DEVOPS
DLA ZDESPEROWANYCH

PRAKTYCZNY PORADNIK PRZETRWANIA

BRADLEY SMITH

Helion

no starch press

Tytuł oryginału: DevOps for the Desperate: A Hands-On Survival Guide

Tłumaczenie: Robert Górczyński

ISBN: 978-83-289-1126-0

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

Helion S.A.
ul. Kościuszki 1c, 44-100 Gliwice
tel. 32 230 98 63
e-mail: *helion@helion.pl*
WWW: *https://helion.pl* (księgarnia internetowa, katalog książek)

Printed in Poland.

Spis treści

Część III
OBSERWOWALNOŚĆ I ROZWIĄZYWANIE PROBLEMÓW

2

Używanie Ansible do zarządzania hasłami, użytkownikami i grupami

Po przygotowaniu maszyny wirtualnej możesz przystąpić do wykonywania zadań administracyjnych, takich jak zarządzanie użytkownikami. Automatyzacja praktyk DevOps ma znaczenie kluczowe podczas tworzenia zasobów i zarządzania nimi. Aby zarządzać jakimkolwiek hostem Linuksa, musisz mieć podstawy wiedzy z zakresu działania haseł, użytkowników i grup. Użytkowników i grupy można uznać za najważniejsze elementy konstrukcyjne w dziedzinie zarządzania tożsamością, podczas gdy grupy pozwalają na zarządzanie zbiorem użytkowników oraz kontrolowanie dostępu do plików, katalogów i poleceń. Podział odpowiedzialności między użytkowników i grupy może mieć ogromne znaczenie i chronić przed nieupoważnionym dostępem do zasobów.

W rozdziale będę kontynuować omawianie sposobu pracy z oprogramowaniem Ansible. Usprawnisz również utworzoną wcześniej maszynę wirtualną, aby wzmocnić w niej podstawową politykę zapewnienia bezpieczeństwa. Wybrane z dostarczanych standardowo zadań Ansible wykorzystasz do wymuszenia stosowania silnych haseł, zarządzania użytkownikami i grupami, a także do kontrolowania dostępu do współdzielonego katalogu bądź pliku. Gdy opanujesz te podstawy dotyczące zapewnienia bezpieczeństwa, możesz je później stosować jako punkt wyjścia w każdym scenariuszu.

Wymuszenie stosowania silnych haseł

Pozostawienie użytkownikom decyzji o tym, co można uznać za silne hasło, jest drogą prowadzącą wprost do katastrofy. Dlatego też konieczne jest wymuszenie stosowania silnych haseł we wszystkich hostach, do których użytkownicy mogą mieć dostęp. Skoro automatyzacja jest jedną z nadrzędnych reguł, kod można wykorzystać do wymuszenia, aby każdy użytkownik stosował silne hasło. W tym celu można użyć zadania Ansible do zainstalowania wtyczki dla mechanizmu tzw. **dołączalnych modułów uwierzytelniania** (ang. *pluggable authentication module*, PAM). Ten mechanizm ma po prostu postać frameworka uwierzytelniania wykorzystywanego w większości dystrybucji systemów Linux. Wtyczka zapewniająca obsługę silnych haseł nosi nazwę `pam_pwquality`. Ten moduł przeprowadza weryfikację haseł na podstawie zdefiniowanych kryteriów.

Instalowanie narzędzia libpam-pwquality

Moduł PAM `pwquality` jest w repozytorium oprogramowania Ubuntu dostępny pod nazwą `libpam-pwquality`. Zadania Ansible znajdujące się w materiałach przygotowanych do książki pozwalają na zainstalowanie i skonfigurowanie tego pakietu. Pamiętaj, że ostatecznym celem jest automatyzacja wszystkiego, co tylko możliwe, a wspomniane zadanie zapewnia mechanizm wykonywania operacji administracyjnych. Te zadania są zdefiniowane w repozytorium, które zostało sklonowane (patrz „Wprowadzenie”). Przejdź do katalogu *ansible/chapter2/*, a następnie w ulubionym edytorze tekstu otwórz plik *pam_pwquality.yml*. Ten plik zawiera dwa zadania: `Install libpam-pwquality` i `Configure pam_pwquality`.

Skoncentrujmy się na pierwszym zadaniu, w którym moduł `package` Ansible został użyty do zainstalowania `libpam-pwquality` w maszynie wirtualnej. Na początku pliku znajduje się kod zadania instalującego wymieniony pakiet i przedstawia się on następująco:

```
---
- name: Install libpam-pwquality
  package:
    name: "libpam-pwquality"
    state: present
--cięcie--
```

