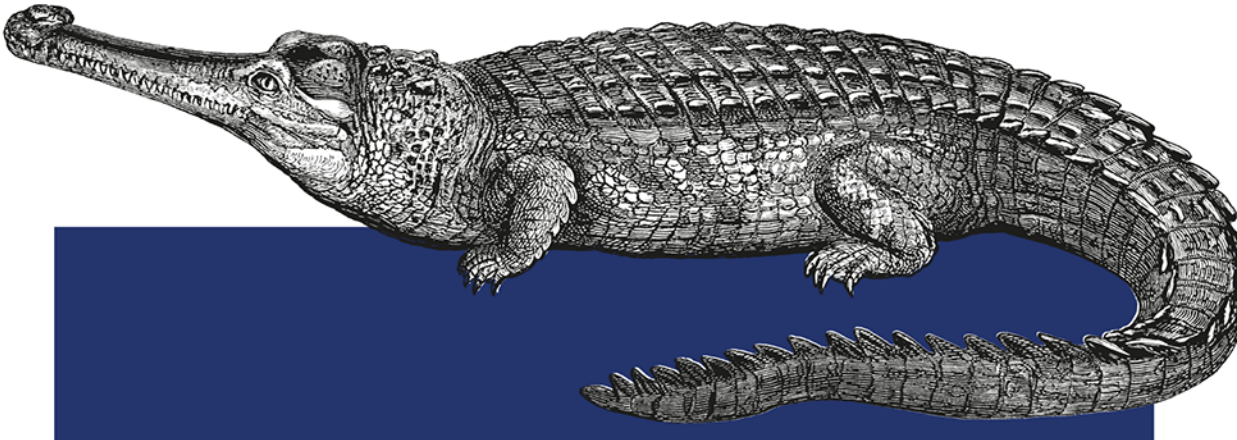
O'REILLY®



Programowalność i automatyzacja sieci

PORADNIK INŻYNIERA SIECI NASTĘPNEJ GENERACJI

Jason Edelman
Scott S. Lowe
Matt Oswalt

Helion

Tytuł oryginału: Network Programmability and Automation: Skills for the Next-Generation Network Engineer

Tłumaczenie: Jacek Litka

ISBN: 978-83-283-5045-8

Helion SA
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/prausi
Możesz tam wpisać swoje uwagi, spostrzeżenia, recenzję.

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

Printed in Poland.

Spis treści

ROZDZIAŁ 2.

Automatyzacja sieci

W rozdziale tym skupimy się na przedstawieniu podstaw dotyczących koncepcji wysokopoziomowej automatyzacji sieci, abyś był lepiej wyposażony w wiedzę potrzebną do pełnego zrozumienia materiału prezentowanego w następnych rozdziałach.

W tym celu podzieliliśmy ten rozdział na poniższe podrozdziały:

Po co automatyzować sieci?
: Przedstawiamy tu różne powody adaptacji automatyzacji poprawiającej wydajność operacji sieciowych, jednocześnie podkreślając, że automatyzacja to coś więcej niż tylko szybsze dostarczanie konfiguracji do urządzeń sieciowych.

Typy automatyzacji sieci
: Przegląd różnego typu automatyzacji, od tradycyjnego zarządzania konfiguracją po automatyzację diagnostyki i rozwiązywanie problemów. Przypominamy raz jeszcze, że automatyzacja sieci to coś więcej niż tylko skrócenie czasu potrzebnego na zastosowanie zmian w konfiguracji urządzenia.

Ewolucja płaszczyzny zarządzania od SNMP do API urządzeń
: Krótkie wprowadzenie do API sieciowych dostępnych w starszych i współczesnych urządzeniach sieciowych.

Automatyzacja sieci w erze SDN
: Krótkie podsumowanie, wyjaśniające, dlaczego automatyzacja narzędzi sieciowych jest wciąż warta stosowania mimo wdrażania rozwiązań SDN. Mamy tutaj na myśli przede wszystkim architektury oparte na kontrolerach.

Rozdział ten nie jest rozdziałem technicznym. Jego zadaniem jest wprowadzenie do idei i koncepcji automatyzacji sieci. Stanowi jedynie podstawę i kontekst kolejnych rozdziałów.

Po co automatyzować sieci?

Automatyzacja sieci, tak jak automatyzacja ogólnie, została opracowana jako metoda przyspieszenia pracy. Mimo że szybkie wykonywanie pracy jest wygodne, to redukcja czasu potrzebnego na wdrożenia i zmiany w konfiguracji w wielu organizacjach IT nie zawsze jest problemem koniecznym do rozwiązania.

Mając na uwadze oszczędność czasu, rzucimy okiem na kilka powodów, dla których różnej wielkości organizacje IT powinny zacząć stopniowo wprowadzać automatyzację sieci. Zwróć uwagę, że przyczyny wprowadzania automatyzacji w innych dziedzinach (aplikacje, systemy, przechowywanie danych, telefonia itd.) były identyczne.

Uproszczone architektury

Dzisiaj w większości przypadków żadne dwa urządzenia sieciowe nie mają identycznej konfiguracji. Różnią się między sobą niczym dwa unikalne płatki śniegu (posiadają wiele unikatowych, niestandardowych konfiguracji). Wielu inżynierów sieciowych ma powód do dumy, gdy udaje się im rozwiązać problemy dotyczące transportu i aplikacji, wynikające z dokonywania jednorazowych zmian w sieci, które w ostatecznym rozrachunku prowadzą do tego, że sieci stają się nie tylko trudniejsze w zarządzaniu i utrzymaniu, ale też i trudniejsze do zautomatyzowania.

Zamiast traktować automatyzację sieci i zarządzania jako drugorzędny projekt lub jako „dodatek", powinno się ją traktować jako kluczowy element nowo tworzonej architektury. Powinno się uwzględniać zarezerwowanie budżetu zarówno na personel, jak i narzędzia automatyzacji. Niestety, narzędzia automatyzacji często jako pierwsze padają ofiarą cięć budżetowych.

Architektura od końca do końca i powiązane z nią operacje dnia drugiego powinny być jednakowe. Przed wdrożeniem architektury należy znaleźć odpowiedź na poniższe pytania:

- Jakie funkcje działają niezależnie od dostawców?
- Jakie rozszerzenia działają niezależnie od platform?
- Jakie rodzaje API lub narzędzi automatyzacji działają z danymi platformami urządzeń sieciowych?
- Czy API zostało solidnie udokumentowane?
- Jakie biblioteki programistyczne można wykorzystać w danym projekcie?

Jeżeli odpowiedzi na te pytania zostaną znalezione na wczesnym etapie projektu, wynikowa architektura będzie prosta, łatwa do powielenia oraz utrzymania i automatyzacji, a to wszystko przy zmniejszonej liczbie działających w sieci rozszerzeń zależnych od dostawców.

Nawet przy tak uproszczonej architekturze, z poprawnym zarządzaniem i narzędziami automatyzacji, należy pamiętać o tym, aby zminimalizować liczbę jednorazowych zmian, by zapewnić, że konfiguracje urządzeń sieciowych nie staną się ponownie różnorodne jak płatki śniegu.

