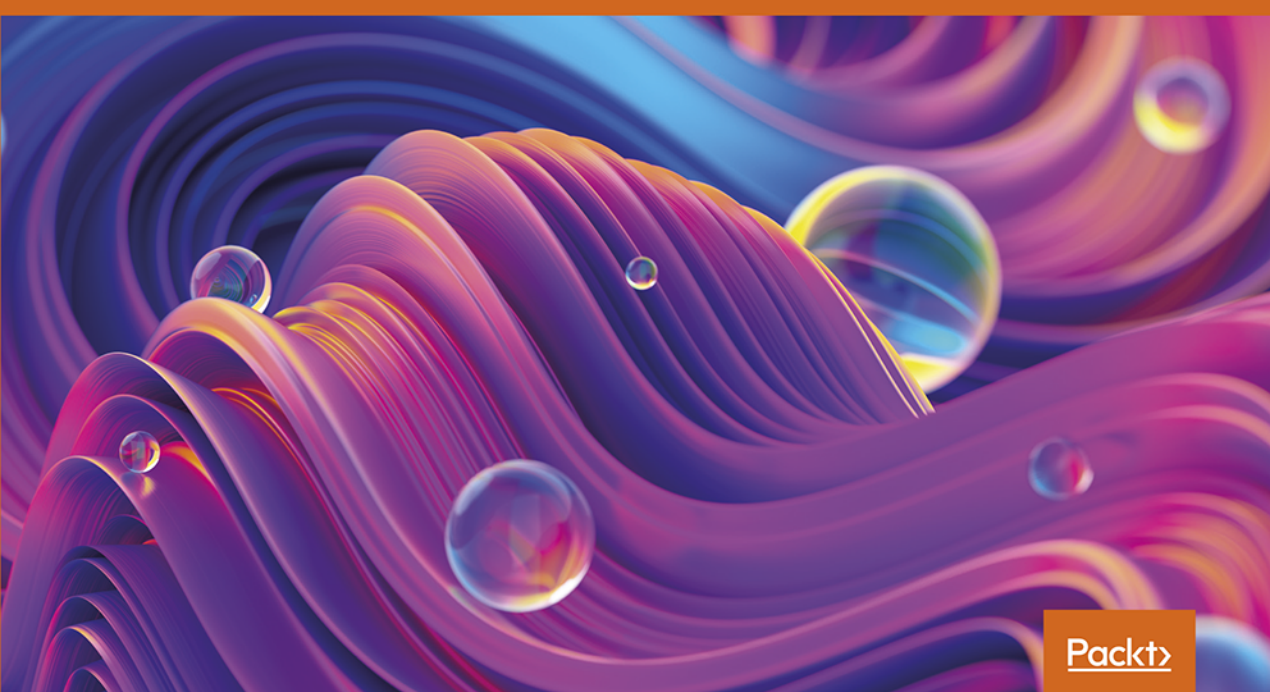
Helion

Ansible 2 w praktyce

Automatyzacja infrastruktury,
zarządzanie konfiguracją i wdrażanie aplikacji



Packt›

Daniel Oh, James Freeman,
Fabio Alessandro Locati

Tytuł oryginału: Practical Ansible 2: Automate infrastructure, manage configuration, and deploy applications with Ansible 2.9

Tłumaczenie: Robert Górczyński

ISBN: 978-83-283-7823-0

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

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

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

Printed in Poland.

Spis treści

O autorach 9

O recenzentach technicznych 11

Wprowadzenie 13

Część I. Podstawy Ansible 19

Rozdział 1. Rozpoczęcie pracy z Ansible 21

Wymagania techniczne 22
Instalacja i konfiguracja Ansible 22
Instalacja Ansible w systemach Linux i FreeBSD 22
Instalacja Ansible w macOS 25
Konfiguracja hosta Windows do pracy z Ansible 27
Poznajemy oprogramowanie Ansible 32
Jak Ansible nawiązuje połączenie z hostem? 32
Weryfikacja poprawności instalacji oprogramowania Ansible 35
Wymagania dotyczące węzła zarządzanego przez Ansible 37
Instalacja Ansible na podstawie kodu źródłowego kontra instalacja z pakietu RPM 39
Podsumowanie 41
Pytania 42
Dalsza lektura 42

Rozdział 2. Podstawy Ansible 43

Wymagania techniczne 44
Poznajemy framework Ansible 44
Komponenty tworzące Ansible 48
Składnia YAML 51
Organizowanie kodu automatyzacji 55

Plik konfiguracyjny Ansible **59**
Argumenty powłoki **63**
Polecenia jednorazowe 65
Definiowanie zmiennych **70**
Filtry Jinja2 **74**
Podsumowanie **78**
Pytania **78**
Dalsza lektura **79**

Rozdział 3. Ewidencja **81**

Wymagania techniczne **81**
Utworzenie pliku ewidencji i dodanie hostów **82**
Używanie grup hostów 84
Dodawanie hostów i zmiennych grup do ewidencji 88
Generowanie pliku ewidencji dynamicznej **94**
Używanie wielu źródeł ewidencji 97
Używanie grup statycznych i dynamicznych 98
Zarządzanie hostami za pomocą wzorców **99**
Podsumowanie **102**
Pytania **102**
Dalsza lektura **103**

Rozdział 4. Scenariusze i role **105**

Wymagania techniczne **106**
Poznawanie frameworka scenariuszy **106**
Porównanie scenariusza i polecenia jednorazowego 111
Definiowanie zbiorów i zadań 114
Poznawanie ról — sposób organizowania scenariuszy **116**
Definiowanie zależności i zmiennych na podstawie roli 121
Ansible Galaxy 127
Konstrukcje warunkowe w kodzie Ansible **128**
Wielokrotne wykonywanie zadań w pętli **133**
Grupowanie zadań za pomocą bloków **138**
Strategie wykonywania scenariusza **143**
Używanie ansible-pull **146**
Podsumowanie **148**
Pytania **149**
Dalsza lektura **149**

Część II. Rozszerzanie możliwości Ansible **151**

Rozdział 5. Tworzenie i używanie modułów **153**

Wymagania techniczne **154**
Wykonywanie wielu modułów w powłoce **154**
Praca z repozytorium modułów **156**
Uzyskanie z poziomu powłoki dostępu do dokumentacji modułu **159**
Wartość zwrotna modułu **161**

Samodzielne opracowanie modułu **163**
Unikanie najczęściej występujących problemów 171
Testowanie i dokumentowanie modułu 173
Lista rzeczy do sprawdzenia podczas tworzenia modułu Ansible 177
Przekazanie kodu modułu do projektu Ansible 178
Podsumowanie **181**
Pytania **182**
Dalsza lektura **183**

Rozdział 6. Tworzenie i używanie wtyczek **185**

Wymagania techniczne **186**
Ustalanie typów wtyczek **186**
Wyszukiwanie wtyczek w standardowej instalacji Ansible **189**
Samodzielne tworzenie wtyczki Ansible **191**
Integracja własnej wtyczki z kodem źródłowym Ansible 198
Przekazanie kodu wtyczki do projektu Ansible 199
Podsumowanie **202**
Pytania **203**
Dalsza lektura **203**

Rozdział 7. Najlepsze praktyki podczas tworzenia kodu **205**

Wymagania techniczne **206**
Preferowana struktura katalogów **206**
Najlepsze praktyki dotyczące ewidencji w chmurze **211**
Odróżnianie poszczególnych typów środowisk **215**
Właściwe podejście w zakresie definiowania zmiennych hostów i grup **216**
Używanie scenariuszy najwyższego poziomu **221**
Wykorzystanie narzędzi systemu kontroli wersji **221**
Definiowanie wariantów systemu operacyjnego i dystrybucji **224**
Przenoszenie kodu między różnymi wersjami Ansible **227**
Podsumowanie **229**
Pytania **230**
Dalsza lektura **230**

Rozdział 8. Zagadnienia zaawansowane w Ansible **231**

Wymagania techniczne **232**
Akcje asynchroniczne kontra synchroniczne **232**
Kontrolowanie wykonywania zbioru podczas stosowania nieustannych uaktualnień **235**
Określenie maksymalnego poziomu niepowodzenia **238**
Konfiguracja delegowania zadań **240**
Używanie opcji run_once **244**
Lokalne uruchamianie scenariuszy **247**
Praca z proxy i używanie hostów pośrednich **249**
Pobieranie danych wejściowych dla scenariusza **250**
Używanie tagów w zbiorach i zadaniach **252**
Zabezpieczanie danych za pomocą Ansible Vault **255**
Podsumowanie **259**
Pytania **259**
Dalsza lektura **260**

Część III. Użycie Ansible w przedsiębiorstwach 261

Rozdział 9. Automatyzacja sieci z Ansible 263

Wymagania techniczne 264
Dlaczego w ogóle należy automatyzować zarządzanie siecią? 264
Jak Ansible zarządza urządzeniami sieciowymi? 266
Jak włączyć automatyzację sieci? 267
Dostępne moduły Ansible przeznaczone do obsługi sieci 269
Nawiązywanie połączenia z urządzeniami sieciowymi 271
Zmienne środowiskowe dla urządzeń sieciowych 273
Konstrukcje warunkowe dla urządzeń sieciowych 275
Podsumowanie 277
Pytania 278
Dalsza lektura 278