Każde zadanie Ansible powinno rozpoczynać się od deklaracji `name` definiującej cel. W omawianym przykładzie celem jest zainstalowanie pakietu `libpam-pwquality`, więc element `name` ma wartość `Install libpam-pwquality`. Następnie moduł `package` w Ansible przeprowadza właściwą operację instalacji oprogramowania. Moduł `package` wymaga zdefiniowania dwóch parametrów: `name` i `state`. W tym zadaniu nazwa pakietu (znajduje się on w repozytorium Ubuntu) to `libpam-pwquality`, natomiast jego stan powinien być określony jako `present`. W celu usunięcia pakietu należy jego stan zmienić na **`absent`**. To jest dobry przykład polecenia deklaracyjnego, ponieważ nakazujesz Ansible upewnienie się o zainstalowaniu wskazanego pakietu. Nie musisz się przejmować tym, jak pakiet będzie zainstalowany, o ile tak się stanie. Jeżeli zainstalujesz pakiet (`present`), a następnie usuniesz zadanie z Ansible, wówczas ten pakiet wciąż będzie instalowany w trakcie następnej operacji przygotowania zasobu. Konieczne jest wyraźne wskazanie stanu pakietu jako `absent`, aby host odzwierciedlał żądany stan.

Jak już wspomniałem w rozdziale 1., moduły Ansible (takie jak wymieniony) przeprowadzają pewne zadania w systemie operacyjnym, np. włączenie zapory sieciowej, zarządzanie użytkownikami lub (jak w omawianym przykładzie) instalowanie oprogramowania. Ansible pozwala, aby działania były *idempotentne*, co oznacza, że dane działanie można wykonywać wielokrotnie, a ostateczny wynik zawsze będzie taki sam jak podczas poprzedniej operacji wykonania danego działania. Dlatego też należy automatyzować wszystko, co tylko możliwe! Możesz zaoszczędzić dużo czasu i uniknąć pomyłek popełnianych w wyniku ręcznego wykonywania zadań. Wyobraź sobie, że musisz w ciągu dnia skonfigurować 1000 komputerów. Bez wykorzystania automatyzacji to byłoby praktycznie niemożliwe.

Konfiguracja modułu pam_pwquality w celu wymuszenia silniejszej polityki haseł

W domyślnym systemie Ubuntu hasła nie są tak silne, jak mogłyby być. Minimalna wymagana długość hasła to sześć znaków, a ponadto przeprowadzane są jedynie najprostsze operacje jego sprawdzenia. Aby wymusić stosowanie znacznie silniejszych haseł, konieczne będzie skonfigurowanie `pam_pwquality` do wymuszenia polityki silniejszych haseł.

Za konfigurację modułu `pam_pwquality` odpowiada plik o nazwie */etc/pam.d/common-*
↪password. To w tym pliku zadanie Ansible wprowadzi niezbędne zmiany reguł weryfikacji haseł. Tak naprawdę musisz zmienić tylko jeden wiersz tego pliku. Najczęściej używany sposób edytowania wiersza przez Ansible polega na wykorzystaniu modułu `lineinfile`, który pozwala zmienić wiersz w pliku bądź sprawdzić, czy dany wiersz istnieje.

Mając w edytorze tekstu otwarty plik zadania Ansible `pam_pwquality`, przyjrzyj się dokładniej drugiemu zadaniu, które powinno przedstawiać się następująco:

```
--cięcie--
- name: Configure pam_pwquality
  lineinfile:
    path: "/etc/pam.d/common-password"
```

```
    regexp: "pam_pwquality.so"

    line: "password required pam_pwquality.so minlen=12 lcredit=-1 ucredit=-1
           dcredit=-1 ocredit=-1 retry=3 enforce_for_root"
    state: present
--cięcie--
```

Także to zadanie rozpoczyna się od nazwy, `Configure pam_pwquality`, wskazującej jego przeznaczenie. Następnie nakazywane jest oprogramowaniu Ansible użycie modułu `lineinfile` do przeprowadzenia edycji pliku hasła PAM. Moduł `lineinfile` wymaga ścieżki dostępu do pliku (`path`), w którym mają być wprowadzone zmiany. W omawianym przykładzie tym plikiem jest */etc/pam.d/common-password*. Przy użyciu wyrażeń regularnych (`regexp`) następuje odszukanie wiersza przeznaczonego do zmiany. Tutaj wyrażenie regularne odszukuje wiersz zawierający `pam_pwquality.so` i zastępuje go nowym wierszem. Zamiennik (`line`) zawiera zmianę konfiguracyjną (`pwquality`), która wymusza stosowanie silniejszych haseł. Użyte tutaj opcje będą wymuszały następujące polityki dotyczące hasła:

- minimalna długość hasła wynosi 12 znaków,
- hasło musi zawierać co najmniej jedną małą literę,
- hasło musi zawierać co najmniej jedną wielką literę,
- hasło musi zawierać co najmniej jedną cyfrę,
- hasło musi zawierać co najmniej jeden znak inny niż alfanumeryczny,
- dozwolone są trzy próby wpisania hasła,
- wyłączona jest możliwość zmiany hasła przez użytkownika `root`.

Dodanie tych wymagań powoduje znaczne wzmocnienie domyślnej polityki stosowania haseł w Ubuntu. Każde nowe hasło będzie musiało spełnić te wymagania, co powoduje, że potencjalnym atakującym znacznie trudniej będzie przeprowadzać ataki typu *brute force*.