Deterministyczny rezultat

W firmach organizuje się spotkania w celu oceny proponowanych zmian w sieci, ich wpływu na zewnętrzne systemy i plany przywrócenia poprzedniego stanu. W świecie, w którym jeden człowiek sięga po wiersz poleceń w celu wprowadzenia jednej zmiany, niepoprawnie wpisana komenda może być katastrofalna w skutkach. Wyobraź sobie zespół składający się z 3, 4, 5 lub 50 inżynierów. Każdy z tych inżynierów może mieć własny pomysł na zrealizowanie planowanej zmiany. Ponadto zdolność do wykorzystania wiersza poleceń albo nawet i GUI nie eliminuje ani nie redukuje możliwości wystąpienia błędu w trakcie okna czasowego przeznaczonego na wprowadzenie potrzebnych zmian.

Używanie sprawdzonych i przetestowanych narzędzi automatyzacji sieci w celu dokonywania zmian pomaga uzyskać *bardziej przewidywalny* wynik niż w przypadku manualnych zmian i daje zespołowi je wykonującemu *większą szansę na* deterministyczny rezultat, co przybliża go do pewności, że zadanie zostanie wykonane bez wystąpienia błędu ludzkiego. Może to być dowolne zadanie, od zmiany w wirtualnej sieci lokalnej VLAN do wykonania szeregu zmian w sieci u klienta.

Biznesowa swoboda działania

Wiemy, że automatyzacja sieci gwarantuje szybkość i swobodę działania przy wdrażaniu zmian, ale umożliwia ona również pozyskiwanie danych od urządzeń sieciowych tak szybko, jak wymagają tego potrzeby biznesowe, lub ujmując to w sposób bardziej praktyczny, tak szybko, jak jest to potrzebne, aby w dynamiczny sposób rozwiązywać zaistniałe w sieci problemy.

Wraz z nadejściem wirtualizacji serwerów administratorzy serwerów i wirtualizacji otrzymali możliwość niemal natychmiastowego wdrażania nowych aplikacji. Z szybszym wdrażaniem aplikacji pojawiają się pytania o to, dlaczego tyle czasu zajmuje konfigurowanie zasobów sieciowych, takich jak sieci VLAN, trasy, zasady filtrowania ruchu, zasady równoważenia ruchu lub wszystkie powyższe naraz w przypadku wdrażania aplikacji trzeciego poziomu.

Powinno być w miarę oczywiste, że wraz z adaptacją automatyzacji sieci zespoły inżynierów sieciowych i zespoły operacyjne mogą reagować szybciej od swoich odpowiedników IT przy wdrażaniu aplikacji, jednak co ważniejsze, pomaga ona biznesowi uzyskać większą swobodę działania. Z perspektywy adaptacji krytyczne jest zrozumienie istniejących, a do tego często manualnych przepływów pracy przed podjęciem jakiejkolwiek próby automatyzacji, aby Twój biznes otrzymał więcej swobody działania.

Jeżeli nie masz pojęcia, co dokładnie chcesz zautomatyzować, skomplikuje to i wydłuży proces. Nasza rada *numer jeden* — kiedy zaczniesz swoją pracę z automatyzacją sieci, powinieneś zawsze rozumieć istniejące ręczne przepływy pracy, dokumentować je i rozumieć ich wpływ na biznes. Wtedy proces wprowadzenia technologii automatyzacji narzędzi stanie się dużo prostszy.

W sekcji tej przedstawiliśmy ważne powody, dla których powinieneś rozważyć automatyzację sieci (od uproszczonej architektury do swobody działania biznesu). W następnym podrozdziale przyjrzymy się różnym rodzajom automatyzacji sieci.

Typy automatyzacji sieci

Automatyzacja powszechnie utożsamiana jest z szybkością, a skoro nie każde działanie w sieci wymaga szybkości, łatwe do zrozumienia staje się, dlaczego niektóre zespoły IT nie dostrzegają w niej wartości. Konfiguracja sieci VLAN może być świetnym tego przykładem, ponieważ można pomyśleć: „Jak *szybko* powinno się tak naprawdę tworzyć te sieci VLAN? Jak dużo sieci VLAN zostaje utworzonych każdego dnia? Czy ja naprawdę potrzebuję tej całej automatyzacji?”. Wszystkie te pytania są zasadne.

W tej sekcji skupimy się na szeregu innych zadań, przy których automatyzacja ma sens, takich jak zaopatrywanie urządzeń, zbieranie danych, rozwiązywanie problemów, raportowanie i zachowywanie zgodności. Pamiętaj jednak, że jak zaznaczyliśmy wcześniej, automatyzacja to dużo więcej niż tylko szybkość i swoboda działania; oferuje ona też Tobie, Twojemu zespołowi i Twojemu biznesowi bardziej przewidywalne i bardziej deterministyczne rezultaty.

Zaopatrywanie urządzeń

Jedną z najprostszych i najszybszych metod na rozpoczęcie pracy z automatyzacją sieci jest automatyzacja tworzenia plików konfiguracji urządzeń, które wykorzysta się do wstępnego zaopatrywania urządzeń, oraz automatyzacja przesyłania ich do urządzeń sieciowych.

W celu zautomatyzowania procesu tworzenia plików konfiguracyjnych należy oddzielić **dane wejściowe** (parametry konfiguracyjne) od podległej im zależnej od dostawcy składni konfiguracyjnej (wiersza poleceń). Oznacza to, że skończymy z oddzielnymi plikami, z których każdy zawierać będzie wartości i parametry konfiguracyjne dotyczące różnych zagadnień, takich jak sieci VLAN, informacje o domenie, interfejsy, routing i wszystkie inne rzeczy wchodzące w skład konfiguracji oraz, rzecz jasna, szablon konfiguracji. Jest to temat, któremu poświęcimy dużo więcej uwagi w rozdziale 6.

Na razie traktuj szablon konfiguracji jako ekwiwalent standardowego złotego szablonu (ang. *golden template*), który wykorzystuje się dla wszystkich wdrażanych urządzeń. Stosując technikę zwaną **szablonami konfiguracji sieciowej**, uzyskujesz w szybki sposób spójne pliki konfiguracyjne dla wszystkich urządzeń w Twojej sieci. Oznacza to również, że już nigdy więcej nie uruchomisz Notatnika w celu skopiowania i wklejenia konfiguracji z pliku do pliku — nie najwyższa na to pora?

Dwoma narzędziami, które upraszczają korzystanie z szablonów konfiguracji ze zmiennymi (danymi wejściowymi), są Ansible i Salt. W mniej niż kilka sekund narzędzia te są zdolne wygenerować setki plików konfiguracyjnych w sposób przewidywalny i niezawodny.

Budowanie i generowanie plików konfiguracyjnych zostało przedstawione szczegółowo w rozdziale 6., a wykorzystanie narzędzi Ansible i Salt zostało opisane w rozdziale 9. Ten rozdział jedynie przedstawia wysokopoziomowy, podstawowy przykład.

Spójrzmy na przykład przerobienia istniejącej konfiguracji na szablon i oddzielny plik ze zmiennymi (danymi wejściowymi).