Rozdział 10. Zarządzanie kontenerami i chmurami 279

Wymagania techniczne 280
Opracowanie i budowanie kontenerów za pomocą scenariuszy 280
Zarządzanie wieloma platformami kontenerów 282
Wdrażanie do Kubernetes za pomocą narzędzia ansible-container 283
Zarządzanie obiektami Kubernetes za pomocą Ansible 284
Użycie Ansible do automatyzacji Dockera 287
Poznanie modułów związanych z kontenerami 289
Automatyzacja usługi Amazon Web Services 291
Instalacja 292
Uwierzytelnienie 292
Utworzenie pierwszej maszyny wirtualnej 292
Uzupełnienie Google Cloud Platform za pomocą automatyzacji 295
Instalacja 295
Uwierzytelnienie 295
Utworzenie pierwszej maszyny wirtualnej 296
Bezproblemowa integracja automatyzacji z Azure 297
Instalacja 297
Uwierzytelnienie 297
Utworzenie pierwszej maszyny wirtualnej 298
Rozbudowa środowiska za pomocą Rackspace Cloud 300
Instalacja 300
Uwierzytelnienie 301
Utworzenie pierwszej maszyny wirtualnej 301
Użycie Ansible do orkiestracji OpenStack 302
Instalacja 302
Uwierzytelnienie 302
Utworzenie pierwszego scenariusza 303
Podsumowanie 306
Pytania 306
Dalsza lektura 307

Rozdział 11. Rozwiązywanie problemów i strategie testowania 309

Wymagania techniczne 310
Sprawdzanie pod kątem problemów występujących podczas wykonywania scenariuszy 310
Używanie informacji dotyczących hosta do analizy niepowodzeń 311
Testowanie scenariuszy 311
Używanie trybu sprawdzenia 313
Rozwiązywanie problemów dotyczących połączenia z hostem 316
Przekazywanie poprzez CLI zmiennych roboczych 318
Ograniczanie możliwości działania w hoście 319
Opróżnianie bufora kodu 322
Sprawdzanie kodu pod kątem niepoprawnej składni 322
Podsumowanie 323
Pytania 324
Dalsza lektura 324

Rozdział 12. Rozpoczęcie pracy z Ansible Tower 325

Wymagania techniczne 326
Instalacja AWX 326
Uruchomienie z poziomu AWX pierwszego scenariusza 328
Utworzenie projektu AWX 328
Utworzenie ewidencji 330
Utworzenie szablonu zadania 332
Uruchomienie zadania 334
Kontrolowanie dostępu do AWX 335
Utworzenie użytkownika 336
Utworzenie zespołu 337
Utworzenie organizacji 337
Przypisywanie uprawnień w AWX 338
Podsumowanie 339
Pytania 339

ROZDZIAŁ 3

Ewidencja

Jak się dowiedziałeś w dwóch pierwszych rozdziałach, oprogramowanie Ansible nie będzie mogło wykonać żadnych operacji, dopóki nie zostaną wskazane hosty, za których obsługę jest odpowiedzialne. To jest logiczne — nie potrzebujesz narzędzia automatyzacji, które niezależnie od tego, jak jest łatwe w użyciu i konfiguracji, będzie przejmowało kontrolę nad wszystkimi urządzeniami znajdującymi się w danej sieci. Dlatego też absolutnym minimum jest wskazanie Ansible hostów, w których mają być automatyzowane zadania, a także określenie najbardziej podstawowych warunków. Do tego celu służy ewidencja.

Jednak ewidencja to znacznie więcej niż tylko lista hostów, które będą poddawane automatyzacji. Ewidencja Ansible może być zdefiniowana w wielu formatach, może być statyczna lub dynamiczna, a także może zawierać ważne zmienne określające sposób, w jaki Ansible współdziała z poszczególnymi hostami (lub grupami hostów). Dlatego też ewidencja zasługuje na oddzielny rozdział w książce. W tym rozdziale dowiesz się, czym jest ewidencja i jak można wykorzystać jej zalety do automatyzacji infrastruktury za pomocą Ansible.

Oto tematy, które zostały omówione w rozdziale:

- tworzenie pliku ewidencji i dodawanie do niego hostów,
- tworzenie dynamicznego pliku ewidencji,
- użycie wzorców do zarządzania hostami.

Wymagania techniczne

W tym rozdziale przyjęliśmy założenie, że masz zdefiniowany host kontrolny Ansible, jak to zostało dokładnie omówione w rozdziale 1. Ponadto zakładamy, że używasz najnowszej dostępnej wersji Ansible — przykłady zamieszczone w rozdziale zostały przetestowane z Ansible 2.9. Powinieneś mieć również przynajmniej jeden dodatkowy host do testowania; najlepiej, żeby działał

pod kontrolą systemu operacyjnego Linux. Wprawdzie w przykładach zostały użyte konkretne nazwy hostów, ale możesz je zastąpić nazwami hostów lub adresami IP komputerów w swojej sieci. W odpowiednich miejscach znajdują się informacje o tym, jak to należy zrobić.

Przykładowe fragmenty kodu dla tego rozdziału zostały umieszczone w archiwum dostępnym pod adresem *https://ftp.helion.pl/przyklady/ansibl.zip*.

Utworzenie pliku ewidencji i dodanie hostów

Gdy w Ansible napotykasz odwołanie do operacji „utworzenia ewidencji", możesz bezpiecznie przyjąć założenie, że to będzie ewidencja statyczna. Ansible obsługuje dwa rodzaje ewidencji — statyczną i dynamiczną (ta druga jest omówiona w dalszej części rozdziału). Ewidencja statyczna jest z natury bardzo statyczna i pozostaje niezmieniona aż do chwili, gdy będzie zmodyfikowana przez użytkownika. To jest doskonałe rozwiązanie, gdy zaczynasz pracę z Ansible i testujesz to oprogramowanie, ponieważ pozwala bardzo szybko i łatwo przygotować całą niezbędną konfigurację. Nawet w przypadku małych i zamkniętych środowiskach, ewidencja statyczna to świetny sposób na zarządzanie środowiskiem, zwłaszcza jeśli zmiany w infrastrukturze następują bardzo rzadko.

Większość instalacji Ansible będzie szukała domyślnego pliku ewidencji w */etc/ansible/hosts*. (Tę ścieżkę dostępu można zmienić w pliku konfiguracyjnym Ansible, co zostało dokładniej omówione w rozdziale 2.). Możesz umieścić dane w wymienionym pliku bądź też dostarczyć oddzielną ewidencję dla każdego scenariusza. Bardzo często można spotkać się z praktyką dołączania ewidencji do scenariuszy. Tak naprawdę rzadko się zdarza, aby jedna ewidencja była odpowiednia dla wszystkich scenariuszy. Chociaż można podzielić ewidencję na grupy (więcej informacji na ten temat znajdziesz w dalszej części rozdziału), często łatwiejszym rozwiązaniem będzie dostarczenie mniejszego statycznego pliku ewidencji dla danego scenariusza. Jak miałeś okazję zobaczyć w poprzednich rozdziałach książki, większość poleceń zawiera opcję `-i`, która pozwala podać położenie pliku ewidencji, jeśli nie jest używany domyślny plik ewidencji. Spójrz na przykład użycia wspomnianej opcji:

```
$ ansible -i /home/cloud-user/inventory all -m ping
```

Większość plików ewidencji, z którymi można się spotkać, jest tworzona w formacie INI, choć trzeba w tym miejscu dodać, że obsługiwane są również inne formaty. Drugi najczęściej używany format to YAML. Więcej informacji na temat typów plików ewidencji znajdziesz w dokumencie opublikowanym na stronie *https://docs.ansible.com/ansible/latest/user_guide/intro_inventory.html*.

W rozdziale zostaną przedstawione przykłady plików ewidencji w formatach INI i YAML, dlatego powinieneś wiedzieć o ich istnieniu. Niektórzy użytkownicy Ansible przez wiele lat pracują z plikami ewidencji sformatowanymi jako INI lub ewidencjami dynamicznymi, więc nie zaszkodzi mieć wiedzę dotyczącą obu tych formatów.

Rozpoczynamy od utworzenia pliku ewidencji statycznej. Ten plik będzie definiował ewidencję oddzielną od ewidencji domyślnej.

Utwórz plik */etc/ansible/my_inventory* zawierający przedstawiony tutaj kod sformatowany jako INI.

```
target1.example.com ansible_host=192.168.81.142 ansible_port=3333

target2.example.com ansible_port=3333 ansible_user=danieloh

target3.example.com ansible_host=192.168.81.143 ansible_port=5555
```

Nie trzeba umieszczać pustych wierszy między hostami — w tym miejscu zostały użyte po to, by ewidencja w tekście była bardziej czytelna. Ten plik ewidencji jest bardzo prosty i nie zawiera żadnego grupowania. Jednak podczas pracy z ewidencją do wszystkich hostów razem można się odwołać za pomocą grupy specjalnej `all`. Ta grupa jest zdefiniowana niejawnie, niezależnie od sposobu sformatowania i podzielenia pliku ewidencji.