Uwaga

*Wartości ujemne w przedstawionym w kodzie nowym wierszu konfiguracyjnym wskazują modułowi `pam_pwquality`, że hasło musi mieć przynajmniej „jeden znak" z danej kategorii. Więcej informacji na ten temat znajdziesz na stronie podręcznika systemowego dla tego modułu (zostanie wyświetlony po wydaniu polecenia **`man pam_pwquality`**).*

Zamknij plik *pam_pwquality.yml*, co pozwoli przejść do tworzenia użytkowników za pomocą modułu Ansible.

Rodzaje użytkowników w systemie Linux

W przypadku systemu Linux mamy trzy rodzaje użytkowników: zwykły, systemowy i tzw. *root*. **Zwykły użytkownik** to po prostu konto przeznaczone dla człowieka, użytkownika korzystającego z komputera. Za chwilę zajmiemy się utworzeniem takiego konta. Każdy zwykły użytkownik jest najczęściej powiązany z hasłem, grupą i nazwą użytkownika. Z kolei **użytkownik systemowy** to konto nieprzeznaczone dla człowieka, ale dla użytkownika wymaganego do działania np. serwera Nginx. Tak naprawdę użytkownik systemowy jest praktycznie identyczny ze zwykłym użytkownikiem, przy czym jego identyfikator (ang. *user id*, UID) z różnych względów będzie znajdował się w innym zakresie. Konto **użytkownika *root*** (tzw. *superużytownika*) ma niczym nieograniczony dostęp do systemu operacyjnego. Identyfikatorem konta tego użytkownika zawsze jest zero. Podobnie jak w przypadku innych zadań administracyjnych, także podczas tworzenia i konfigurowania użytkowników konieczne jest podniesienie uprawnień modułu Ansible.

Rozpoczęcie pracy z modułem user w Ansible

Ansible zawiera moduł `user` niezwykle ułatwiający zarządzanie użytkownikami. Ten moduł zajmuje się wszelkimi szczegółami dotyczącymi kont użytkowników, takimi jak powłoka, klucze, grupy i katalog domowy. W omawianym przykładzie moduł `user` będzie użyty do utworzenia nowego użytkownika o nazwie *bender*. Wprawdzie użytkownikowi możesz nadać dowolną nazwę, ale skoro wszystkie przykłady zamieszczone w książce będą korzystały z nazwy użytkownika *bender*, pamiętaj o jej zmianie, jeśli zdecydujesz się na utworzenie innego użytkownika.

W edytorze tekstu otwórz plik *user_and_group.yml* znajdujący się w katalogu *ansible/chapter2/*. Ten plik zawiera pięć wymienionych tutaj zadań:

1. Upewnienie się o istnieniu grupy *developers*.
2. Utworzenie użytkownika *bender*.
3. Przypisanie użytkownika *bender* do grupy *developers*.
4. Utworzenie katalogu o nazwie *engineering*.
5. Utworzenie pliku w katalogu *engineering*.

Te zadania spowodują utworzenie grupy i użytkownika, przypisanie użytkownika do grupy, a następnie utworzenie współdzielonego katalogu i pliku.

Wprawdzie to może wydawać się dziwne, ale pracę najlepiej jest rozpocząć od skoncentrowania się na drugim zadaniu z listy, czyli na utworzeniu użytkownika *bender*. (Pierwszym zadaniem zajmiemy się w następnym podrozdziale). Zadanie tworzące użytkownika przedstawiłem w kolejnym fragmencie kodu.

```
--cięcie--
- name: Create the user 'bender'
  user:
    name: bender
    shell: /bin/bash
    password: $6$...(cięcie)
--cięcie--
```

To zadanie, podobnie jak inne, rozpoczyna się od elementu `name` opisującego jego przeznaczenie. W omawianym przykładzie celem jest utworzenie użytkownika o nazwie *bender* (`name: Create the user 'bender'`). Do tego zostanie wykorzystany moduł `user` w Ansible. Ten moduł ma wiele opcji, z których wymagana jest tylko jedna, `name`, określająca nazwę tworzonego użytkownika. Zdefiniowanie hasła dla tego użytkownika podczas jego tworzenia może być użyteczne, więc parametrowi `password` należy przypisać wartość przedstawiającą wartość tzw. skrótu hasła (do tego zagadnienia jeszcze powrócę). Wartość parametru `password`, rozpoczynająca się od `$6`, to kryptograficzna wartość skrótu obsługiwana przez Linuksa. Zdecydowałem się na podanie przykładowej wartości skrótu hasła dla użytkownika *bender*, aby pokazać możliwość automatyzacji tego kroku. W następnym punkcie znacznie dokładniej przedstawię proces wygenerowania tej wartości.