Oto przykład wycinka z pliku konfiguracyjnego:

```
hostname leaf1
ip domain-name ntc.com
!
vlan 10
name web
!
vlan 20
name app
!
vlan 30
name db
!
```

Jeżeli oddzielimy dane od komend wiersza poleceń, plik zostanie przetransformowany na dwa pliki: szablon i plik danych (zmiennych).

Spójrzmy na plik YAML (YAML opisujemy szczegółowo w rozdziale 5.) ze zmiennymi:

```
---
hostname leaf1
domain_name: ntc.com
vlans:
- id: 10
name: web
- id: 20
name: app
- id: 30
name: db
```

Zwróć uwagę, że plik YAML to nasze jedyne *dane*.

W tym przykładzie pokazujemy język szablonowania Jinja oparty na Pythonie. Jinja jest opisana w rozdziale 6.

Wynikowy szablon, który zostanie utworzony wraz z plikiem danych, ma nazwę *leaf.j2* i wygląda następująco:

```
!
hostname {{ inventory_hostname }}
ip domain-name {{ domain_name }}
!
!
{% for vlan in vlans %}
vlan {{ vlan.id }}
name {{ vlan.name }}
{% endfor %}
!
```

W tym przykładzie **podwójny nawias klamrowy** wskazuje na zmienną języka Jinja. Innymi słowy, jest to miejsce, w którym dane o zmiennych zostają wprowadzone, kiedy szablon jest stworzony wraz z danymi. Skoro podwójne nawiasy klamrowe wskazują na zmienne i widzimy, że ich wartości nie znajdują się w szablonie, to znaczy, że musiały zostać przechowane gdzieś indziej.

Przechowane zostały w pliku YAML. Zamiast wykorzystywać płaskie pliki YAML, mógłbyś także wykorzystać skrypt, który sięgnąłby po te informacje z zewnętrznego systemu, takiego jak system zarządzania siecią (ang. *Network Management System* — NMS) lub system zarządzania adresami IP (ang. *IP address management system* — IPAM).

Jeżeli w tym przykładzie zespół kontrolujący sieci VLAN zechce dodać VLAN do urządzeń sieciowych, to nie będzie to stanowić żadnego problemu. Wystarczy, że zespół wprowadzi zmiany w pliku ze zmiennymi i utworzy na nowo plik konfiguracyjny, wykorzystując Ansible lub silnik renderujący swojego wyboru (Salt, czysty Python itp.).

W rozdziale 6. opisujemy również, jak wykorzystać czysty Python z szablonami języka Jinja w celu utworzenia skryptu Pythona, który może działać jako podstawowy silnik renderujący.

Teraz w naszym przykładzie wygenerowaną konfigurację należy przesłać do urządzenia sieciowego. Procesu *przesłania* i *wykonania* tutaj nie przedstawiamy — istnieje mnóstwo sposobów na zrobienie tego, wliczając w to zamknięte automatyczne rozwiązania zaopatrywania i kilka innych metod, o których wspomnimy w rozdziałach 7. i 9.

Wszystko to miało posłużyć wyłącznie jako wysokopoziomowe wprowadzenie do szablonów; nie martw się, jeśli coś jeszcze nie jest jasne. Jak wspomnieliśmy, praca z szablonami jest opisana szczegółowo w rozdziale 6.

Obok budowania konfiguracji i wysyłania ich do urządzeń ważne jest zbieranie danych, co stanowi nasz następny temat.

Zbieranie danych

Narzędzia do monitorowania zazwyczaj wykorzystują Simple Network Management Protocol (SNMP). Narzędzia te pobierają pewne bazy informacji zarządzania (ang. *management information base* — MIB) i zwracają informacje do narzędzia monitorowania. Pozyskanych danych może być więcej lub mniej, niż faktycznie potrzebujesz. A jeżeli pozyskiwane są statystyki interfejsów? Możesz otrzymać każdy licznik, który jest przedstawiony przez polecenie `show interface`. Ale co, jeśli potrzebujesz wyłącznie informacji o liczbie *resetów interfejsu*, a nie wszystkie te informacje o *błędach CRC*, *ramkach typu jumbo*, *błędach wyjściowych* itd.? Co więcej, co, jeżeli chcesz zobaczyć liczbę resetów interfejsu w korelacji do interfejsów posiadających sąsiadów CDP/LLDP i chcesz to wszystko zobaczyć *teraz*, a nie przy następnym cyklu pozyskiwania danych? Jak automatyzacja sieci w tym pomaga?

Biorąc pod uwagę to, że naszym celem jest danie Ci więcej możliwości i kontroli, możesz wykorzystać narzędzia i technologie open source, aby dopasować dokładnie do swoich potrzeb, co chcesz otrzymać i kiedy, jak to ma być sformatowane i jak dane mają być wykorzystane, gdy już zostaną zebrane, sprawiając w ten sposób, aby uzyskać z tych danych jak najwięcej.

Zamieszczamy tutaj *bardzo* prosty przykład zbierania danych z urządzenia IOS z wykorzystaniem biblioteki Pythona *netmiko*, której przyjrzymy się uważniej w rozdziale 7.

```
from netmiko import ConnectHandler
device = ConnectHandler(device_type='cisco_ios', ip='csr1',username='ntc',
password='ntc123')
output = device.send_command('show version')
print(output)
```

Najlepsze w tym wszystkim jest to, że *wyjście* zawiera odpowiedź na `show version`, a Ty otrzymujesz możliwość przetworzenia tego wszystkiego według własnych potrzeb.

W przedstawionym przykładzie opisujemy *pozyskiwanie* danych z urządzeń, co nie musi być idealnym rozwiązaniem we wszystkich środowiskach, lecz dla wielu może być użyteczne. Miej na uwadze, że nowsze urządzenia zaczynają obsługiwać także *przepychanie* danych, często nazywane streamowaniem telemetrii, w którym to przypadku samo urządzenie streamuje dane w czasie rzeczywistym, takie jak statystyki interfejsów, do serwera aplikacji, który wybrałeś.

Rzecz jasna, wiele z tego wymaga wykonania nieco pracy z dostosowaniem wszystkiego do własnych potrzeb, jednak w ostatecznym rozrachunku jest to gra warta świeczki, ponieważ pozyskujesz dane, których potrzebujesz, a nie dane, które dostarcza Ci określone narzędzie lub określony dostawca. Czyż jest to nie jest powód, dla którego czytasz tę książkę?

Urządzenia sieciowe są wyposażone w ogromne ilości statycznych i efemerycznych danych zagrzebanych gdzieś w ich głębi, a korzystanie z narzędzi open source i opracowywanie ich samemu umożliwia Ci do nich dostęp. Przykładem takich danych są aktywne wpisy w tablicy BGP, sąsiedztwo OSPF, aktywni sąsiedzi, statystyki interfejsów, szczegółowe liczniki, liczniki resetów, a nawet liczniki ze specjalizowanych układów scalonych ASIC w przypadku nowszych platform. Dodatkowo istnieje nieco faktów i charakterystyk urządzeń, które można zebrać, takich jak numer seryjny, nazwa hosta, czas pracy, wersja systemu operacyjnego i platformy sprzętowej. Lista jest niemal nieskończona.

Kiedy rozpoczynasz projekt automatyzacji, zawsze rozważ następujące pytania: „Czy ma sens zbudowanie tego, kupienie tego lub dostosowanie tego?” i „Czy ma sens korzystanie z tego i operowanie tym?”.