Każdy wiersz w przedstawionym wcześniej pliku zawiera pojedynczy host ewidencji. W pierwszej kolumnie znajduje się nazwa hosta, która będzie używana przez Ansible (i dostępna za pomocą zmiennej magicznej `inventory_hostname`, omówionej w rozdziale 2.). Wszystkie parametry umieszczone w tym samym wierszu po zmiennej są przypisane danemu hostowi. To mogą być zmienne zdefiniowane przez użytkownika lub zmienne specjalne Ansible.

Istnieje wiele takich zmiennych, choć w omawianym przykładzie zostały użyte jedynie trzy:

- `ansible_host`. Jeżeli nazwa hosta wymienionego w pliku ewidencji jest niedostępna bezpośrednio, być może nie została zdefiniowana w DNS — wówczas ta zmienna będzie zawierała nazwę hosta lub adres IP przeznaczony do użycia przez Ansible.
- `ansible_port`. Domyślnie Ansible próbuje prowadzić komunikację przez SSH na porcie `22`. Jeżeli demon SSH nasłuchuje na innym porcie, wówczas trzeba o tym poinformować Ansible za pomocą omawianej zmiennej.
- `ansible_user`. Domyślnie Ansible próbuje nawiązać połączenie ze zdalnym hostem przy użyciu konta bieżącego użytkownika, które jest stosowane do wykonywania poleceń Ansible. To ustawienie można nadpisać na wiele sposobów, w tym za pomocą omawianej zmiennej.

W omawianym przykładzie zostały zdefiniowane trzy hosty:

- Host `target1.example.com` powinien być używany z wykorzystaniem adresu IP `192.168.81.142` i portu `3333`.
- Host `target2.example.com` powinien być używany poprzez port `3333`, choć tym razem należy wykorzystać użytkownika `danieloh` zamiast konta użytkownik, w ramach którego są wydawane polecenia Ansible.
- Host `target3.example.com` powinien być używany z wykorzystaniem adresu IP `192.168.81.143` i portu `5555`.

Dzięki temu nawet bez użycia żadnych konstrukcji można dostrzec potężne możliwości drzemiące w statycznych plikach ewidencji sformatowanych jako INI.

Jeżeli chciałbyś zdefiniować dokładnie tę samą ewidencję, ale w formacie YAML, wówczas jej kod byłby następujący:

```
---
ungrouped:
  hosts:
    target1.example.com:
      ansible_host: 192.168.81.142
      ansible_port: 3333
    target2.example.com:
      ansible_port: 3333
      ansible_user: danieloh
    target3.example.com:
      ansible_host: 192.168.81.143
      ansible_port: 5555
```

Będziesz spotykał się z przykładami plików ewidencji zawierającymi parametry, takie jak `ansible_ssh_port`, `ansible_ssh_host` i `ansible_ssh_user`. Te nazwy zmiennych (i wiele innych) były używane w wersjach Ansible wcześniejszych niż 2.0. Wprawdzie są one nadal obsługiwane w celu zachowania wstecznej zgodności, ale mimo wszystko powinieneś je uaktualnić, ponieważ obsługa tych nazw może zostać usunięta z przyszłych wersji Ansible.

Jeżeli użyjesz przedstawionej przed chwilą ewidencji w Ansible za pomocą zwykłego polecenia `shell`, wówczas otrzymasz wynik podobny do tutaj przedstawionego.

```
$ ansible -i /etc/ansible/my_inventory.yaml all -m shell -a 'echo hello-yaml' -f 5
target1.example.com | CHANGED | rc=0 >>
hello-yaml
target2.example.com | CHANGED | rc=0 >>
hello-yaml
target3.example.com | CHANGED | rc=0 >>
hello-yaml
```

W ten sposób zostały omówione podstawy tworzenia prostego pliku ewidencji statycznej. Rozbudujemy teraz tę ewidencję przez dodanie grup hostów.

Używanie grup hostów

Rzadko się zdarza, że jeden scenariusz jest odpowiedni dla całej infrastruktury. Wprawdzie bardzo łatwo jest nakazać Ansible użycie alternatywnej ewidencji dla scenariusza, ale takie rozwiązanie może bardzo szybko stać się uciążliwe i doprowadzić do powstania setek małych plików ewidencji rozsianych po całej sieci. Wyobraź sobie, jak szybko zarządzanie takim rozwiązaniem stanie się niemożliwe. Celem przyświecającym stworzeniu Ansible było ułatwienie zarządzania hostami, a nie na odwrót. Jednym z możliwych rozwiązań jest dodanie grup do ewidencji.

Przyjmujemy założenie o istnieniu prostej, składającej się z trzech warstw architektury sieci, zawierającej po wiele hostów na każdej warstwie w celu zapewnienia wysokiej dostępności i/lub mechanizmu równoważenia obciążenia. Trzema warstwami w takiej architekturze mogą być:

- serwery frontendu,
- serwery aplikacji,
- serwery baz danych.

Przystępujemy do utworzenia ewidencji dla wspomnianej architektury i wykorzystamy przy tym formaty YAML i INI. W celu zachowania przejrzystości i spójności przykładów zostało przyjęte założenie, że dostęp do wszystkich serwerów odbywa się za pomocą tzw. **w pełni kwalifikowanych nazw domen**, stąd brak jakichkolwiek zmiennych w plikach ewidencji. To oczywiście nie powinno Cię powstrzymać przed dodaniem wspomnianych zmiennych, a każdy przykład jest inny.

Zaczynamy od utworzenia dla wspomnianej architektury ewidencji z użyciem formatu INI. Plik będzie nosił nazwę *hostsgroups-ini*, a jego zawartość jest następująca:

```
loadbalancer.example.com

[frontends]
frt01.example.com
frt02.example.com

[apps]
app01.example.com
app02.example.com

[databases]
dbms01.example.com
dbms02.example.com
```

W tej ewidencji mamy trzy grupy o nazwach `frontends`, `apps` i `databases`. Warto zwrócić uwagę na to, że w przypadku ewidencji sformatowanej jako INI nazwy grup są umieszczone w nawiasach kwadratowych. W ramach każdej nazwy grupy znajdują się wymienione serwery należące do danej grupy. W omawianym tutaj przykładzie mamy po dwa serwery w poszczególnych grupach. Zwróć uwagę na hosta o nazwie `loadbalancer.example.com`, który nie znajduje się w żadnej z wymienionych grup. Wszystkie niegrupowane hosty muszą być wymienione na początku pliku ewidencji w formacie INI.

Zanim przejdziemy dalej, warto dodać, że ewidencja może zawierać również grupy grup, co okazuje się niezwykle użyteczne podczas przetwarzania wybranych zadań. Co się stanie, jeśli w przedstawionej wcześniej ewidencji serwery frontendu działają pod kontrolą systemu operacyjnego Ubuntu, a serwery baz danych działają pod kontrolą dystrybucji CentOS? Będą istniały pewne duże różnice w sposobach obsługi tych hostów — w Ubuntu do zarządzania pakietami używa się menedżera pakietów `apt`, a w CentOS — `yum`.

By zapewnić obsługę takiej sytuacji, można do poszczególnych grup dodać informacje dotyczące systemu operacyjnego używanego przez serwery należące do danej grupy. Jeżeli zdecydujesz się na takie rozwiązanie, utwórz nową wersję pliku ewidencji, jak w poniższym fragmencie kodu:

```
loadbalancer.example.com

[frontends]
```

```
frt01.example.com
frt02.example.com

[apps]
app01.example.com
app02.example.com

[databases]
dbms01.example.com
dbms02.example.com

[centos:children]
apps
databases

[ubuntu:children]
Frontends
```

Słowo kluczowe `children` w definicji grupy (w nawiasie kwadratowym) pozwala utworzyć grupę grup. Dlatego też istnieje możliwość sprytnego grupowania hostów, co pomaga podczas projektowania scenariuszy i jednocześnie pozwala uniknąć konieczności podawania hosta więcej niż tylko raz.

Taka struktura w formacie INI jest czytelna. Potrzeba nieco czasu, by przywyknąć do postaci otrzymanej po skonwertowaniu tej struktury na format YAML. W kolejnym fragmencie kodu możesz zobaczyć wersję YAML poprzedniej ewidencji. Z perspektywy oprogramowania Ansible obie ewidencje są identyczne i to do użytkownika należy decyzja, który format wykorzysta podczas pracy.

```
all:
  hosts:
    loadbalancer.example.com:
  children:
    centos:
      children:
        apps:
          hosts:
            app01.example.com:
            app02.example.com:
        databases:
          hosts:
            dbms01.example.com:
            dbms02.example.com:
    ubuntu:
      children:
        frontends:
          hosts:
            frt01.example.com:
            frt02.example.com:
```

Widać wyraźnie, że słowo kluczowe `children` jest wciąż używane w ewidencji zdefiniowanej w formacie YAML, choć w tym formacie struktura jest znacznie bardziej hierarchiczna niż w formacie INI. Wprawdzie wcięcia mogą ułatwiać zrozumienie struktury, ale zwróć uwagę na to, jak hosty są zdefiniowane za pomocą znacznie większego poziomu wcięć — w zależności od wybranego podejścia rozszerzenie tego formatu może być znacznie trudniejsze.

