na poziomie vhosta + rekonfiguracja modułu 'dir'

3. Spraw, aby otworzenie adresu "drugi.pl/abc" powodowało przekierowanie użytkownika na stronę [mikr.us](http://mikr.us).

**Podpowiedź:** aktywny mod\_rewrite + odpowiednia reguła

4. Spraw, aby serwer do każdej swojej odpowiedzi doklejał nagłówek 'X-Test: siema!'.

**Podpowiedź:** aktywny mod\_headers + odpowiednia konfiguracja

5. Spraw, aby Twój Apache przedstawiał się jako Nginx w nagłówku 'server'.

**Podpowiedź:** mod\_header

6. Zablokuj użytkownikom pobieranie plików z rozszerzeniem JPG w ramach hosta 'drugi.pl' (pozostałe mają działać).

**Podpowiedź:** FilesMatch + operator 'require all ...' lub 'deny from ...'

7. Serwer 'pierwszy.pl' ma obsługiwać wszystkie możliwe subdomeny w ramach swojej domeny, czyli np: 'test01.pierwszy.pl', 'test02.pierwszy.pl', czy nawet 'losowy-ciag-znakow.pierwszy.pl'.

**Podpowiedź:** ServerName + ServerAlias

8. Zablokuj ludziom z adresów IP 123.123.\*.\* wchodzić na stronę 'drugi.pl'.

**Podpowiedź:** 'Deny from ...' połączone z Location lub Directory w ramach vhosta

9. Sprawdź, aby każde odwołanie do nieistniejącego pliku (error 404) wyświetlało komunikat "Ojeeej... pliku nie ma!".

**Podpowiedź:** ErrorDocument + ścieżka do pliku z komunikatem lub sam komunikat

10. Twój Apache przy domyślnej konfiguracji, prawdopodobnie na starcie uruchamia 5 swoich procesów. Sprawdź, aby na starcie uruchomił ich 10, ale w razie dużego ruchu nigdy nie uruchamiał więcej niż 20 instancji.

**Podpowiedź:** konfiguracja MPM (moduły)

11. Sprawdź, jak długo uruchomiony jest Apache, ale korzystając jedynie z narzędzia CURL. Metoda ta zwróci Ci wiele innych, ciekawych statystyk, ale działa tylko na Twoim serwerze (nie da się jej użyć 'z internetu').

**Podpowiedź:** curl + mod\_status + przyda się przełącznik '?auto', ale to tylko dla poprawienia czytelności

Chcesz lepiej poznać Apache? Rzuć okiem na kurs "[Apache w godzinę](#)"



## Serwer Nginx

Do testowania serwera webowego przyda Ci się polecenie 'CURL' (apt install curl). Chcąc odpytać lokalny serwer o domenę np. mrugalski.pl, zrobisz to w następujący sposób:

```
curl -H 'Host: mrugalski.pl' http://localhost
```

1. Stwórz trzy serwery webowe, obsługujące kolejno domeny 'pierwszy.pl', 'drugi.pl' oraz 'trzeci.pl'. Każda z domen ma odpowiadać innym tekstem (aby dało się je rozróżnić), a odpytanie domeny nieistniejącej ma wyświetlać zawartość serwera 'drugi.pl'

**Podpowiedź:** sites-available + sites-enabled + domyślny vhost (parametr 'default')

2. Twoja instancja Nginx ma uruchamiać domyślnie 8 workerów (domyślnie uruchamia mniej, zazwyczaj tyle, ile masz wątków w procesorze).

**Podpowiedź:** nginx.conf

3. Spraw, aby katalog o nazwie 'dane' w ramach domeny 'pierwszy.pl', jeśli nie zawiera pliku 'start.html', był listowany (czyli w przeglądarce wyświetla się spis znajdujących się w nim plików). Jeśli jednak plik 'start.html' istnieje, to ma się wyświetlić jako domyślny.

**Podpowiedź:** konfiguracja na poziomie vhosta + dyrektywa 'index' + autoindex na odpowiednie location

4. Pod adresem 'drugi.pl/test01' powinny ładować się pliki z katalogu /var/www/abc, a pod adresem 'drugi.pl/test02' pliki z katalogu /var/www/xyz. Dla wszystkich pozostałych adresów pliki powinny ładować się z katalogu /var/www/domyslne.

**Podpowiedź:** dyrektywa 'root' połączona z odpowiednim location + globalna definicja 'root'.

5. Spraw, aby wszystkie pliki \*.JPG oraz \*.PNG podawane z domeny 'trzeci.pl', posiadały nagłówek cachujący (Expires: ....) ustawiony na 30 dni od daty wywołania zapytania.