Migracje

Migracja z jednej platformy na drugą nigdy nie jest prostym zadaniem, niezależnie od tego, czy migruje się pomiędzy platformami od tego samego dostawcy, czy od różnych. Dostawca może oferować skrypt lub narzędzie pomocne przy migracji na *jego* platformę, za to różnego rodzaje automatyzacje mogą zostać wykorzystane do zbudowania szablonów konfiguracji, takich jak nasz wcześniejszy przykładowy szablon, dla wszystkich rodzajów urządzeń sieciowych i systemów operacyjnych w taki sposób, że mógłbyś wygenerować plik konfiguracyjny dla każdego dostawcy, pod warunkiem że wykorzystywaliby wspólny zestaw danych wejściowych (wspólny model danych).

Oczywiście nie można przy tym zapomnieć o unikalnych dla dostawcy rozszerzeniach. Piękne w tym wszystkim jest to, że narzędzia migracyjne takie jak te są dużo prostsze do zbudowania samemu niż przez dostawcę, gdyż dostawca musi wziąć pod uwagę całą funkcjonalność oferowaną

przez swoje rozwiązanie w przeciwieństwie do organizacji, która jest zainteresowana wyłącznie pewnym jej zbiorem. W rzeczywistości jest to jedna z rzeczy, które nie zaprzątają głowy dostawcom; skupieni są na swoim własnym ekwipunku, a nie na tym, abyś Ty, operator sieci, miał ułatwioną pracę w środowisku złożonym z elementów od różnych dostawców.

Posiadanie tego typu swobody działania pomaga nie tylko przy migracjach, ale też w środowiskach odtwarzania zniszczeń DR. Powszechną praktyką jest posiadanie różnych przełączników w centrach danych w produkcji i w DR, często różnych dostawców. Jeżeli urządzenie zawiedzie z jakiegoś powodu, a jego zastępstwem będzie inna platforma, uzyskasz możliwość szybkiego wykorzystania wspólnego modelu danych (mamy tutaj na myśli dane wejściowe) i natychmiastowego wygenerowania nowej konfiguracji. Zaczynamy luźno wykorzystywać termin *model danych*, lecz nie martw się, w rozdziale 5. poświęcimy nieco miejsca na wyjaśnienie, czym są modele danych.

Kiedy wykonujesz migrację, myśl o niej na bardziej abstrakcyjnym poziomie i przemyśl zadania niezbędne do przejścia z jednej platformy na drugą. Następnie sprawdź, co można by zrobić, aby zautomatyzować te zadania, ponieważ tylko Ty, a nie wielcy dostawcy sieciowi, masz motywację do tego, aby automatyzacja wykorzystująca sprzęt wielu różnych dostawców stała się rzeczywistością. Na przykład pomyśl o dodaniu sieci VLAN jak o abstrakcyjnym kroku — potem możesz się martwić o niskopoziomowe komendy na każdej platformie. Ważne jest to, że gdy rozpoczniesz adaptację automatyzacji, niesłychanie istotne stanie się to, abyś myślał o zadaniach i dokumentował je w zrozumiałym dla człowieka formacie, który jest niezależny od dostawców, zanim sięgniesz po klawiaturę, aby wprowadzić komendy wiersza poleceń lub napisać program (na każdą z platform).

Zarządzanie konfiguracją

Jak wspomnieliśmy wcześniej, zarządzanie konfiguracją jest najpospolitszym rodzajem automatyzacji, dlatego też nie poświęcimy tutaj na to zbyt wiele miejsca. Powinieneś być świadom, że kiedy wspominamy o zarządzaniu konfiguracją, mamy na myśli jej wdrażanie i wysyłanie do urządzeń oraz zarządzanie jej stanem. Wliczamy w to zarówno rzeczy tak podstawowe, jak zaopatrywanie sieci VLAN, jak i skomplikowane przepływy pracy, które konfigurują przełączniki na szczycie stojaka, firewalle, równoważniki ruchu i zaawansowane infrastruktury bezpieczeństwa w celu wdrożenia aplikacji trzeciopoziomowych.

Jak już widziałeś w kilku przypadkach automatyzacji sieci, które służą *tylko do odczytu*, musisz rozpocząć swoją podróż przez automatyzację sieci od wysyłania konfiguracji. Jeżeli spędzasz niezliczone godziny na przesyłaniu tych samych zmian na przestrzeni wielu routerów lub przełączników, możesz być temu pomysłowi przychylny!

W rzeczywistości liczba sposobów na rozpoczęcie podróży przez automatyzację sieci jest duża, lecz pamiętaj, że z wielką mocą wiąże się wielka odpowiedzialność. Co ważniejsze, nie zapomnij o tym, by uważnie przetestować swoje narzędzia automatyzacji przed ich wdrożeniem do środowiska produkcyjnego.

Następne typy automatyzacji sieci, o których opowiemy, wyrosły z automatyzacji procesu zbierania danych. Wybraliśmy kilka z nich w celu przedstawienia szerszego kontekstu, a zaczniemy od automatyzacji testów zgodności.

Zgodność

Tak jak w wielu formach automatyzacji, wprowadzanie zmian w konfiguracji z zastosowaniem jakiegokolwiek narzędzia automatyzacji postrzegane jest jako ryzyko. Choć wprowadzanie ręcznych zmian można uznać za ryzykowniejsze, o czym mogłeś przeczytać lub czego mogłeś sam doświadczyć, to możesz zacząć zbierać dane, monitorować je i budować konfigurację — są to działania *jedynie czytające dane* i akcje *niskiego ryzyka*. Jedynym scenariuszem niskiego ryzyka, w którym wykorzystywane są zebrane dane, są testy zgodności konfiguracji i weryfikacja konfiguracji. Czy wdrożona konfiguracja spełnia wymagania bezpieczeństwa? Czy wymagane sieci zostały skonfigurowane? Czy protokół XYZ został wyłączony? Jeżeli masz kontrolę nad wdrożonymi narzędziami, to łatwe staje się zweryfikowanie, co jest *prawdą*, a co *fałszem*. Proste jest rozpoczęcie od jednokrotnego sprawdzenia zgodności, a potem dokonywanie dodatkowych sprawdzeń w razie potrzeby.

Opierając się na poprawności tego, co sprawdzasz, możesz określić, co stanie się później — może zostanie to tylko zapisane w logach, a może zostanie przeprowadzona kompleksowa operacja, czyniąc Twoją aplikację zdolną do automatycznego radzenia sobie z problemami. Są to formy automatyzacji sterowanej zdarzeniami, o której opowiemy przy omówieniu narzędzi StackStorm i Salt w rozdziale 9. Przy automatyzacji sieci najlepiej jest zawsze zacząć od podstaw, ale trzeba być świadomym pełni jej możliwości. Na przykład, jeżeli po prostu zapisujesz w logach i drukujesz wiadomości, aby zobaczyć, jaka jest maksymalna jednostka transmisji (ang. *maximum transmission unit* — MTU), to jesteś też gotowy, kiedy tylko zechcesz, do jej automatycznej rekonfiguracji, jeżeli ta wartość jest dla Ciebie niezadowalająca. Wystarczyłoby tylko dodać kilka dodatkowych linijek do kodu odpowiedzialnego za zapisywanie do logów (wyświetlanie) wiadomości. Należy zacząć ostrożnie, ale zawsze trzeba myśleć o tym, co może przydać się w przyszłości.