Gdy chcesz pracować z dowolną grupą wymienioną w poprzedniej ewidencji, wystarczy się do wybranej grupy odwołać w scenariuszu lub w powłoce. Na przykład spójrz na przedstawione tutaj polecenie, w którym odwołujemy się do ostatniej sekcji:

```
$ ansible -i /etc/ansible/my_inventory.yaml all -m shell -a 'echo hello-yaml' -f 5
```

Zwróć uwagę na słowo kluczowe `all` w środku polecenia. To jest grupa specjalna `all`, niejawnie definiowana we wszystkich ewidencjach i wyraźnie wymieniona we wcześniejszym przykładzie w formacie YAML. Jeżeli chcesz wykonać to samo polecenie, ale tym razem jedynie w hostach należących do grupy `centos` w ewidencji zdefiniowanej w formacie YAML, wówczas powinieneś użyć nieco innej wersji tego polecenia:

```
$ ansible -i hostgroups-yml centos -m shell -a 'echo hello-yaml' -f 5
app01.example.com | CHANGED | rc=0 >>
hello-yaml
app02.example.com | CHANGED | rc=0 >>
hello-yaml
dbms01.example.com | CHANGED | rc=0 >>
hello-yaml
dbms02.example.com | CHANGED | rc=0 >>
hello-yaml
```

Jak możesz zobaczyć, ten sposób zarządzania ewidencją zapewnia potężne możliwości, m.in. pozwala łatwo wykonywać polecenia w wybranych hostach. Funkcjonalność tworzenia wielu grup znacznie ułatwia pracę, zwłaszcza jeśli trzeba wykonywać różne zadania w odmiennych grupach serwerów.

Przy okazji tworzenia ewidencji warto wspomnieć o notacji skróconej, która pozwala na zdefiniowanie wielu hostów. Przyjmujemy założenie o istnieniu 100 serwerów aplikacji, których nazwy są sekwencyjne, jak możesz zobaczyć w kolejnym fragmencie kodu:

```
[apps]
app01.example.com
app02.example.com
...
app99.example.com
app100.example.com
```

Zdefiniowanie takiej konfiguracji jest jak najbardziej możliwe. Jednak definiowanie jej ręcznie będzie zadaniem żmudnym i podatnym na błędy, a jednocześnie ewidencja stanie się bardzo trudna w odczycie i interpretacji. Na szczęście Ansible oferuje notację skróconą. Poniższy fragment kodu powoduje zdefiniowanie ewidencji składającej się ze 100 serwerów aplikacji, dokładnie tej samej, która w poprzednim przykładzie była tworzona ręcznie:

```
[apps]
app[01:100].prod.com
```

Istnieje również możliwość użycia zakresu alfabetycznego. Spójrz na rozszerzenie naszego przykładu o serwery buforowania:

```
[caches]
cache-[a:e].prod.com
```

Jeżeli tę samą konfigurację serwerów buforowania chciałbyś utworzyć ręcznie, wówczas musiałbyś przygotować następującą definicję:

```
[caches]
cache-a.prod.com
cache-b.prod.com
cache-c.prod.com
cache-d.prod.com
cache-e.prod.com
```

W ten sposób poznałeś różne formaty ewidencji statycznej i dowiedziałeś się, jak można tworzyć grupy oraz grupy potomne. W następnym punkcie zobaczysz, jak do ewidencji można dodawać hosty i zmiennych grup.

Dodawanie hostów i zmiennych grup do ewidencji

Wcześniej został już poruszony temat zmiennych hostów — miałeś okazję je poznać, gdy były użyte do nadpisania informacji szczegółowych dotyczących połączenia, np. nazwy konta użytkownika, z którym jest nawiązywane połączenie, adresu docelowego czy numeru portu. Jednak zmienne ewidencji w Ansible mają znacznie większe możliwości. Trzeba pamiętać, że mogą być definiowane nie tylko na poziomie hosta, ale również na poziomie grup. To zapewnia wręcz ogromne możliwości w zakresie efektywnego zarządzania infrastrukturą z poziomu jednej centralnej ewidencji.

Będziemy kontynuować pracę z przykładem architektury trójwarstwowej i przyjmujemy założenie, że potrzebne są dwie zmienne dla każdego z dwóch serwerów frontendu. To nie są zmienne specjalne Ansible, ale całkowicie dowolnie wybrane przez użytkownika. Następnie te zmienne będą wykorzystywane w scenariuszu. Przyjmujemy założenie o utworzeniu następujących zmiennych:

- `https_port` — definiuje numer portu, na którym ma nasłuchiwać proxy frontendu,
- `lb_vip` — definiuje w pełni kwalifikowaną ścieżkę dostępu mechanizmu równoważenia obciążenia dla serwerów frontendu.

Zobacz teraz, jak to zostało zrobione.

1. Niezbędne dane można dodać do każdego hosta w sekcji `frontends` pliku ewidencji. To rozwiązanie podobne do zastosowanego wcześniej z użyciem zmiennych połączenia Ansible. W takim przypadku sekcja sformatowanego jako INI pliku ewidencji może wyglądać tak:

```
[frontends]
frt01.example.com https_port=8443 lb_vip=lb.example.com
frt02.example.com https_port=8443 lb_vip=lb.example.com
```

Jeżeli wykonasz polecenie jednorazowe dla tego pliku ewidencji, będziesz miał dostęp do obu zmiennych:

```
$ ansible -i hostvars1-hostgroups-ini frontends -m debug -a
"msg=\"Nawiązanie połączenia z {{ lb_vip }}, nasłuchiwanie na
↪porcie {{ https_port }}\""
frt01.example.com | SUCCESS => {
    "msg": "Nawiązanie połączenia z lb.example.com, nasłuchiwanie na porcie 8443"
}
frt02.example.com | SUCCESS => {
    "msg": "Nawiązanie połączenia z lb.example.com, nasłuchiwanie na porcie 8443"
}
```

Wprawdzie takie rozwiązanie działa zgodnie z oczekiwaniami, ale jest nieefektywne, ponieważ dla każdego hosta trzeba użyć tych samych zmiennych.

2. Na szczęście istnieje możliwość przypisania zmiennych grupie hostów lub pojedynczym hostom. Jeżeli przeprowadzimy edycję wcześniej przedstawionego pliku ewidencji, wówczas sekcja `frontends` będzie miała następującą postać:

```
[frontends]
frt01.example.com
frt02.example.com

[frontends:vars]
https_port=8443
lb_vip=lb.example.com
```

Zwróć uwagę, że ta wersja jest znacznie czytelniejsza od poprzedniej. Jeśli dla nowo zorganizowanej ewidencji wydasz te same polecenia, wygenerowane w wyniku ich wykonania dane wyjściowe będą takie same.

```
$ ansible -i groupvars1-hostgroups-ini frontends -m debug -a
"msg=\"Nawiązanie połączenia z {{ lb_vip }}, nasłuchiwanie na
↪porcie {{ https_port }}\""
frt01.example.com | SUCCESS => {
    "msg": "Nawiązanie połączenia z lb.example.com, nasłuchiwanie na porcie 8443"
}
frt02.example.com | SUCCESS => {
    "msg": "Nawiązanie połączenia z lb.example.com, nasłuchiwanie na porcie 8443"
}
```

3. Zdarzają się sytuacje, gdy trzeba pracować ze zmiennymi dla poszczególnych hostów, w innych zaś właściwsze jest stosowanie zmiennych grup. Do Ciebie należy ustalenie, który z tych scenariuszy jest bardziej odpowiedni. Musisz jednak pamiętać, że zmienne hostów mogą być używane razem. Warto także dodać, że zmienne hostów nadpisują zmienne grup. Dlatego też jeśli konieczna jest zmiana numeru portu połączenia na `8444` dla hosta `frt1.example.com`, można to zrobić w następujący sposób:

```
[frontends]
frt01.example.com https_port=8444
frt02.example.com

[frontends:vars]
```

```
https_port=8443
lb_vip=lb.example.com
```

Jeżeli teraz ponownie wydasz polecenie jednorazowe dla nowej ewidencji, to zobaczysz, że została nadpisana zmienna dla jednego z hostów:

```
$ ansible -i hostvars2-hostgroups-ini frontends -m debug -a
"msg=\"Nawiązanie połączenia z {{ lb_vip }}, nasłuchiwanie na
↪porcie {{ https_port }}\""
frt01.example.com | SUCCESS => {
    "msg": "Nawiązanie połączenia z lb.example.com, nasłuchiwanie na porcie 8444"
}
frt02.example.com | SUCCESS => {
    "msg": "Nawiązanie połączenia z lb.example.com, nasłuchiwanie na porcie 8443"
}
```

Oczywiście, zastosowanie takiego podejścia dla pojedynczego hosta, gdy istnieją tylko dwa hosty wydaje się bezcelowe. Jednak jeśli ewidencja składa się z setek hostów, wówczas taka metoda nadpisywania hosta staje się niezwykle użyteczna.