Wygenerowanie silnego hasła

Można użyć wielu różnych metod w celu wygenerowania hasła odpowiadającego poziomowi złożoności zdefiniowanemu w module `pam_pwquality`. Jak już wcześniej wspomniałem, plik zawiera wartość skrótu hasła dopasowanego do wymagań, więc dzięki temu możesz zaoszczędzić nieco czasu. W celu wygenerowania silnego hasła zostało użyte połączenie dwóch narzędzi (poleceń) powłoki, `pwgen` i `mkpasswd`. Polecenie `pwgen` potrafi generować silne hasła, natomiast polecenie `mkpasswd` może generować hasła z użyciem różnych algorytmów skrótów. To pierwsze polecenie, `pwgen`, jest dostarczane razem z pakietem o takiej samej nazwie, z kolei `mkpasswd` znajduje się w pakiecie o nazwie `whois`. Razem te polecenia pozwalają na wygenerowanie, oczekiwanej przez Ansible i Linuksa, wartości skrótu hasła.

Skróty haseł systemu Linux przechowuje w pliku o nazwie *shadow*. W systemie Ubuntu algorytmem przeznaczonym do tworzenia skrótu dla hasła jest domyślnie SHA-512. Aby samodzielnie utworzyć wartość skrótu SHA-512 dla modułu `user` w Ansible, w hoście Ubuntu wydaj wymienione tutaj polecenia.

```
$ sudo apt update
$ sudo apt install pwgen whois
$ pass=`pwgen --secure --capitalize --numerals --symbols 12 1`
$ echo $pass | mkpasswd --stdin --method=sha-512; echo $pass
```

Skoro wymagane pakiety nie są zainstalowane domyślnie, trzeba zacząć od ich instalacji za pomocą menedżera pakietów `apt`. Polecenie `pwgen` powoduje wygenerowanie silnego hasła, które spełnia warunki zdefiniowane przez moduł `pam_pwquality`, i zapisuje to hasło w zmiennej `pass`. Następnie zawartość zmiennej `pass` jest przekazywana do polecenia `mkpasswd` w celu wygenerowania wartości skrótu z użyciem algorytmu SHA-512. Ostatecznie wyświetlone dane wyjściowe powinny składać się z dwóch wierszy. W pierwszym będzie znajdowała się wartość skrótu SHA-512, natomiast w drugim nowe hasło. Wartość skrótu można wykorzystać jako wartość parametru `password` w trakcie zadania polegającego na utworzeniu użytkownika. Wypróbuj taką możliwość!

Ostrzeżenie

Z oczywistych powodów w istniejącym środowisku produkcyjnym nie należy umieszczać wartości skrótu hasła w systemie kontroli wersji bądź w zadaniu Ansible. W tym przykładzie zdecydowałem się na to tylko dlatego, aby pokazać, jak łatwo można tworzyć silne hasła spełniające wymagania zdefiniowane przez moduł `pam_pwquality`*. W celu zapewnienia ochrony wszelkim informacjom wrażliwym, takim jak hasła lub klucze prywatne, należy korzystać z narzędzia typu Ansible Vault. Przechowuje ono dane wrażliwe w zaszyfrowanych plikach zamiast w scenariuszach bądź w zadaniach. Wykorzystanie takiej techniki wykracza poza zakres tematyczny książki, a więcej informacji na temat Ansible Vault znajdziesz na stronie* https://docs.ansible.com/ansible/latest/vault_guide/index.html.

Grupy w systemie Linux

Grupy w systemie Linux pozwalają na zarządzanie wieloma użytkownikami w hoście. Tworzenie grup to również efektywny sposób ograniczenia dostępu do zasobów hosta. Znacznie łatwiej można administrować zmianami w grupie niż oddzielnie dla setek użytkowników. Następnym przykładem jest zadanie Ansible, którego działanie polega na utworzeniu grupy o nazwie *developers*. Ta grupa zostanie użyta w celu ograniczenia dostępu do katalogu i pliku.

Rozpoczęcie pracy z modułem group w Ansible

Podobnie jak w przypadku pracy z użytkownikami Ansible korzysta z modułu `user`, tak do tworzenia grup i zarządzania nimi Ansible używa modułu o nazwie `group`. W porównaniu z innymi modułami Ansible moduł `group` jest minimalny i może jedynie tworzyć bądź usuwać grupę.

W edytorze tekstu otwórz plik *user_and_group.yml* w celu zapoznania się z zadaniem utworzenia grupy. Pierwsze zadanie w tym pliku ma następującą postać:

```
- name: Ensure group 'developers' exists
  group:
    name: developers
    state: present
--cięcie--
```

Zgodnie z zawartością parametru `name` to zadanie ma zagwarantować istnienie grupy o podanej nazwie. Do tworzenia grupy jest używany moduł `group`. Wymaga ona zdefiniowania parametru `name`, który w omawianym przykładzie ma wartość `developers`. Z kolei parametr `state` ma wartość `present`, więc omawiane zadanie spowoduje utworzenie grupy, jeśli ona jeszcze nie istnieje.