Raportowanie

Gdy już zaczniesz zbierać dane, to może zechcesz też zacząć budować dostosowane przez siebie dynamiczne raporty. Może zwrócone dane staną się danymi wejściowymi w innych zadaniach zarządzania konfiguracją (zarówno opartą na zdarzeniach, jak i prostszą konfiguracją warunkową) lub może przydadzą Ci się w tworzeniu raportów.

Biorąc pod uwagę, że raporty można łatwo generować z szablonów połączonych z faktycznymi efemerycznymi danymi z urządzenia, które zostaną wprowadzone do szablonu, to proces tworzenia i wykorzystywania raportów jest tym samym procesem co ten używany do tworzenia szablonów konfiguracji, o którym wspominaliśmy wcześniej w tym rozdziale (z szablonami zapoznasz się szczegółowo w rozdziale 6.).

Z powodu mało złożonej natury korzystania z szablonów opartych na tekście możliwe jest tworzenie raportów w dowolnym formacie, np.:

- prostych plików tekstowych,
- plików ze znacznikami, które z łatwością przejrzysz w serwisie GitHub lub jakimś czytniku z obsługą znaczników,
- raportów HTML, które wdrażane są na serwerze sieciowym w celu łatwego przejrzenia.

Wszystko zależy od Twoich wymagań. Wspaniałą rzeczą jest to, że *automat sieciowy* jest zdolny do wygenerowania dokładnie takiego raportu, jakiego sobie zażyczysz. W rzeczy samej, możesz wykorzystać jeden zestaw danych do wygenerowania różnego rodzaju raportów, może nawet jakichś raportów technicznych i raportów wysokiego poziomu dla zarządu.

Teraz przyjrzymy się zautomatyzowanemu rozwiązywaniu problemów.

Rozwiązywanie problemów

Któż nie lubi być angażowany do spraw związanych z problemem zepsutego sprzętu, zwłaszcza kiedy powinien spać lub skupić się na czymś innym?

Gdy już uzyskasz dostęp do danych w czasie rzeczywistym i nie będziesz musiał ich przetwarzać ręcznie, automatyzacja rozwiązywania problemów stanie się rzeczywistością.

Pomyśl nad tym, *jak* rozwiązujesz problemy. Czy masz własną metodologię? Czy metodologia wszystkich członków Twojego zespołu jest spójna? Czy każdy sprawdza warstwę łącza danych przed rozwiązaniem problemów w warstwie sieci? Jakie kroki podejmujesz, aby rozwiązać dany problem?

Przyjrzyjmy się rozwiązywaniu problemów z OSPF:

- Czy wiesz, co jest wymagane, aby nawiązać sąsiedztwo pomiędzy dwoma urządzeniami?
- Czy jesteś zdolny odpowiedzieć na powyższe pytanie o drugiej nad ranem lub w trakcie urlopu na plaży?
- Może pamiętasz, że urządzenia muszą być w tej samej podsieci, mieć takie samo MTU, mieć spójne zegary, ale zapomniałeś, że muszą należeć do sieci OSPF tego samego typu.
- Czy naprawdę musimy pamiętać to wszystko i powiązane z tym komendy wiersza poleceń, aby odzyskać wszystkie dane?

A to jedynie *kilka* z rzeczy, które muszą pasować w przypadku routingu OSPF.

W każdym środowisku tego typu muszą być przeprowadzone testy kompatybilności. Możesz wyobrazić sobie uruchomienie skryptu lub wykorzystanie narzędzia do weryfikacji sąsiada OSPF zamiast przeprowadzenia całego tego procesu ręcznie? Co byś preferował?

OSPF to jedynie wierzchołek góry lodowej. Pomyśl o pozostałych pytaniach, też będących tylko takim wierzchołkiem:

- Czy potrafisz skorelować ze sobą odpowiednie wiadomości w logach, aby wiedzieć, co zaszło w sieci?
- Co z sąsiedztwem BGP? Jak formuje się to sąsiedztwo?
- Czy widzisz wszystkie trasy, które Twoim zdaniem powinny być w tablicy routingu?
- Co z konfiguracjami VPC i MLAG?
- Co z kanałami portów? Czy występują tutaj jakieś nieścisłości?
- Czy sąsiedzi pasują do konfiguracji kanałów portów (aż do poziomu vSwitcha)?
- Co z okablowaniem? Czy wszystkie przewody są poprawnie podłączone?

Nawet z tymi pytaniami jedynie delikatnie ocieramy się o możliwości, które dają zautomatyzowana diagnostyka i rozwiązywanie problemów.

Kiedy zaczniesz rozważać wszystkie możliwe rodzaje automatyzacji, wyobraź sobie system zamkniętej pętli, w którym dane są gromadzone przetwarzane i analizowane w sposób zautomatyzowany. Kiedy zacznie się to dziać jednocześnie, stanie się zamkniętą pętlą, całkowicie zmieniającą sposób, w jaki operujemy siecią i zarządzamy nią wewnątrz organizacji.

Jeżeli jesteś gwiazdą w swoim zespole inżynierów, to możesz pomyśleć o nawiązaniu współpracy z programistą lub przynajmniej zacząć dokumentować swoje przepływy pracy, dzięki czemu prostsze stanie się przekazywanie wiedzy i jej *pozyskiwanie*.

Gdy już rozpoczniesz swoją podróż przez automatyzację, będziesz mógł wreszcie porządnie sypiać. Zaznajom wszystkich ze zautomatyzowanymi przepływami pracy diagnostycznej.

Jak widzisz, automatyzacja sieci to dużo więcej niż tylko szybsze wdrażanie konfiguracji. Po przyjrzeniu się kilku różnym rodzajom automatyzacji zajmiemy się nowym tematem i rzucimy okiem na kilka różnych metod, którymi narzędzia automatyzacji i aplikacje komunikują się z urządzeniami sieciowymi, zaczynając od SSH, a kończąc na API NETCONF i API bazujących na HTTP.

Ewolucja płaszczyzny zarządzania od SNMP do API urządzeń

Jeżeli chcesz poprawić codzienną operacyjność i zarządzanie swoimi sieciami, ulepszeniu należy poddać interfejs, który stosujesz do komunikacji z zarządzanymi przez Ciebie urządzeniami. Ten interfejs to sposób, w jaki komunikujesz się zarówno Ty, jak i komunikują się narzędzia automatyzacji z urządzeniami sieciowymi w celu przeprowadzenia zautomatyzowanych akcji, takich jak gromadzenie danych i zarządzanie konfiguracją.

W tej sekcji przedstawimy przegląd różnych metod łączenia się z płaszczyzną zarządzania urządzeń sieciowych, zaczynając od SNMP, a potem przesuwając się w stronę metod znacznie nowocześniejszych, takich jak NETCONF i RESTful API. Później rzucimy okiem na wpływ ruchu *open networking* na operacyjność i automatyzację sieci.