4. Aby przykład był kompletny, rozważymy jeszcze jedną sytuację. Jeżeli zdefiniowane wcześniej zmienne hostów zostaną dodane do pliku ewidencji przygotowanej w formacie YAML, wtedy sekcja `frontends` będzie miała przedstawioną tutaj postać (pozostała część pliku ewidencji została usunięta w celu zachowania zwięzłości):

```
frontends:
  hosts:
    frt01.example.com:
      https_port: 8444
    frt02.example.com:
  vars:
    https_port: 8443
    lb_vip: lb.example.com
```

Wydanie tego samego polecenia jednorazowego, które było użyte wcześniej, powoduje wygenerowanie tych samych danych wyjściowych jak w przypadku ewidencji zdefiniowanej w formacie INI:

```
$ ansible -i hostvars2-hostgroups-yml frontends -m debug -a
"msg=\"Nawiązanie połączenia z {{ lb_vip }}, nasłuchiwanie na
↪porcie {{ https_port }}\""
frt01.example.com | SUCCESS => {
    "msg": "Nawiązanie połączenia z lb.example.com, nasłuchiwanie na porcie 8444"
}
frt02.example.com | SUCCESS => {
    "msg": "Nawiązanie połączenia z lb.example.com, nasłuchiwanie na porcie 8443"
}
```

5. Dotychczas zostało pokazanych kilka sposobów na dostarczanie zmiennych hostów i grup do pliku ewidencji. Mamy jeszcze jeden sposób zasługujący na omówienie — to rozwiązanie jest szczególnie przydatne, gdy ewidencja staje się większa i coraz bardziej złożona.

Przykłady omówione w rozdziale są małe, zwięzłe i składają się z niewielu grup i zmiennych. Jednak po skalowaniu rozwiązania w górę do postaci pełnej infrastruktury serwerów użycie pojedynczego i jednorodnego pliku ewidencji może doprowadzić do powstania infrastruktury trudnej lub wręcz niemożliwej do zarządzania. Na szczęście Ansible oferuje rozwiązanie także tego problemu. Dwa katalogi specjalne, `host_vars` i `group_vars`, są automatycznie sprawdzane pod kątem odpowiedniej treści zmiennych, które występują w katalogu scenariusza. Można to sprawdzić przez ponowne utworzenie zmiennych frontendu wcześniej omówionego przykładu. Tym razem wykorzystamy specjalną strukturę katalogu, zamiast umieszczać zmienne w pliku ewidencji.

Rozpoczynamy od utworzenia nowej struktury katalogów:

```
$ mkdir vartree
$ cd vartree
```

6. W nowo utworzonym katalogu dodajemy dwa podkatalogi przeznaczone dla zmiennych:

```
$ mkdir host_vars group_vars
```

7. Teraz w katalogu *host_vars* zostanie utworzony plik o nazwie hosta wymagającego ustawienia proxy i rozszerzeniu *.yml* (w omawianym przykładzie to plik *frt01.example.com.yml*). Zawartość tego pliku powinna być następująca:

```
---
https_port: 8444
```

8. Podobnie w katalogu *group_vars* należy utworzyć plik YAML o nazwie grupy, której mają być przypisane zmienne (w omawianym przykładzie to *frontends.yml*). Zawartość tego pliku powinna być następująca:

```
---
https_port: 8443
lb_vip: lb.example.com
```

9. Następnym krokiem jest utworzenie pliku ewidencji w sposób podobny jak wcześniej, ale tym razem bez zmiennych:

```
loadbalancer.example.com

[frontends]
frt01.example.com
frt02.example.com

[apps]
app01.example.com
app02.example.com

[databases]
dbms01.example.com
dbms02.example.com
```

Spójrz na ostateczną strukturę katalogów, którą tutaj utworzyliśmy.

```
$  tree
.
├── group_vars
│   └── frontends.yml
├── host_vars
│   └── frt01.example.com.yml
└── inventory
2 directories, 3 files
```

10. Spróbuj teraz wydać wcześniej używane polecenie jednorazowe i zobacz, jakie zostaną wygenerowane dane wyjściowe:

```
$ ansible -i inventory frontends -m debug -a "msg=\"Nawiązanie połączenia z
{{ lb_vip }}, nasłuchiwanie na porcie {{ https_port }}\""
frt02.example.com | SUCCESS => {
    "msg": "Nawiązanie połączenia z lb.example.com, nasłuchiwanie na porcie 8443"
}
frt01.example.com | SUCCESS => {
    "msg": "Nawiązanie połączenia z lb.example.com, nasłuchiwanie na porcie 8444"
}
```

Jak możesz zobaczyć, takie rozwiązanie działa dokładnie tak samo jak wcześniej. Bez żadnych dodatkowych informacji Ansible porusza się po strukturze katalogów i przetwarza wszystkie pliki zmiennych.

11. Jeżeli masz setki zmiennych (lub wymagasz podejścia zapewniającego większą kontrolę), pliki YAML możesz zastąpić katalogami o nazwach odpowiadających nazwom grup i hostów. Przystępujemy teraz do odtworzenia struktury katalogów, choć tym razem wykorzystamy katalogi zamiast plików.

```
$ tree
.
├── group_vars
│   └── frontends
│       ├── https_port.yml
│       └── lb_vip.yml
├── host_vars
│   └── frt01.example.com
│       └── main.yml
└── inventory
```

Zwróć uwagę na nazwy katalogów odpowiadające nazwom grupy `frontends` i hosta `frt01.example.com`. W katalogu *frontends* zmienne zostały umieszczone w dwóch plikach. Takie podejście może być niezwykle użyteczne, zwłaszcza gdy scenariusze staną się większe i bardziej złożone.

Te pliki to zmodyfikowane wersje wcześniej użytych:

```
$ cat host_vars/frt01.example.com/main.yml
---
https_port: 8444
$ cat group_vars/frontends/https_port.yml
---
```

```
https_port: 8443
$ cat group_vars/frontends/lb_vip.yml
---
lb_vip: lb.example.com
```

Nawet pomimo tej znacznie bardziej szczegółowej struktury katalogów wynik wykonania polecenia jednorazowego jest dokładnie taki sam:

```
$ ansible -i inventory frontends -m debug -a "msg=\"Nawiązanie połączenia z
{{ lb_vip }}, nasłuchiwanie na porcie {{ https_port }}\""
frt01.example.com | SUCCESS => {
    "msg": "Nawiązanie połączenia z lb.example.com, nasłuchiwanie na porcie 8444"
}
frt02.example.com | SUCCESS => {
    "msg": "Nawiązanie połączenia z lb.example.com, nasłuchiwanie na porcie 8443"
}
```

12. Trzeba w tym miejscu koniecznie wspomnieć o jeszcze jednej kwestii. Jeżeli ta sama zmienna zostanie zdefiniowana na poziomie grupy i grupy potomnej, wówczas pierwszeństwo ma zmienna na poziomie grupy potomnej. To nie jest takie oczywiste, jak mogłoby się wydawać. Rozważ przykład wcześniejszej ewidencji, w której grupy potomne pozwoliły na rozróżnianie między hostami CentOS i Ubuntu — jeżeli zmienną o tej samej nazwie dodasz do grupy potomnej `ubuntu` i grupy `frontends` (będącej grupą *potomną* grupy `ubuntu`) w pokazany tutaj sposób, to jaki będzie wynik działania tego kodu? Spójrz na definicję pliku ewidencji:

```
loadbalancer.example.com

[frontends]
frt01.example.com
frt02.example.com

[frontends:vars]
testvar=childgroup

[apps]
app01.example.com
app02.example.com

[databases]
dbms01.example.com
dbms02.example.com

[centos:children]
apps
databases

[ubuntu:children]
frontends

[ubuntu:vars]
testvar=group
```

Teraz wykonaj polecenie jednorazowe i sprawdź wartość zmiennej `testvar`:

```
$ ansible -i hostgroups-children-vars-ini ubuntu -m debug -a "var=testvar"
frt01.example.com | SUCCESS => {
    "testvar": "childgroup"
}
frt02.example.com | SUCCESS => {
    "testvar": "childgroup"
}
```

Koniecznie zapamiętaj, że grupa `frontends` jest grupą potomną grupy `ubuntu` w tym pliku ewidencji (stąd definicją grupy jest `[ubuntu:children]`). Dlatego też wartość zmiennej zdefiniowana na poziomie grupy `frontends` ma w omawianym przykładzie pierwszeństwo.

W tym podrozdziale dowiedziałeś się, jak pracować z plikami ewidencji statycznej. Jednak tematu ewidencji Ansible nie można uznać za wyczerpany bez omówienia ewidencji dynamicznej. Tym rodzajem ewidencji zajmiemy się w następnym podrozdziale.

Generowanie pliku ewidencji dynamicznej

W czasach przetwarzania w chmurze i stosowania rozwiązań typu „infrastruktura jako kod" host poddawany automatyzacji może się zmieniać nie tylko w poszczególnych dniach, ale również z godziny na godzinę. Obsługa statycznych plików ewidencji Ansible może być zadaniem wymagającym osoby zatrudnionej na pełnym etacie. Dlatego w wielu ogromnych rozwiązaniach próby stosowania plików ewidencji statycznej zakończyły się niepowodzeniem.