**Podpowiedź:** location z dopasowaniem do wzorca + expires

6. Skonfiguruj serwer tak, aby wszystkie adresy zaczynające się od słowa 'skok' w domenie 'drugi.pl' przekierowywały na stronę [mokr.us](http://mokr.us). Czyli poprawne URLe to np 'drugi.pl/skokXY' lub 'drugi.pl/skokowy', czy 'drugi.pl/skok999' **Podpowiedź:** dyrektywa 'rewrite' połączona z prostym wyrażeniem regularnym
7. Spraw, aby odwołanie do nieistniejącego pliku (error 404) skutkowało wyświetleniem komunikatu o treści 'Pliku nie ma!', a do tego aby strona z błędem miała kod odpowiedzi '200 OK'.

**Podpowiedź:** error\_page + odpowiednie trzy parametry

8. Zezwól na wejście na hosta 'trzeci.pl' jedynie adresom 1.2.3.4 oraz .9.8.7.6. Wszystkie pozostałe mają otrzymać komunikat "Zakaz wstępu!".

**Podpowiedź:** location + 'allow ...' + 'deny ...'

9. Spraw, aby domena 'pierwszy.pl' obsługiwała jedynie zapytania typu GET. Wszystkie pozostałe mają otrzymać komunikat "Błędna metoda!"

**Podpowiedź:** dodaj warunek sprawdzający zmienną \$request\_method. Jeśli nie jest równa 'GET', zwróć dowolny error (np. 'return 405') lub użyj konstrukcji 'limit\_except ...'

10. Dodaj do wszystkich plików, których nazwa zaczyna się na literę 'a', a do tego pochodzą z hosta 'pierwszy.pl', nagłówek 'X-Test: OK'

**Podpowiedź:** add\_header + location z odpowiednią maską RegEx

11. Spraw, aby każdemu otwierającemu hosta 'trzeci.pl' wyświetlała się zawartość portalu wp.pl. To nie ma być przekierowanie, a bezpośrednie podanie zawartości. Dodatkowo podmień wszystkie wystąpienia słowa 'wczoraj' na 'jutro'.

**Podpowiedź:** location + proxy\_pass + nadpisanie odpowiednich nagłówków

Chcesz lepiej poznać Nginx? Rzuć okiem na kurs "[Nginx w godzinę](#)"

## Automatyzacja Ansible

W zadaniach zakładam, że Twój plik “inventory” wygląda jak poniżej, a wszystkie serwery działają pod kontrolą systemu Ubuntu.

```
[web]
  web01.mikr.us
  web02.mikr.us
  web03.mikr.us
[other]
  mysql.mikr.us
  psql.mikr.us
```

1. Wykonaj polecenie ‘uptime’ na wszystkich serwerach z grupy WEB z wyjątkiem web03. Zrób to jednym poleceniem, bez tworzenia playbooka

**Podpowiedź:** wywołanie ad-hoc. Moduł command (domyślny) lub shell. Trzeba użyć wykluczeń w ‘limit’

2. Na serwerze ‘mysql’ odblokuj na firewallu (ufw) ruch na port 3306 ze wszystkich serwerów z grupy WEB

**Podpowiedź:** wymagane jest stworzenie playbooka z pętlą. Adresy IP hostów pobierzesz pluginem ‘lookup’ z parametrem ‘dig’. Moduł do zarządzania UFW nazywa się po prostu ‘ufw’.

3. Załóż katalog “klucze” i wrzuć do niego trzy publiczne klucze SSH (każdy w osobnym pliku). Każdy z nich dodaj do authorized\_keys na wszystkich serwerach z inventory

**Podpowiedź:** moduł authorized\_keys + pętla po serwerach + pętla po kluczach

4. Na każdym serwerze zainstaluj fail2ban. Skonfiguruj go tak, aby nie banował żadnego serwerów z inventory, czyli skonfiguruj odpowiednio liniijkę 'ignoreip' w pliku jail.conf. Po zmianach przeładuj fail2ban, ale spraw aby reload nie oznaczał statusu serwera jako 'changed'

**Podpowiedź:** dwa taski (zmiana configi i reload) + moduł lineinfile + sklejanie nazw serwerów w jednego stringa przez modyfikator 'join' + definicja when\_changed przy zadaniu z reload

5. Przygotuj playbooka gotowego do obsłużenia zarówno Ubuntu jak i systemów z rodziny RedHat. Zainstaluj na każdym serwerze paczkę 'vim'. Dla serwerów z rodziny Debiana użyj 'apt', a dla tych z rodziny RedHata użyj 'yum'

**Podpowiedź:** dodaj argument 'when' i sprawdzaj wartość 'os\_family' w tzw. faktach

6. Załóż na każdym serwerze plik /tmp/host\_XYZ, gdzie XYZ to nazwa domenowa danego serwera. Jako zawartość pliku wstaw tekst 'OK'. Jeśli w katalogu /tmp/ istnieją inne pliki z prefiksem 'host\_', usuń je.