Interfejs programowania aplikacji (API)

Jako inżynier sieci, wraz z kolejnymi krokami doskonalenia się musisz zaznajamiać się z API, a nie się ich obawiać. Pamiętaj, że API to tylko mechanizm, który wykorzystywany jest przez oprogramowanie komputerowe na jednym urządzeniu do tego, aby „rozmawiać" z oprogramowaniem komputerowym na innych urządzeniach. API są dzisiaj wykorzystywane praktycznie wszędzie w internecie — dostawcy sieciowi wreszcie poświęcają im więcej uwagi. Wkrótce zobaczymy, że API staną się podstawową metodą zarządzania urządzeniami sieciowymi.

API w szczegółach opisujemy w rozdziale 7.; ta sekcja dostarcza wyłącznie wysokopoziomowego przeglądu różnych rodzajów API, które występują w dzisiejszych urządzeniach sieciowych.

SNMP

Protokół SNMP był szeroko wdrażany przez ostatnie 20 lat w urządzeniach sieciowych. Nie powinien stanowić dla Ciebie nowości. Jest to protokół powszechnie wykorzystywany do pozyskiwania danych z urządzeń sieciowych, takich jak informacje o czasie działania czy wyłączenia, obciążeniu procesora, wykorzystaniu pamięci i interfejsów.

Aby korzystać z SNMP, wymagana jest obecność agenta SNMP na zarządzanym urządzeniu i system zarządzania siecią NMS, który służy jako *serwer* monitorujący i (lub) kontrolujący zarządzane urządzenia.

Każde z zarządzanych urządzeń udostępnia zestaw danych, który można zebrać i skonfigurować z wykorzystaniem agenta SNMP. Zestaw danych zarządzany przez SNMP jest opisywany i modelowany za pomocą baz informacji zarządzania. Jedynie kiedy MIB udostępnia pewną funkcjonalność, może ona być monitorowana lub zarządzana, wliczając w to zmiany konfiguracyjne poprzez SNMP. Jest to często przeoczana funkcjonalność; SNMP nie tylko wspiera żądania typu `GetRequest` dla monitoringu, lecz wspiera także żądania typu `SetRequest` do manipulowania obiektami i zmiennymi udostępnianymi przez MIB. Problemem jest to, że nie wszyscy dostawcy oferują pełne wsparcie dla zarządzania konfiguracją za pomocą SNMP; kiedy to robią, często wykorzystują własne, odpowiednio dostosowane MIB, spowalniając w ten sposób proces integracji z platformami zarządzania siecią.

Jak wspominaliśmy, SNMP istnieje od dziesięcioleci, lecz nie został zbudowany jako działający w czasie rzeczywistym programistyczny interfejs do urządzeń sieciowych. Już teraz słychać o dostawcach, którzy zapowiadają nadciągającą śmierć SNMP w kontekście zarządzania następnej generacji i narzędzi automatyzacji. Mimo to SNMP występuje na praktycznie każdym urządzeniu sieciowym, a do tego istnieją biblioteki Pythona do obsługi SNMP. Tak że jeśli musisz zebrać podstawowe informacje z dużej grupy różnego typu urządzeń sieciowych, użycie SNMP może wciąż stanowić rozsądne rozwiązanie.

Tak jak SNMP od lat wykorzystywano do monitoringu sieci, tak SSH/Telnet i wiersz poleceń od lat wykorzystywano do zarządzania konfiguracją. Spójrzmy teraz na SSH/Telnet i wiersz poleceń.

SSH/Telnet i wiersz poleceń

Jeżeli kiedykolwiek zarządzałeś jakimś urządzeniem sieciowym, to z pewnością wykorzystywałeś wiersz poleceń do wysyłania komend w celu przeprowadzania jakichś akcji na urządzeniu. Najpewniej wprowadzałeś polecenia przez konsolę i w ramach sesji Telnet i SSH. Jak wspomnieliśmy w rozdziale 1., rzeczywistość jest taka, że migracja z protokołu Telnet na SSH jest największą zmianą, która zaszła w ciągu ostatniej dekady w operacyjności sieciowej, a przecież sama zmiana nie dotyczyła bezpośrednio operowania urządzeniem. Zmiana miała za zadanie zwiększenie bezpieczeństwa, gdyż wprowadzała szyfrowanie do komunikacji z urządzeniem sieciowym.

Najistotniejszą rzeczą do zrozumienia w odniesieniu do zarządzania urządzeniami poprzez wiersz poleceń jest to, że wiersze poleceń stworzono dla ludzi. Osadzono je w urządzeniach, aby poprawić wygodę użytkowania dla ludzkich operatorów. Wiersz poleceń *nie* był przeznaczony do komunikacji maszyna-maszyna (tzn. do automatyzacji i oskryptowania sieci).

Jeżeli wywołasz polecenie `show` w wierszu poleceń urządzenia, otrzymasz surowy tekst. Nie ma w nim żadnej struktury. Najlepszą opcją *przetworzenia* odpowiedzi jest wykorzystanie *przetwarzania potokowego* (|) i słów kluczowych, takich jak `grep`, `include` i `begin`, do wyszukania odpowiedniej linijki w konfiguracji. Przykładem tego jest sprawdzenie opisu interfejsu za pomocą polecenia `show interface Eth1 | include description`. Oznacza to, że jeżeli chcesz wiedzieć, ile błędów CRC wystąpiło w interfejsie po wywołaniu `show interface` w ramach skryptu, zmuszony jesteś do wykorzystania tych samych typów wyrażeń regularnych lub przetworzenia ich w celu znalezienia tej informacji. Jest to nie do zaakceptowania.

Jeżeli jednak wiersz poleceń to jedyna dostępna opcja, to trzeba z niej korzystać. Dlatego też istnieje mnóstwo platform zarządzania siecią i skryptów zbudowanych w ostatnich dwóch dekadach. Narzędzia te służą do zarządzania operacjami i automatyzują operacje poprzez umożliwienie wykorzystania wiersza poleceń w ramach sesji SSH do radzenia sobie ze skryptami pobierającymi informacje lub ręcznym przetwarzaniem. To nie jest tak, że SSH (wiersz poleceń) czyni automatyzację niemożliwą; raczej czyni automatyzację niezwykle pracochłonną i podatną na błędy.

Dostawcy sieciowi zaczęli zdawać sobie z tego sprawę i teraz większość nowych platform sprzętowych posiada jakiś rodzaj API, który upraszcza komunikację maszyna-maszyna (wiele z nich jest niekompletnych, dlatego przetestuj uważnie API swojego ulubionego urządzenia). Uzyskuje się w ten sposób dużo prostsze podejście do automatyzacji, które jest bardziej dostosowane do powszechnych założeń wytwarzania oprogramowania.

Po krótkim przejrzeniu powszechnych protokołów, takich jak SSH i SNMP, zwrócimy uwagę na protokół NETCONF, który jest coraz popularniejszy, gdyż odnosi się do automatyzacji sieci.

NETCONF

NETCONF to protokół warstwy zarządzania siecią. Na najwyższym poziomie może być porównywany do SNMP, jako że oba protokoły wykorzystywane są do wprowadzania zmian w konfiguracji i do pozyskiwania danych od urządzeń sieciowych.