Zadanie utworzenia grupy nie bez powodu zostało zdefiniowane jako pierwsze w pliku. Utworzenie grupy *developers* będzie konieczne jeszcze przed wykonaniem wszelkich pozostałych zadań. Ponieważ zadania są wykonywane po kolei, trzeba się najpierw upewnić, że istnieje żądana grupa. Jeżeli spróbujesz się odwołać do grupy przed jej utworzeniem, otrzymasz komunikat błędu informujący, że grupa o nazwie *developers* nie istnieje, i proces przygotowania zasobu zakończy się niepowodzeniem. Poznanie kolejności wykonywania zadań Ansible ma znaczenie kluczowe w przypadku przeprowadzania znacznie bardziej skomplikowanych operacji.

Podczas analizowania pozostałych zadań plik *user_and_group.yml* pozostaw otwarty w edytorze tekstu.

Przypisywanie użytkownika do grupy

W celu dodania użytkownika do grupy w Ansible ponownie korzysta się z modułu `user`. W pliku *user_and_group.yml* odszukaj zadanie przypisujące użytkownika *bender* grupie *developers* (to będzie trzecie zadanie w pliku). Kod tego zadania przedstawia się następująco:

```
--cięcie--
- name: Assign 'bender' to the 'developers' group
  user:
    name: bender
    groups: developers
    append: yes
--cięcie--
```

Przede wszystkim mamy parametr `name` opisujący przeznaczenie zadania. Moduł `user` spowoduje dodanie użytkownika *bender* do grupy *developers*. Opcja `groups` może przyjąć wiele grup, których nazwy trzeba rozdzielić przecinkami. Dzięki użyciu opcji `append` pozostawiasz nietkniętą dotychczasową listę grup użytkownika *bender* i jedynie dodajesz go do grupy *developers*. W przypadku pominięcia tej opcji użytkownik zostanie usunięty ze wszystkich grup poza jego grupą podstawową i grupą (bądź grupami) wymienioną w parametrze `groups`.

Tworzenie zasobów chronionych

Po wyjaśnieniu kwestii przynależności użytkownika *bender* do grup można przejść do dwóch ostatnich zadań zdefiniowanych w pliku *user_and_group.yml*, które odpowiadają za utworzenie katalogu (*/opt/engineering*) i pliku (*/opt/engineering/private.txt*) w maszynie wirtualnej. Wymienione plik i katalog zostaną później użyte do sprawdzenia uprawnień dostępu użytkownika *bender*.

Mając otwarty w edytorze tekstu plik *user_and_group.yml*, odszukaj te dwa wymienione zadania. Rozpocznij od zadania polegającego na utworzeniu katalogu (to będzie czwarte zadanie w wymienionym pliku). Oto kod tego zadania:

```
- name: Create a directory named 'engineering'
  file:
    path: /opt/engineering
    state: directory
    mode: 0750
    group: developers
```

Przede wszystkim, podobnie jak wcześniej, parametr `name` określa przeznaczenie danego zadania. Użycie modułu `file` pomaga w zarządzaniu katalogiem i jego atrybutami. Parametr `path` wskazuje ścieżkę dostępu, w której będzie utworzony katalog. W omawianym przykładzie to */opt/engineering*. Skoro chcesz utworzyć katalog, parametrowi `state` należy przypisać wartość określającą typ tworzonego zasobu, czyli w omawianym przykładzie to `directory` (pol. *katalog*). W tym miejscu można użyć innych typów zasobów, jak się przekonasz podczas tworzenia pliku nieco dalej w rozdziale. Parametr `mode` określa uprawnienia, które w tym przykładzie zostały zdefiniowane jako `0750`. Pozwalają one właścicielowi (*root*) na odczytywanie, zapisywanie i wykonywanie w katalogu, podczas gdy członkowie grupy mają jedynie uprawnienia odczytu i wykonywania. Uprawnienie wykonywania jest niezbędne w celu wejścia do katalogu i wyświetlenia jego zawartości. Do definiowania uprawnień dla plików i grup system Linux używa zapisu ósemkowego (tutaj to wartość `0750`). Zapoznaj się z podręcznikiem systemowym dla polecenia `chmod`, w którym znajdziesz więcej informacji na temat trybów uprawnień. Ostatnim działaniem w tym zadaniu jest określenie grupy *developers* jako właściciela tworzonego katalogu. To oznacza, że użytkownicy przypisani do grupy *developers* będą mogli odczytywać i wyświetlać zawartość tego katalogu.