W tym miejscu do gry wchodzą pliki ewidencji dynamicznej w Ansible. Ogólnie rzecz biorąc, oprogramowanie Ansible ma możliwość pobierania danych ewidencji z wszelkich plików wykonywalnych (choć większość spotykanych ewidencji dynamicznych jest tworzona w Pythonie). Jedynym wymaganiem jest, aby plik wykonywalny zwracał dane ewidencji w formacie JSON. Jeżeli chcesz, możesz tworzyć własne skrypty ewidencji dynamicznej, choć na szczęście nie musisz tego robić samodzielnie — dostępnych jest wiele plików zapewniających potencjalną obsługę źródeł, takich jak Amazon EC2, Microsoft Azure, Red Hat Satellite, katalogi LDAP i wiele innych systemów.

Trudno było nam określić, który ze skryptów ewidencji dynamicznej najlepiej będzie wykorzystać w przykładzie, ponieważ nie każdy czytelnik ma konto w serwisie Amazon EC2 pozwalające bezpłatnie wypróbować przykład. Dlatego też w tym miejscu wykorzystamy system Cobbler, który jest dostępny bezpłatnie i łatwy do użycia w dystrybucji CentOS. Jeżeli jesteś zainteresowany, to powinieneś wiedzieć, że Cobbler to przeznaczony do użycia w systemach operacyjnych Linux system provisioningu dynamicznego, który zapewnia obsługę różnych aspektów: DNS, DHCP, PXE itd. Jeżeli chcesz skorzystać z tego systemu provisioningu w komputerze fizycznym lub w maszynie wirtualnej, wówczas sensowne będzie użycie tego systemu jako źródła ewidencji, ponieważ Cobbler odpowiada przede wszystkim za tworzenie systemów, więc zna wszystkie niezbędne nazwy.

Następny przykład pokazuje podstawy pracy z ewidencją dynamiczną. Można go później rozbudować i użyć skryptów ewidencji dynamicznej dla innych systemów. Rozpoczynamy od zainstalowania systemu Cobbler — omówiona tutaj procedura została przetestowana w dystrybucji CentOS 7.8.

1. Pracę trzeba rozpocząć od zainstalowania za pomocą menedżera pakietów `yum` odpowiednich pakietów dla systemu Cobbler. Warto w tym miejscu dodać, że w chwili gdy piszemy te słowa, mechanizm SELinux w dystrybucji CentOS 7 nie obsługuje funkcjonalności Cobbler i uniemożliwia działanie jego niektórych komponentów. Wprawdzie takiego rozwiązania nie należy stosować w środowisku produkcyjnym, ale podczas testów najprościej będzie wyłączyć mechanizm SELinux.

```
$ yum install -y cobbler cobbler-web
$ setenforce 0
```

2. Następnym krokiem jest upewnienie się, że usługa `cobblerd` została skonfigurowana do nasłuchiwania adresu loopback. To wymaga sprawdzenia ustawień w pliku */etc/cobbler/settings*. Odpowiedni fragment tego pliku powinien wyglądać tak:

```
# default, localhost
server: 127.0.0.1
```

> To nie jest adres publiczny, więc *nie należy używać* adresu `0.0.0.0`. Istnieje również możliwość przypisania adresu IP serwera Cobbler.

3. Po zakończeniu konfiguracji można uruchomić usługę `cobblerd` za pomocą `systemctl`.

```
$ systemctl start cobblerd.service
$ systemctl enable cobblerd.service
$ systemctl status cobblerd.service
```

4. Po uruchomieniu usługi Cobbler następnym krokiem jest przeprowadzenie procesu dodawania dystrybucji do systemu Cobbler i utworzenia pewnych hostów na jego podstawie. Ten proces jest bardzo prosty, choć trzeba dodać plik jądra i początkowy plik RAM-dysku. Najłatwiejszym sposobem na pobranie niezbędnych plików będzie wykorzystanie katalogu */boot*, oczywiście przy założeniu, że system Cobbler został zainstalowany w dystrybucji CentOS. W systemie testowym użytym na potrzeby tego przykładu zostały wydane przedstawione tutaj polecenia. W nazwach plików *vmlinuz* i *initramfs* musisz zmienić numery wersji i podać odpowiednie, które sprawdziłeś w katalogu systemowym */boot*.

```
$ cobbler distro add --name=CentOS
--kernel=/boot/vmlinuz-3.10.0-957.el7.x86_64
--initrd=/boot/initramfs-3.10.0-957.el7.x86_64.img

$ cobbler profile add --name=webservers --distro=CentOS
```

   Ta definicja jest dość podstawowa i niekoniecznie będzie gwarantowała wygenerowanie działających obrazów serwera. Mimo to okazuje się wystarczająca dla omawianego przykładu i pozwala na dodanie pewnych systemów opartych na obrazie CentOS. Zwróć uwagę na to, że nazwa tworzonego profilu, `webservers`, później stanie się nazwą grupy ewidencji w naszej ewidencji dynamicznej.

5. Przystępujemy do kolejnej operacji, czyli dodania systemów do Cobbler. Dwa dalsze polecenia powodują dodanie do systemu Cobbler hostów o nazwach `frontend01` i `frontend02`. Podczas tej operacji używany jest utworzony wcześniej profil `webservers`.

```
$ cobbler system add --name=frontend01 --profile=webservers
--dns-name=frontend01.example.com --interface=eth0

$ cobbler system add --name=frontend02 --profile=webservers
--dns-name=frontend02.example.com --interface=eth0
```

   Trzeba pamiętać, że aby to rozwiązanie działało, musi być zapewniony dostęp do w pełni kwalifikowanych ścieżek dostępu wymienionych w parametrze `--dns-name`. W tym celu należy do pliku */etc/hosts* w systemie Cobbler dodać odpowiednie wpisy dla dwóch wymienionych komputerów. Te wpisy mogą prowadzić do dwóch dowolnie wybranych komputerów.

W tym momencie masz poprawnie zainstalowany system Cobbler, utworzony profil i dodane do tego profilu dwa systemy. Następnym krokiem w procesie jest pobranie i zainstalowanie skryptów ewidencji dynamicznej w Ansible przeznaczonych do pracy z dodanymi systemami. Zapoznaj się z omówieniem kolejnego procesu.

1. Z repozytorium Ansible w serwisie GitHub pobierz plik ewidencji dynamicznej i powiązany z nim szablon pliku konfiguracyjnego. Pamiętaj, że większość dostarczonych przez Ansible skryptów ewidencji dynamicznej ma również szablon pliku konfiguracyjnego, który zawiera parametry niezbędne do zapewnienia działania skryptu ewidencji dynamicznej. W omawianym przykładzie do katalogu bieżącego muszą być pobrane dwa pliki:

```
$ wget https://raw.githubusercontent.com/ansible/ansible/devel/contrib/
↪inventory/cobbler.py
$ wget https://raw.githubusercontent.com/ansible/ansible/devel/contrib/
↪inventory/cobbler.ini
$ chmod +x cobbler.py
```

   Jest bardzo ważne, żeby pobranemu plikowi skryptu ewidencji dynamicznej dodać uprawnienia do wykonywania pliku, jak to zostało pokazane w trzecim poleceniu tego fragmentu kodu. Jeżeli tego nie zrobisz, oprogramowanie Ansible nie będzie mogło uruchomić skryptu, nawet jeśli całość będzie doskonale skonfigurowana.

2. Przeprowadź edycję pliku *cobbler.ini* i upewnij się, że prowadzi do komputera lokalnego, ponieważ oprogramowanie Ansible i system Cobbler w omawianym przykładzie działają w tym samym komputerze. W rzeczywistości trzeba będzie podać zdalny adres URL systemu Cobbler. Przedstawiony tutaj fragment pliku konfiguracyjnego pokazuje odpowiednią konfigurację:

```
[cobbler]
# Określenie adresu IP lub nazwy hosta serwera Cobbler server. To jest wartość domyślna.
host = http://127.0.0.1/cobbler_api

# (Opcjonalnie) W przypadku użycia buforowania odpowiedź na wywołanie API do serwera Cobbler
# będzie szybsza.
```

```
cache_path = /tmp
cache_max_age = 900
```

3. Teraz możesz w już znany Ci sposób wydać polecenie jednorazowe — z tą różnicą, że tym razem należy podać nazwę pliku skryptu ewidencji dynamicznej zamiast nazwy pliku ewidencji statycznej. Przy założeniu, że zostały zdefiniowane hosty o dwóch adresach podanych wcześniej podczas konfiguracji systemu Cobbler, wygenerowane dane wyjściowe powinny być podobne do tutaj przedstawionych:

```
$ ansible -i cobbler.py webservers -m ping
frontend01.example.com | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python"
    },
    "changed": false,
    "ping": "pong"
}
frontend02.example.com | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python"
    },
    "changed": false,
    "ping": "pong"
}
```

I to tyle! W ten sposób zaimplementowałeś swoją pierwszą ewidencję dynamiczną w Ansible. Oczywiście można się spodziewać, że wielu czytelników używa systemu odmiennego niż Cobbler, a część z innych wtyczek ewidencji dynamicznej jest nieco bardziej skomplikowana. Na przykład skrypty ewidencji dynamicznej Amazon EC2 wymagają szczegółów uwierzytelnienia Amazon Web Services (lub odpowiedniego konta IAM), a także zainstalowania bibliotek Pythona `boto` i `boto3`. Skąd to wszystko wiadomo? Odpowiednie informacje znajdują się w nagłówkach skryptów ewidencji dynamicznej lub plików konfiguracyjnych. Oto najlepsza rada, jakiej można w związku z tym udzielić: po pobraniu nowego skryptu ewidencji dynamicznej, upewnij się o dokładnym przeanalizowaniu plików w ulubionym edytorze tekstów, ponieważ w pliku znajdziesz dokumentację dotyczącą wymagań danego skryptu.