Różnice dotyczą szczegółów, rzecz jasna. Kilku najistotniejszym sprawom przyjrzymy się tutaj, więcej miejsca przeznaczymy na NETCONF w rozdziale 7.

- NETCONF to protokół połączeniowy; często wykorzystuje SSH do transportu.
- Dane wymieniane pomiędzy klientem NETCONF (narzędzie, skrypt automatyzacji) a serwerem NETCONF (urządzenie sieciowe) kodowane są w formacie XML. Nie martw się, jeżeli nie zapoznałeś się dotąd z formatem XML; opiszemy go w rozdziale 5.
- Zdalne wywołania procedur (ang. *remote procedure calls* — RPC) kodowane są w ramach dokumentu XML wysyłanego do urządzenia, a urządzenie procedury te przetwarza. Element `<rpc>` jest wykorzystywany do enkapsulacji żądania NETCONF wysyłanego z klienta do serwera. W kontekście tego pomyśl o tych zdalnych wywołaniach procedur jak o wykonywaniu opracowanych zawczasu operacji na urządzeniu. RPC są dla klienta metodą na zakomunikowanie serwerowi, jakiej struktury i jakiego typu żądania się wykonuje.
- Wspierane RPC dopasowywane są bezpośrednio do wspieranych *operacji* i *możliwości* NETCONF dla danych urządzeń. Dla przykładu, dokonując zmiany w urządzeniu, wykonujesz operację `edit-config`. Jeżeli pozyskujesz dane, to wykonujesz operację `get` lub `get-config`. Operacje te zostają osadzone wewnątrz dokumentu XML, wewnątrz elementu `<rpc>` wysyłanego do urządzenia.

Dodatkowo NETCONF ma swoją wartość w tym, że wspiera zmiany opierające się na transakcjach. Oznacza to, że jeżeli wprowadzasz więcej niż jedną zmianę w trakcie danej sesji NETCONF lub w ramach jednego dokumentu XML i wprowadzenie jednej z tych zmian zawiedzie, to cały komplet zmian *nie* zostanie zastosowany na urządzeniu (rzecz jasna, tego typu ustawienia można także zmienić). Stoi to w opozycji do sekwencyjnego wysyłania komend przez wiersz poleceń, kończącego się tylko częściową zmianą konfiguracji, czego przyczyną jest literówka lub niepoprawna komenda.

Było to proste wprowadzenie do protokołu NETCONF; jak wspomnieliśmy wcześniej, zagłębimy się w temat protokołu NETCONF w rozdziale 7.

Warto zwrócić uwagę na to, że samo wsparcie protokołu NETCONF (lub jakiejkolwiek innej powszechnej metody transportu) przez dwa różne urządzenia nie oznacza, że są one kompatybilne z perspektywy narzędzia lub programisty. Nawet przy założeniu, że oba urządzenia wspierają tę samą funkcjonalność i możliwości NETCONF, to niestety sposób, w jaki dane są modelowane, i tak zazwyczaj jest zależny od dostawcy. Modelowanie danych to sposób, w jaki urządzenie reprezentuje dane i status konfiguracji. O powszechnych metodach reprezentacji danych w formacie JSON i XML oraz o modelach YANG i powszechnym języku modelowania danych będzie mowa w rozdziale 5.

API typu RESTful

REST (skrót od *REpresentational State Transfer*) jest stylem wykorzystywanym do projektowania i wytwarzania aplikacji sieciowych. Dlatego też systemy, które implementują zasady architektur opartych na REST i im podlegają, nazywane są systemami RESTful.

Z perspektywy sieciowej najpowszechniejsze urządzenia, które udostępniają API i podlegają stylowi architektury REST, to kontrolery sieciowe. Istnieją też urządzenia sieciowe, które oprócz RESTful udostępniają powszechne API bazujące na HTTP.

Mimo że terminy **REST** i **RESTful API** są nowe z punktu widzenia sieci, to tak naprawdę wchodzisz w interakcję z wieloma systemami RESTful codziennie, kiedy przeglądasz internet z wykorzystaniem przeglądarki internetowej. Powiedzieliśmy, że REST jest stylem wykorzystywanym do wytwarzania aplikacji sieciowych. Styl ten opiera się na bezstanowym modelu klient-serwer, w którym klient śledzi postęp sesji i żadna informacja o stanie lub kontekście nie jest przetrzymywana na serwerze. A najlepsze w tym wszystkim jest to, że podległy protokół transportowy to zazwyczaj HTTP. Czyż nie jest to system, jakich wiele w internecie?

Oznacza to, że API RESTful operują podobnie jak systemy bazujące na HTTP. Najpierw potrzebujesz serwera sieciowego dostępnego przez URL (może to być np. *kontroler SDN* lub *urządzenie sieciowe*, z którym chcesz się komunikować), a następnie musisz wysłać odpowiednie żądanie HTTP na ten URL.

Na przykład, jeżeli chcesz pozyskać listę urządzeń z kontrolera SDN, wystarczy, że wyślesz żądanie HTTP `GET` na odpowiedni URL urządzenia, który może przyjąć taką postać: *http://1.1.1.1/v1/devices*. Odpowiedź, którą uzyskasz, będzie jakiegoś typu ustrukturyzowanymi danymi, takimi jak formaty XML lub JSON (o których powiemy w rozdziale 5.).

Istnieje jeszcze kilka spraw, których nie poruszyliśmy, takich jak uwierzytelnienie, kodowanie danych i to, jak wysłać żądanie HTTP, jeżeli wprowadzasz zmianę konfiguracyjną (HTTP `PUT`/`POST`/`PATCH`). Ta sekcja to jedynie krótkie, wysokopoziomowe wprowadzenie do REST i RESTful API, które omówimy szczegółowo w rozdziale 7.

Następnie spojrzymy pokrótce na wpływ *open networkingu* na zarządzanie urządzeniami.

Wpływ open networkingu

Istnieje rosnący trend robienia wszystkiego *open* (otwartym) — *open source*, *open networking*, otwarte API, OpenFlow, Open Compute, Open vSwitch, OpenDaylight, OpenConfig — lista jest długa. Co prawda definicja *otwartości* jest wciąż dyskutowana, ale jedna rzecz jest pewna: ruch *open networkingu* stara się usprawnić, co tylko się da, w przypadku operacyjności i automatyzacji sieci.

Wraz z działaniami tego ruchu zaczęliśmy dostrzegać drastyczne zmiany w urządzeniach sieciowych, a jest to przecież podstawowy powód, dla którego piszemy tę książkę.

Po pierwsze, obecnie wiele urządzeń wspiera Pythona wewnątrz pudełka. Oznacza to, że możesz od razu sięgnąć po dynamiczny interpreter Pythona (ang. *Python Dynamic Interpreter*) i wykonywać skrypty Pythona lokalnie na każdym urządzeniu sieciowym. Pythona opisujemy szczegółowo w rozdziale 4., gdzie dokładnie dowiesz się, o co nam chodzi.

Po drugie, wiele urządzeń wspiera teraz bardziej rozbudowane API, inne niż SNMP i SSH. Wśród nich na przykład dopiero co przez nas przedstawione NETCONF i RESTful API bazujące na HTTP. Większość urządzeń sieciowych (nie starszych niż 24 miesiące i wyposażonych w nowe systemy operacyjne) wspiera oba lub przynajmniej jedno z wymienionych API. O API urządzeń więcej piszemy w rozdziale 7.