Ostatnie zadanie w pliku *user_and_group.yml* powoduje utworzenie pustego pliku w katalogu */opt/engineering*. To jest piąte zadanie w pliku i znajduje się na jego końcu, a jego kod przedstawia się następująco:

```
- name: Create a file in the engineering directory
  file:
    path: "/opt/engineering/private.txt"
    state: touch
    mode: 0770
    group: developers
```

Parametr `name` określa przeznaczenie tego zadania. Ponownie zostanie użyty moduł `file` w celu utworzenia pliku i nadania mu pewnych atrybutów. Wymagany parametr `path` wskazuje położenie pliku w maszynie wirtualnej. W omawianym przykładzie zostanie utworzony plik o nazwie *private.txt* w katalogu */opt/engineering*. Parametrowi `state` została przypisana wartość `touch`, która oznacza utworzenie pustego pliku, jeśli jeszcze on nie istnieje. Jeżeli zachodzi potrzeba utworzenia niepustego pliku, wówczas można skorzystać z modułu `copy` lub `template` w Ansible. Więcej informacji na ich temat znajdziesz w dokumentacji. Parametr `mode` określa uprawnienia, które w omawianym przykładzie oznaczają możliwość odczytu, zapisu i wykonywania dla każdego użytkownika grupy (`0770`). Na końcu parametr `group` określa, że właścicielem pliku jest grupa *developers*.

Trzeba koniecznie wiedzieć o istnieniu wielu metod, za pomocą których można chronić zasoby w hoście systemu Linux. Ograniczenia dotyczące grup to zaledwie niewielki fragment większego stosu autoryzacji, który będziesz mieć do dyspozycji w środowisku produkcyjnym. W późniejszych rozdziałach zostaną omówione różne mechanizmy kontroli dostępu. Teraz wystarczy wiedzieć, że dzięki zadaniom i modułom Ansible można przeprowadzać wiele najczęściej wykonywanych czynności konfiguracyjnych, takich jak zabezpieczanie plików i katalogów w całym środowisku.

Uaktualnianie maszyny wirtualnej

Dotychczas omawiałem moduły Ansible i przedstawiałem zadania umożliwiające przygotowanie maszyny wirtualnej. Następnym krokiem jest faktyczne użycie tych modułów i zadań. Aby przygotować maszynę wirtualną, konieczne będzie usunięcie znaków komentarzy na początku wierszy zadań w scenariuszu znajdującym się w katalogu *ansible*. Plik o nazwie *site.yml* to scenariusz, do którego odwołania znajdują się w sekcji przygotowania infrastruktury w pliku *Vagrantfile*.

W edytorze tekstu otwórz scenariusz *site.yml*, a następnie odszukaj zadania dla rozdziału 2. Fragment pliku ma pokazaną tutaj postać:

```
--cięcie--
tasks:
  #- import_tasks: chapter2/pam_pwquality.yml
  #- import_tasks: chapter2/user_and_group.yml
--cięcie--
```

W tym momencie zadania są wymienione w komentarzach. Usuń znaki # z początku tych dwóch wierszy, aby oprogramowanie Ansible mogło wykonać te zadania.

Ostrzeżenie

Nie usuwaj znaków # z innych zadań, ponieważ ich wykonanie na obecnym etapie może mieć nieoczekiwane konsekwencje. Kolejne zadania będą używane w dalszej części książki.

Po wprowadzeniu zmian ten fragment scenariusza powinien mieć teraz następującą postać:

```
---
- name: Provision VM
  hosts: all
  become: yes
  become_method: sudo
  remote_user: ubuntu
  tasks:
    - import_tasks: chapter2/pam_pwquality.yml
    - import_tasks: chapter2/user_and_group.yml
--cięcie--
```

Oba zadania dla rozdziału 2., `pam_pwquality` i `user_and_group`, nie są dłużej umieszczane w komentarzach, a więc będą wykonywane w trakcie następnej operacji przygotowania maszyny wirtualnej. Zapisz plik scenariusza i zamknij go.

Maszyna wirtualna została utworzona w poprzednim rozdziale. Jeżeli w tym momencie nie działa, wydaj polecenie **`vagrant up`**, aby ją teraz uruchomić. Po uruchomieniu maszyny wirtualnej trzeba jedynie wydać polecenie **`vagrant provision`** z poziomu katalogu *vagrant*, aby w ten sposób przeprowadzić operację przygotowania zasobu.

```
$ vagrant provision
--cięcie--
PLAY RECAP *********************************************************************
Default : ok=8    changed=7    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
```

Ostatni wiersz danych wyjściowych pokazuje, że scenariusz Ansible został wykonany i przeprowadził osiem czynności. Te *czynności* należy traktować jako zadania i inne operacje przeznaczone do uruchomienia. Siedem z tych ośmiu czynności spowodowało pewną zmianę stanu maszyny wirtualnej. Wspomniany wiersz danych wyjściowych informuje, że operacja przygotowania zasobu zakończyła się i nie było żadnych błędów bądź żadne czynności nie zakończyły się niepowodzeniem.

Jeżeli przygotowanie zasobu powoduje błędy, należy się zatrzymać i spróbować je usunąć. W tym celu ponownie wydaj polecenie **`provision`**, tym razem z opcją **`--debug`**, jak już wcześniej wspomniałem w poprzednim rozdziale, aby w ten sposób otrzymać znacznie większą ilość informacji. Operacja przygotowania zasobów musi zakończyć się powodzeniem, aby można było wykonywać przykłady zamieszczone w książce.