**Podpowiedź:** moduł copy + definicja content + usuwanie wg wzorca (dowolną metodą). Pamiętaj tylko, aby sprzątać przed tworzeniem pliku, a nie po.

7. Na każdym z hostów postaw kontener dockera z Nginx, wystawiając jego port 80 na zewnątrz.

**Podpowiedź:** użyj roli 'docker' (znajdziesz ją w Ansible Galaxy). Po drodze trzeba będzie zadbać o zależności w Pythonie (instalacja modułu 'docker' przez pip)

8. Upewnij się, że na każdym serwerze istnieje grupa 'wheel' (jeśli jej nie ma, to dodaj) i że należy do niej użytkownik 'stefan' (jeśli go nie ma, to załóż). Spraw, aby użytkownicy z grupą wheel mieli prawo używania polecenia 'su'

**Podpowiedź:** moduły user, group, lineinfile. Musisz zmodyfikować plik /etc/sudoers

9. Napisz playbooka, który postawi Wordpresa na aktualnym serwerze dla domeny testuje.pl

**Podpowiedź:** musisz zainstalować Apache + PHP + MySQL i najprościej będzie posłużyć się narzędziem 'wpcli', które za Ciebie ściągnie Wordpresa, rozpakuje go, skonfiguruje i uruchomi instalację

10. Spróbuj przebudować playbooka z zadania numer 9 w taki sposób, aby nie wykorzystywać żadnego wbudowanego modułu Ansible. Wykorzystaj gotowe 'role' do postawienia wcześniej wspomnianej konfiguracji

**Podpowiedź:** gotowe role i opis ich użycia i instalacji znajdziesz na Ansible Galaxy

11. Przygotuj playbooka do zabezpieczenia serwera SSH. Wyłącz możliwość bezpośredniego logowania się na konto 'root'. Zmień domyślny numer portu usługi z 22 na 10022. Skonfiguruj firewalla (UFW) w taki sposób, aby akceptował połączenia na nowy numer portu i odrzucał na stary. Spraw, aby logowanie było możliwe jedynie za pomocą klucza (wyłącz dostęp po hasle). Postaraj się wszystkie zmiany w pliku konfiguracyjnym SSH wprowadzić w jednej pętli (bez wywoływania tylu pasków ile zmian)

**Podpowiedź:** użyjesz tutaj wiedzy zebranej w poprzednich zadaniach oraz modułu lineinfile uruchomionego w pętli. Parametrem do pętli powinien być zbiór par zmiennych - co zmienić i na co zmienić (regex).

Chcesz lepiej poznać Ansible? Rzuć okiem na kurs "[Ansible w godzinę](#)"

## Kontenery Dockera

1. Uruchom kontener z serwerem webowym Nginx w taki sposób, aby domyślna strona startowa podawana była z katalogu /tmp/strona/ (wrzuć tam plik index.html z dowolnym tekstem). Spraw także, aby kontener został automatycznie usunięty po jego wyłączeniu i aby nasłuchiwał na porcie 80 i przyznaj mu maksymalnie 256MB pamięci. Wykonaj to zadanie z linii poleceń, korzystając z tylko jednego polecenia startowego.

**Podpowiedź:** musisz posłużyć się opcjami do montowania katalogów, wystawiania portów (tutaj być może wystarczy samo -P zamiast -p)

2. Wykonaj backup oryginalnego obrazu kontenera z poprzedniego zadanie w taki sposób, aby uzyskać plik TGZ, a następnie postaraj się odtworzyć ten backup.

**Podpowiedź:** poczytaj o opcjach 'save' i 'load'

3. Wykonaj zadanie numer 1 z użyciem pliku Dockerfile.

**Podpowiedź:** [poczytaj w jaki sposób tworzy się taki plik](#)

4. Korzystając z Docker-Compose postaw działającego Wordpressa, składającego się z serwera Apache z obsługą PHP (znajdź gotowy obraz) i serwera MySQL (lub MariaDB jeśli lubisz). Zadbaj, aby w przypadku restartu któregoś z serwerów nie utracić danych należących do Wordpressa.

**Podpowiedź:** [poczytaj o Docker Compose](#)

5. Sprawdź jakie pliki zostały zmienione w kontenerze z MySQL od chwili jego uruchomienia.

**Podpowiedź:** użyteczne będzie polecenie 'diff'

6. Stwórz obraz kontenera uruchamiający skompilowaną aplikację “Hello World” napisaną w GO na kontenerze bazującym na dystrybucji Alpine, na której nie ma kompilatora wspomnianego języka. Postaraj się, aby docelowy obraz miał możliwie niewielki rozmiar. Idealnie byłoby, aby poza czystą dystrybucją znalazła się tam tylko skompilowana aplikacja.