Na koniec — urządzenia sieciowe udostępniają wiele wewnętrznych zasobów Linuxa, które były w przeszłości skrywane przed administratorami sieci. Obecnie możesz sięgnąć po powłokę **bash** na urządzeniach sieciowych i wywoływać polecenia, takie jak `ifconfig`, pisać skrypty bash i instalować narzędzia zarządzania i konfiguracji poprzez menedżery pakietów, takie jak `apt` i `yum`. Dowiesz się o tych rzeczach w rozdziale 3.

Co prawda *open networking* nie zawsze oznacza intensywniejszą współpracę pomiędzy różnymi rozwiązaniami, ale oczywiste jest, że urządzenia sieciowe i kontrolery otwierają się na operowanie na nich w dużo bardziej programowalny sposób, lepiej dostosowany do ulepszonej automatyzacji sieci. Jeszcze wiele lat temu rzeczy takie jak API NX-API firmy Cisco czy eAPI firmy Arista, IOS-XE RESTCONF/NETCONF firmy Cisco lub jakikolwiek inny nowoczesny kontroler SDN wyposażone w API nie istniały. Zysk dla operatorów jest taki, że dzięki zastosowaniu API mogą przejąć kontrolę nad swoimi sieciami i zredukować nieefektywność związaną z operacyjnością.

Automatyzacja sieci w erze SDN

Teraz spojrzymy na to, jak istotna jest automatyzacja sieci mimo wdrażania kontrolerów, takich jak OpenDaylight czy nawet produkty komercyjne, np. Cisco ACI lub VMware NSX. Operacje, które kontrolery przeprowadzają w sieci, np. w obrębie sterowania lub zarządzania zasadami i konfiguracjami, nie są istotne w tej sekcji.

Kontrolery coraz częściej stają się częścią architektur następnej generacji. Dostawcy, tacy jak Cisco, Juniper, VMware, Big Switch, Plexxi, Nuage, Viptela, oferują platformy kontrolerów dla rozwiązań następnej generacji; są także dostępne kontrolery *open source*, takie jak OpenDaylight i OpenContrail.

Niemal każdy kontroler dostępny na rynku udostępnia północny (ang. *nortbound*) interfejs RESTful API, czyniąc kontrolery niesamowicie proste w automatyzacji. Mimo że kontrolery same w sobie upraszczają zarządzanie i zwiększają widoczność urządzeń, to wciąż możesz wprowadzać zmiany ręcznie poprzez GUI kontrolera, co jest podatne na błędy. Jeżeli zostało wdrożonych kilka kontrolerów, bez względu na to, czy od tego samego dostawcy, czy od różnych dostawców, to problem ręcznych zmian, rozwiązywania problemów i zbierania danych i tak nie znika.

Na zakończenie tego rozdziału trzeba zauważyć, że nawet w tej nowej erze architektur SDN i rozwiązań sieciowych opartych na kontrolerach wciąż potrzebne są automatyzacja, lepsza operacyjność i bardziej przewidywalne rezultaty.

Podsumowanie

Rozdział ten przedstawił przegląd korzyści wynikających z różnych rodzajów automatyzacji sieci; wprowadzenie do powszechnych API urządzeń, takich jak SNMP, wiersz poleceń/SSH, a co ważniejsze, API NETCONF i RESTful oraz krótką wzmiankę o języku modelowania sieci YANG, który opiszemy szczegółowo w rozdziale 5.

Zakończyliśmy krótkim przejrzeniem wpływu ruchu open networking na operacyjność i automatyzację sieci. Na koniec poruszyliśmy temat zalet automatyzacji sieci, które są oczywiste, mimo że są wdrażane kontrolery SDN.

W następnych rozdziałach zgłębimy wspomniane technologie, pokażemy praktyczne, z życia wzięte przykłady i przyjrzymy się zasobom ludzkim i kulturze organizacji wymaganym do przeprowadzenia automatyzacji. Ludziom i kulturze organizacji jest poświęcony przede wszystkim Rozdział 11.

Skorowidz

D

E

F

G

H

I

J

K

L

Ł

M

N

O

P

R

S

Ś

T

U

V

W

X

Y

Z

Ż

O'REILLY®

Sieć zautomatyzowana i programowalna — najlepszy przyjaciel admina!

Programowalna i zautomatyzowana sieć upraszcza pracę jej administratora. Rozwój technologii radykalnie komplikuje takie zadania jak zarządzanie i operowanie sprzętem sieciowym, topologiami sieci i połączeniami sieciowymi. Trzeba tu mieć na uwadze systemy operacyjne, nowe metodologie oraz narzędzia. W takich warunkach zarządzanie większą czy nieco bardziej złożoną siecią wyłącznie za pomocą działań manualnych jest obarczone sporym ryzykiem. Profesjonalny inżynier sieciowy musi dziś dobrze orientować się w świecie programowalności i automatyzacji sieci. Powinien poznawać nowe protokoły, technologie, modele dostarczania i pojawiające się w związku z nimi potrzeby biznesowe.

W tej książce znajdziesz solidne podstawy pozwalające zapewnić sieci programowalność i zautomatyzowanie jej pracy. Dowiesz się, jakie narzędzia i umiejętności będą potrzebne do dokonania tego kluczowego przekształcenia w sieć nowej generacji. W bardzo przystępny i praktyczny sposób wyjaśniono, jak korzystać z takich technologii jak Linux, Python, JSON i XML, aby programowo zautomatyzować pracę systemu. Opisano koncept modeli danych, podstawy języka YANG oraz najważniejsze technologie związane z API. Sporo miejsca poświęcono narzędziom open source służącym do automatyzacji pracy sieci. Znalazły się tu również informacje o interfejsach macvlan, sieciach wykorzystujących maszyny wirtualne, sieciowych przestrzeniach nazw oraz o bibliotece Pythona NAPALM i jej integracji z narzędziami: Ansible, Salt i StackStorm.

W książce między innymi:

- powstanie sieci sterowanych programowo
- technologie automatyzacji sieci
- Linux i Python a technologie sieciowe
- praca z szablonami konfiguracji sieciowej
- kontrola źródła w pracy z niektórymi serwisami online
- prosty przepływ pracy w automatyzacji sieci

Jason Edelman — jest inżynierem sieciowym. Specjalizuje się w zagadnieniach oprogramowania, praktykach jego wytwarzania i konwergencji z inżynierią sieciową. Prowadzi małą firmę consultingową Network to Code.

Scott S. Love — jest inżynierem architektury w firmie VMware. Jego ulubioną dziedziną jest przetwarzanie w chmurze i wirtualizacja sieci. Napisał kilka książek na temat vSphere i OpenStack.

Matt Oswalt — jest sieciowym deweloperem oprogramowania. Zajmuje się technicznymi i nietechnicznymi wyzwaniami współdziałania oprogramowania z infrastrukturą sieci.

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

ISBN 978-83-283-5045-8
9 788328 350458

INFORMATYKA W NAJLEPSZYM WYDANIU

Cena: 89,00 zł