Sprawdzanie uprawnień użytkownika i grupy

W celu przetestowania skonfigurowanych przed chwilą uprawnień użytkownika i grupy należy wydać polecenie **`vagrant ssh`**, aby uzyskać dostęp do maszyny wirtualnej. Upewnij się, że nastąpiło przejście do katalogu *vagrant*, ponieważ wtedy będziesz mieć dostęp do pliku *Vagrantfile*. Gdy już się znajdziesz w tym katalogu, wydaj przedstawione tutaj polecenie, dzięki któremu zalogujesz się do maszyny wirtualnej.

```
$ vagrant ssh
vagrant@dftd:~$
```

Zalogujesz się jako użytkownik *vagrant*, czyli użytkownik domyślnie tworzony przez oprogramowanie Vagrant.

Następnym zadaniem jest potwierdzenie utworzenia użytkownika *bender*. W tym celu wykorzystaj polecenie **`getent`** pozwalające na sprawdzenie bazy danych *passwd* pod kątem użytkownika. To polecenie umożliwia sprawdzanie zawartości plików */etc/passwd*, */etc/shadow* i */etc/group*. Dlatego też sprawdzenie istnienia użytkownika *bender* wymaga wydania następującego polecenia:

```
$ getent passwd bender
bender:x:1002:1003::/home/bender:/bin/bash
```

Wygenerowane dane wyjściowe powinny być podobne do tutaj przedstawionych. Jeżeli użytkownik nie został utworzony, wówczas wykonanie polecenia nie wygeneruje żadnych danych wyjściowych.

Teraz trzeba sprawdzić, czy grupa *developers* istnieje i czy użytkownik *bender* jest jej członkiem. Te informacje można uzyskać z bazy danych *group*.

```
$ getent group developers
developers:x:1002:bender
```

Wygenerowane dane wyjściowe powinny być podobne do tutaj przedstawionych — grupa *developers* istnieje i został jej przypisany użytkownik *bender*. Jeżeli podana grupa nie istnieje, wówczas wykonanie polecenia nie wygeneruje żadnych danych wyjściowych.

Ostatnie sprawdzenie polega na ustaleniu, czy tylko członkowie grupy *developers* mogą uzyskać dostęp do katalogu */opt/engineering* i pliku *private.txt*. W tym celu należy spróbować uzyskać dostęp do tego pliku i katalogu raz jako użytkownik *vagrant* i raz jako użytkownik *bender*.

Będąc zalogowanym jako użytkownik *vagrant* wydaj przedstawione tutaj polecenie, którego celem jest wyświetlenie zawartości katalogu */opt/engineering*.

```
$ ls -al /opt/engineering/
ls: cannot open directory '/opt/engineering/': Permission denied
```

Wygenerowane dane wyjściowe wskazują na brak uprawnień dostępu podczas próby wyświetlenia plików katalogu */opt/engineering* jako użytkownik *vagrant*. Tak się dzieje, ponieważ użytkownik *vagrant* nie jest członkiem grupy *developers*, a tym samym nie ma uprawnień odczytu podanego katalogu.

Teraz w celu sprawdzenia uprawnień dostępu do pliku dla użytkownika *vagrant* spróbuj za pomocą polecenia **cat** wyświetlić zawartość pliku */opt/engineering/private.txt*.

```
$ cat /opt/engineering/private.txt
cat: /opt/engineering/private.txt: Permission denied
```

Wygenerowany został ten sam błąd, ponieważ użytkownik *vagrant* nie ma uprawnień dostępu do wymienionego pliku.

Kolejnym krokiem jest potwierdzenie, że użytkownik *bender* ma dostęp do użytego wcześniej pliku i katalogu. W tym celu musisz zalogować się jako użytkownik *bender*. Do zmiany konta użytkownika z *vagrant* na *bender* użyj polecenia **sudo su**. (Dokładne omówienie polecenia `sudo` znajdziesz w rozdziale 4.).

Z poziomu powłoki wydaj następujące polecenie, które pozwoli przejść do konta innego użytkownika:

```
vagrant@dftd:~$ sudo su - bender
bender@dftd:~$
```

Gdy operacja zmiany użytkowników zakończy się sukcesem, ponownie spróbuj wyświetlić zawartość katalogu */opt/engineering/*.

```
$ ls -al /opt/engineering/
total 8
drwxr-x--- 2 root developers    4096 Jul  3 03:59 .
drwxr-xr-x 3 root root          4096 Jul  3 03:59 ..
-rwxrwx--- 1 root developers       0 Jul  3 04:02 private.txt
```

Tym razem jako użytkownik *bender* masz dostęp do tego katalogu i możesz wyświetlić jego zawartość, podobnie jak zawartość pliku *private.txt*.

Wydaj przedstawione tutaj polecenie, aby sprawdzić, czy użytkownik *bender* może odczytać zawartość pliku */opt/engineering/private.txt*.