**Podpowiedź:** wykorzystaj tzw. multistage-build. W pierwszym kontenerze zbuduj aplikację, a następnie efekt builda przerzuć (polecenie COPY wraz z odpowiednim ‘from’ do kontenera docelowego)

7. Postaw kontener z Apache2 oraz jeden z Nginx. Następnie obok nich uruchom kontener z ‘Nginx Proxy Manager’. Wykorzystując swoją prawdziwą domenę spraw, aby pod subdomeną ‘apache’ ładowała się zawartość pierwszego kontenera, a pod subdomeną ‘nginx’ tego drugiego. Zadbaj, aby każda ze stron działała po HTTPS. Spraw, aby oba serwery webowe mogły się komunikować po prywatnej sieci z Proxy Managerem

**Podpowiedź:** musisz nauczyć się [jak konfigurować proxy managera](#). Poczytaj także jak tworzy się sieć pomiędzy serwerami w Dockerze. Kwestię certyfikatów SSL proxy manager powinien ogarnąć automatycznie.

8. Wrzuć do działającego kontenera (może to być np. uruchomiony Nginx) plik ‘test.txt’ o dowolnej zawartości, a następnie uruchom powłokę bash/sh/ash na tym samym kontenerze i za pomocą polecenia ‘cat’ podglądnij zawartość wrzuconego pliku

**Podpowiedź:** skorzystaj z poleceń dockera ‘cp’ oraz exec. Przy tym drugim pamiętaj o alokacji interaktywnego terminala, aby widzieć wpisywane polecenia.

9. Przygotuj własny obraz kontenera z Apache, obsługującym język PHP. Możesz wykorzystać dowolny obraz czystego systemu jako bazowy (najlepiej Alpine, ale równie dobrze może to być Debian, czy Ubuntu jeśli nie zależy Ci na optymalizacji rozmiaru). Następnie wyślij ten obraz na DockerHuba.

**Podpowiedź:** musisz założyć konto na DockerHub (darmowe), zalogować się tam przez dockera, zbudować własny obraz, otagować go i wysłać (push) na Huba.

10. Poznaj podstawy korzystania z API Dockera. Używając narzędzia NetCat (apt install nc) pobierz listę obrazów dostępnych na Twoim serwerze. Zwróć ją w formacie JSON.

**Podpowiedź:** komunikację rozpoczniesz poleceniem 'nc -U /var/run/docker.sock'. Poczytaj [jak korzysta się z API](#). Jeśli wolisz, to zamiast NetCata możesz wykorzystać CURLa z parametrem '—unix-socket'.

11. Przygotuj sobie dwie wirtualne maszyny (VPSy) z zainstalowanym dockerm. Na pierwszym z nich uruchom konfigurację z zadania numer 4. Następnie spróbuj tę konfigurację przenieść na drugi serwer metodą zatrzymania kontenerów, synchronizacji plików i podniesienia kontenerów na drugim serwerze. Ważne jest, aby nie utracić żadnych danych należących do Wordpressa (pliki + DB).

**Podpowiedź:** wykorzystaj polecenia start/stop oraz narzędzie rsync. Ewentualna trudność polega tutaj na zlokalizowaniu, gdzie fizycznie znajdują się dane Dockera, jego kontenery i obrazy.

Chcesz lepiej poznać Dockera? Rzuć okiem na kurs "[Docker w godzinę](#)"



## ZAKOŃCZENIE

Wykonanie zebranych tutaj zadań nie zmieni Cię w eksperta - na to potrzeba czasu - ale z pewnością posunie Cię naprzód na Twojej drodze do zostania lepszym adminem, czy devopsem.

Najlepszą metodą na zdobywanie wiedzy poza praktycznym wykonywaniem zadań jest... obracanie się w towarzystwie osób, które już taką wiedzę posiadają i chcą się nią dzielić. Jeśli masz już za sobą swoją pierwszą pracę, to z pewnością wiesz o czym mówię.

Nie zawsze jednak znalezienie odpowiedniego otoczenia sprzyjającego rozwojowi jest zadaniem łatwym, dlatego dołożyłem wszelkich starań, aby takie miejsce dla początkujących stworzyć.

Dołącz do grupy “**Mikrusy**”. Jeśli masz konto na Facebooku, to zapraszam:

<https://www.facebook.com/groups/mikrusy>

Jeśli preferujesz nieco mniej gadatliwego w naszym przypadku Discorda, to tutaj znajdziesz zaproszenie:

<https://discord.gg/hFcqJGkppq>

Życzę sukcesów w Twoim dalszym rozwoju i do zobaczenia w przyszłości 🙌

[Jakub 'unknow' Mrugalski](#)