W następnej sekcji znajdziesz kilka innych użytecznych podpowiedzi związanych z pracą z ewidencją Ansible. Przede wszystkim dowiesz się, jak można używać wielu źródeł ewidencji.

Używanie wielu źródeł ewidencji

Dotychczas w przykładach zamieszczonych w książce plik ewidencji (statycznej lub dynamicznej) był określany za pomocą opcji `-i` polecenia Ansible. Tej opcji można użyć wielokrotnie i tym samym wskazać jednocześnie wiele ewidencji. To pozwala przeprowadzić takie operacje, jak uruchomienie scenariusza (lub polecenia jednorazowego) jednocześnie w hostach ewidencji zarówno statycznej, jak i dynamicznej. Ansible ustali, co powinno być zrobione — plik ewidencji statycznej nie powinien być oznaczony jako wykonywalny i nie powinien być tak przetwarzany, podczas gdy plik ewidencji dynamicznej ma być oznaczony jako wykonywalny. To drobna, choć zarazem użyteczna sztuczka, pozwalająca łatwo łączyć wiele źródeł ewidencji.

W następnym punkcie zobaczysz, jak używać grup ewidencji statycznej w połączeniu z ewidencją dynamiczną, co można uznać za rozszerzenie funkcjonalności ewidencji pochodzącej z wielu źródeł.

Używanie grup statycznych i dynamicznych

Oczywiście możliwość łączenia ewidencji wiąże się z interesującym pytaniem: Co się stanie z grupami ewidencji dynamicznej i ewidencją statyczną, gdy zostaną zdefiniowane obie? Odpowiedź jest prosta: Ansible pozwala na połączenie obu, co prowadzi do ciekawych rozwiązań. Jak mogłeś zaobserwować, skrypt ewidencji systemu Cobbler wygenerował grupę Ansible o nazwie `webservers` na podstawie profilu Cobbler o tej samej nazwie. To rozwiązanie jest często stosowane u większości dostawców ewidencji dynamicznej. Większość źródeł ewidencji, np. Cobbler i Amazon EC2, nie jest przystosowana do obsługi Ansible i nie oferuje grup, które mogłyby być bezpośrednio używane przez Ansible. W efekcie większość skryptów ewidencji dynamicznej korzysta z pewnych aspektów informacji źródła ewidencji i na tej podstawie generuje grupy. Profil Cobbler jest tutaj przykładem takiego rozwiązania.

Przykład Cobbler z poprzedniego podrozdziału rozbudujemy teraz o połączenie z ewidencją statyczną. Przyjmujemy założenie, że ma być utworzona grupa potomna komputerów `webservers` dla grupy o nazwie `centos`, aby w przyszłości można było grupować ze sobą wszystkie komputery działające pod kontrolą systemu operacyjnego CentOS. Doskonale wiemy, że mamy jedynie profil Cobbler o nazwie `webservers`. W idealnej sytuacji nie chcemy kombinować z konfiguracją Cobbler, aby przeprowadzić operację związaną jedynie z Ansible.

Rozwiązaniem jest tutaj utworzenie pliku ewidencji statycznej z definicjami dwóch grup. Pierwsza musi mieć taką samą nazwę, jaka jest oczekiwana na podstawie ewidencji dynamicznej, choć mamy możliwość pozostawienia pustej nazwy. Gdy Ansible nałoży zawartość ewidencji statycznej i dynamicznej, będzie nakładać na siebie obie grupy i dołączy hosty Cobbler do hostów wymienionych w grupach `webservers`.

Definicja drugiej grupy powinna odzwierciedlać fakt, że `webservers` jest grupą potomną grupy `centos`. Plik wynikowy powinien zawierać fragment kodu podobny do tego:

```
[webservers]

[centos:children]
Webservers
```

Przystępujemy teraz do wydania prostego polecenia jednorazowego `ping` w Ansible, aby zobaczyć, jak przebiegnie analiza obu ewidencji. Zwróć uwagę na wskazanie grupy `centos`, a nie `webservers`, jako docelowej dla polecenia `ping`. Doskonale wiemy, że system Cobbler nie ma grupy `centos`, ponieważ nigdy nie utworzyliśmy w nim grupy o takiej nazwie. To oznacza, że wszystkie wymienione hosty z tej grupy muszą podczas łączenia obu ewidencji pochodzić z grupy `webservers` — ewidencja statyczna nie ma żadnych hostów w grupie `webservers`. Wynikiem wykonania polecenia powinno być wygenerowanie danych wyjściowych podobnych do poniższych:

```
$ ansible -i static-groups-mix-ini -i cobbler.py centos -m ping
frontend01.example.com | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python"
    },
    "changed": false,
    "ping": "pong"
}
frontend02.example.com | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python"
    },
    "changed": false,
    "ping": "pong"
}
```

Jak możesz zobaczyć na podstawie wygenerowanych danych wyjściowych, odwołujemy się do dwóch odmiennych ewidencji: statycznej i dynamicznej. Grupy zostały połączone, hosty istniejące w tylko jednym źródle ewidencji zostały pobrane i połączone z grupą istniejącą w tylko drugim źródle ewidencji. To jest bardzo prosty przykład i bardzo łatwo można go rozbudować o połączenie list hostów ewidencji statycznej i dynamicznej, a także o dodanie zmiennej niestandardowej dla hosta pochodzącego z ewidencji dynamicznej.

To jest mało znana sztuczka Ansible, choć zarazem może zaoferować wraz z rozrastaniem się używanych ewidencji potężne możliwości. Podczas pracy z przykładami zamieszczonymi w rozdziale miałeś okazję zaobserwować, że hosty ewidencji zostały podane bardzo dokładnie, pojedynczo lub w grupach. Na przykład w poleceniu jednorazowym `ansible` wyraźnie wskazaliśmy, że operacje mają być wykonane dla hostów wymienionych w grupie `webservers`. W następnym podrozdziale dowiesz się, jak Ansible może zarządzać zbiorem hostów zdefiniowanych za pomocą wzorca.

Zarządzanie hostami za pomocą wzorców

Dotychczas przekonałeś się, że polecenia jednorazowe lub scenariusze są często wykonywane dla jedynie pewnego podzbioru hostów wymienionych w ewidencji. Jak dotąd dość dokładnie wymienialiśmy te hosty, a w tym podrozdziale dowiesz się, jak Ansible może współpracować z wzorcami w celu wskazania hostów, których mają dotyczyć wykonywane polecenia lub scenariusze.

Zaczynamy od analizy ewidencji zdefiniowanej wcześniej w rozdziale na potrzeby poznawania grup i grup potomnych. Dla ułatwienia w kolejnym fragmencie kodu ponownie znajdziesz definicję tej ewidencji:

```
loadbalancer.example.com

[frontends]
frt01.example.com
frt02.example.com

[apps]
```

```
app01.example.com
app02.example.com

[databases]
dbms01.example.com
dbms02.example.com

[centos:children]
apps
databases

[ubuntu:children]
Frontends
```

Aby pokazać proces wyboru hosta lub grupy hostów za pomocą wzorca, posłużymy się opcją `--list-hosts` polecenia `ansible`. Zadaniem tej opcji jest wyświetlenie hostów, które będą użyte podczas wykonywania danego polecenia. Ten przykład można rozbudować np. o użycie modułu `ping`. Jednak wykorzystamy w nim jedynie opcję `--list-hosts`, aby zachować zwięzłość i czytelność wygenerowanych danych wyjściowych.