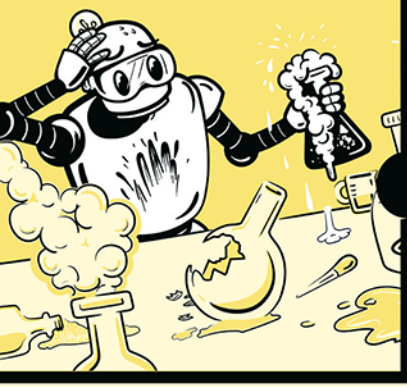
```
$ cat /opt/engineering/private.txt
```

Polecenie **cat** ponownie zostało użyte w celu próby wyświetlenia zawartości pliku. Ten plik jest pusty, więc polecenie nie wygenerowało żadnych danych wyjściowych. Co ważniejsze, próba uzyskania dostępu do tego pliku przez użytkownika *bender* nie spowodowała wygenerowania żadnych komunikatów błędów.

Podsumowanie

W rozdziale zająłem się przygotowaniem maszyny wirtualnej z wykorzystaniem następujących modułów Ansible: `package`, `lineinfile`, `user`, `group` i `file`. Te moduły pozwoliły na skonfigurowanie hosta w celu wymuszenia stosowania silnych haseł, zarządzania użytkownikiem i grupą oraz zabezpieczenia dostępu do pliku i katalogu. To są przykłady jednych z najczęściej wykonywanych zadań przez inżyniera DevOps w typowym środowisku. Dzięki materiałowi w tym rozdziale nie tylko udało Ci się poszerzyć wiedzę z zakresu Ansible, ale również już wiesz, jak możesz automatyzować stosowanie podstawowych reguł bezpieczeństwa w maszynie wirtualnej. W następnym rozdziale będę kontynuował omawianie zadań Ansible i wyjaśnię, jak poprawić bezpieczeństwo SSH przez ograniczenie dostępu do maszyny wirtualnej.


PROGRAM
PARTNERSKI
GRUPY HELION
1. ZAREJESTRUJ SIĘ
2. PREZENTUJ KSIĄŻKI
3. ZBIERAJ PROWIZJĘ
Zmień swoją stronę WWW w działający bankomat!
Dowiedz się więcej i dołącz już dzisiaj!
http://program-partnerski.helion.pl
GRUPA
Helion



DEVOPS: ROZWIJAJ DOSKONAŁOŚĆ NOWOCZESNYCH INFRASTRUKTUR PEŁNEGO STOSU!

Początkowo metodyka DevOps miała wyłącznie ułatwiać współpracę zespołów IT. Dziś obserwuje się rozwój praktyk DevOps związanych z mikrousługami, potokami ciągłej integracji i ciągłego wdrażania, ponadto coraz częściej stosuje się tę metodologię w procesach zapewniania bezpieczeństwa infrastruktury IT, a także optymalizacji z wykorzystaniem uczenia maszynowego i sztucznej inteligencji. Wciąż jednak głównym przedmiotem zainteresowania praktyków DevOps jest dostarczanie standaryzowanego i przewidywalnego oprogramowania. Praktyczna znajomość zasad DevOps przydaje się szczególnie w pracy z nowoczesnym stosem aplikacji.

Tę książkę docenią przede wszystkim inżynierowie oprogramowania i administratorzy systemów, którzy muszą szybko zrozumieć praktyki DevOps. Znajdziesz tu bezcenną wiedzę, która ułatwi Ci efektywną pracę z nowoczesnym stosem aplikacji i sprawne przystąpienie do zadań związanych z DevOps. Poznasz najważniejsze praktyki stosowane podczas projektowania bezpiecznych i stabilnych systemów — implementację infrastruktury jako kodu (IaC) i rozwiązania z zakresu zarządzania konfiguracją. Zagłębisz się w tematy pobierania informacji o stanie systemu i definiowania powiadomień, gdy coś nie działa zgodnie z oczekiwaniami. Lektura pozwoli Ci przyswoić zagadnienia związane z konteneryzacją i przygotowaniem zautomatyzowanego potoku ciągłej integracji i ciągłego wdrażania (CI/CD), dzięki czemu skompilujesz, przetestujesz i wdrożysz kod.

Najciekawsze zagadnienia:

- tworzenie maszyny wirtualnej Ubuntu za pomocą oprogramowania Vagrant i Ansible
- zarządzanie użytkownikami, grupami i bezpieczeństwem haseł
- wdrażanie klucza publicznego i uwierzytelniania wielopoziomowego podczas sesji SSH
- automatyzacja zapory sieciowej bazujcej na hoście
- technologie Docker i Kubernetes
- stos monitorowania i rozwiązywania problemów z wydajnością

Bradley Smith jest doświadczonym inżynierem, który zajmował kierownicze stanowiska w wielu firmach różnej wielkości. Utworzył i wyszkolił liczne zespoły DevOps, SRE i inżynierii oprogramowania.

Helion
helion.pl
HELION SA
ul. Kościuszki 1c
44-100 Gliwice
tel.: 32 230 98 63
helion@helion.pl

KOD KORZYŚCI
Sięgnij po więcej! ▶

ISBN 978-83-289-1126-0

Cena: 67,00 zł

no starch press