1. Wcześniej dowiedziałeś się o istnieniu grupy specjalnej `all`, oznaczającej wszystkie hosty w danej ewidencji:

```
$ ansible -i hostgroups-children-ini all --list-hosts
  hosts (7):
    loadbalancer.example.com
    frt01.example.com
    frt02.example.com
    app01.example.com
    app02.example.com
    dbms01.example.com
    dbms02.example.com
```

   Gwiazdka ma taki sam efekt jak słowo kluczowe `all`, choć musi być ujęta w apostrofy, aby została właściwie zinterpretowana przez powłokę.

```
$ ansible -i hostgroups-children-ini '*' --list-hosts
  hosts (7):
    loadbalancer.example.com
    frt01.example.com
    frt02.example.com
    app01.example.com
    app02.example.com
    dbms01.example.com
    dbms02.example.com
```

2. Użycie dwukropka odpowiada logicznemu *LUB*. W omawianym przykładzie to oznacza „zastosuj dla hostów w podanej grupie lub w tej grupie”.

```
$ ansible -i hostgroups-children-ini frontends:apps --list-hosts
  hosts (4):
    frt01.example.com
    frt02.example.com
    app01.example.com
    app02.example.com
```

3. Wykrzyknik powoduje wykluczenie wskazanej grupy. Wykrzyknik można łączyć z innymi znakami, np. z dwukropkiem, w celu wyświetlenia wszystkich hostów poza znajdującymi się w pewnej grupie (tutaj `apps`). Pamiętaj, że wykrzyknik to znak specjalny w powłoce i dlatego musi być ujęty w apostrofy, aby został właściwie zinterpretowany przez powłokę i działał zgodnie z oczekiwaniami.

```
$ ansible -i hostgroups-children-ini 'all:!apps' --list-hosts
  hosts (5):
    loadbalancer.example.com
    frt01.example.com
    frt02.example.com
    dbms01.example.com
    dbms02.example.com
```

4. Znak `&` oznacza logiczne *I* między np. dwiema grupami. Jeżeli w omawianym przykładzie chcesz otrzymać wszystkie hosty z grup `centos` i `apps`, możesz to zrobić w sposób pokazany w poleceniu. (Znak `&` również ma znaczenie specjalne w powłoce i dlatego musi być ujęty w apostrofy).

```
$ ansible -i hostgroups-children-ini 'centos:&apps' --list-hosts
  hosts (2):
    app01.example.com
    app02.example.com
```

5. Gwiazdka działa podobnie jak gwiazdka w powłoce, o czym możesz się przekonać, analizując poniższe polecenie:

```
$ ansible -i hostgroups-children-ini 'db*.example.com' --list-hosts
  hosts (2):
    dbms02.example.com
    dbms01.example.com
```

Kolejny sposób pozwalający ograniczyć hosty używane w poleceniu to wykorzystanie opcji `--limit` w Ansible. Składnia tej opcji i notacja wzorca są dokładnie takie, jak przedstawiono we wcześniejszej części podrozdziału. Zaletą tej opcji jest możliwość jej użycia w połączeniu z poleceniem `ansible-playbook`, podczas gdy stosowanie wzorca hostów w powłoce jest możliwe jedynie dla samego polecenia `ansible`. Spójrz na kolejny fragment kodu, w którym zostało wydane polecenie `ansible-playbook`:

```
$ ansible-playbook -i hostgroups-children-ini site.yml --limit frontends:apps
PLAY [Prosty scenariusz prezentujący przykład użycia wzorców ewidencji]
******************
TASK [Gathering Facts]
************************************************************
ok: [frt02.example.com]
ok: [app01.example.com]
ok: [frt01.example.com]
ok: [app02.example.com]
TASK [Ping each host]
************************************************************
ok: [app01.example.com]
ok: [app02.example.com]
ok: [frt02.example.com]
ok: [frt01.example.com]
```

```
PLAY RECAP
*********************************************************************
app01.example.com : ok=2 changed=0 unreachable=0 failed=0 skipped=0
rescued=0 ignored=0
app02.example.com : ok=2 changed=0 unreachable=0 failed=0 skipped=0
rescued=0 ignored=0
frt01.example.com : ok=2 changed=0 unreachable=0 failed=0 skipped=0
rescued=0 ignored=0
frt02.example.com : ok=2 changed=0 unreachable=0 failed=0 skipped=0
rescued=0 ignored=0
```

Wzorce są niezwykle użyteczne i jednocześnie bardzo ważne podczas pracy z ewidencją. Nie ulega wątpliwości, że będziesz uznawał wzorce za nieocenioną funkcjonalność. W tym rozdziale poznałeś ewidencję w Ansible. Przedstawiony tutaj materiał powinien dostarczyć Ci wiedzy potrzebnej do wygodnej pracy z tą ewidencją.

Podsumowanie

Tworzenie ewidencji i zarządzanie nimi to bardzo ważny aspekt pracy z Ansible. Dlatego też związane z tym koncepcje zostały omówione na początku książki. Ewidencja ma duże znaczenie, ponieważ bez niej oprogramowanie Ansible nie będzie wiedziało, w których hostach mają być przeprowadzane operacje automatyzacji. Jednak możliwości ewidencji są znacznie większe. Umożliwia ona integrację z systemami zarządzania konfiguracją, stanowi rozsądne źródło zmiennych dla hosta lub grupy, a także zapewnia elastyczny sposób na wykonanie kodu scenariusza.

W rozdziale dowiedziałeś się wiele o tworzeniu prostych plików ewidencji statycznej oraz dodawaniu do nich hostów. Następnie zobaczyłeś, jak dodawać grupy hostów i przypisywać hostom zmienne. Omówione zostało również zagadnienie organizacji ewidencji i zmiennych, gdy pojedynczy jednorodny plik ewidencji staje się trudny w obsłudze. Później omówiliśmy temat używania plików ewidencji dynamicznej oraz przedstawiliśmy kilka podpowiedzi związanych z łączeniem źródeł ewidencji i używaniem wzorców w celu określenia hostów docelowych. Dzięki zdobytej wiedzy praca z ewidencjami stanie się łatwiejsza i jednocześnie będziesz mógł wykorzystać pełnię dostępnych możliwości.

W następnym rozdziale dowiesz się, jak opracowywać scenariusze i role pozwalające konfigurować i wdrażać za pomocą Ansible zdalne komputery oraz nimi zarządzać.

Pytania

1. Jak można dodać zmienne grupy `frontends` do ewidencji?
   a. `[frontends::]`
   b. `[frontends::values]`
   c. `[frontends:host:vars]`

d. `[frontends::variables]`

e. `[frontends:vars]`

2. Co pozwala na automatyzację zadań systemu Linux, takich jak provisioning DNS, zarządzanie serwerem DNS, uaktualnianie pakietów i zarządzanie konfiguracją?

a. scenariusz

b. menedżer pakietów `yum`

c. system Cobbler

d. powłoka Bash

e. rola

3. Prawda czy fałsz? Ansible pozwala wskazać położenie pliku ewidencji za pomocą opcji `-i` w powłoce.

a. Prawda

b. Fałsz

Dalsza lektura

Wszystkie najczęściej stosowane ewidencje dynamiczne Ansible znajdziesz w repozytorium GitHub pod adresem *https://github.com/ansible/ansible/tree/devel/lib/ansible/inventory*.

Żmudne i nudne? Ansible wykona to za Ciebie!

Framework Ansible jest potężnym narzędziem służącym do automatyzacji wdrożeń oprogramowania i zarządzania jego konfiguracjami. Oferuje proste i bardzo przydatne funkcje przeznaczone do automatyzacji wielowarstwowych środowisk za pomocą komunikacji niewymagającej agenta. Przydaje się także do obsługi ciągłej integracji i wdrażania oprogramowania (CI/CD) bez żadnego przestoju. Może służyć do różnych celów: przygotowania infrastruktury jako kodu, wdrożeń aplikacji czy automatyzacji codziennych, czasochłonnych zadań.

Ta książka jest przeznaczona dla osób zajmujących się automatyzacją — od żmudnych codziennych zadań po złożone wdrożenia infrastruktury jako kodu. Ten praktyczny przewodnik pozwoli na sprawne rozpoczęcie korzystania z frameworka Ansible 2.9. Na początku pokazano, jak go zainstalować i skonfigurować, później zaprezentowano proste, jednowierszowe polecenia automatyzacji, aby stopniowo wprowadzać czytelników do tworzenia własnego kodu rozszerzającego możliwości Ansible, a nawet automatyzującego infrastrukturę chmury i kontenerów. Znalazło się tu mnóstwo praktycznych przykładów kodu, a zdobyte umiejętności pozwolą na korzystanie z Ansible w sposób skalowalny, powtarzalny i niezawodny.

W książce:

- podstawy pracy z frameworkiem Ansible
- konfiguracja zależności i zmiennych bazujących na rolach
- unikanie najczęściej popełnianych błędów podczas tworzenia kodu w Ansible
- tworzenie modułów i wtyczek dla Ansible
- rozwiązywanie problemów podczas wykonywania scenariuszy Ansible

Daniel Oh pracuje w Red Hat, gdzie zajmuje się środowiskami uruchomieniowym i frameworkami. Jest również ambasadorem CNCF i DevOps Institute. Udziela się podczas warsztatów i konferencji.

James Freeman jest konsultantem, specjalizuje się we wdrażaniu rozwiązań opartych na Ansible. Jest również autorem i prelegentem na międzynarodowych konferencjach dotyczących tego frameworka.

Fabio Alessandro Locati jest starszym architektem w Red Hat, prelegentem i twórcą oprogramowania open source. Zajmuje się systemem Linux, automatyzacją oraz technologiami chmury.

ISBN 978-83-283-7823-0
9 788328 378230
Cena: 79,00 zł

Packt