Ansible, AWX, Semaphore, Rudder, Terraform: co do czego

Poziom: podstawy **Dotyczy: narzędzia automatyzacji infrastruktury, stan na 2026**

Czas: ~20 min

Pięć narzędzi w tytule odpowiada na dwa różne pytania — „jak skonfigurować to, co już istnieje" i „jak w ogóle stworzyć to, co ma istnieć" — i mylenie tych pytań jest najczęstszym powodem sięgnięcia po niewłaściwe narzędzie.

Skróty użyte na tej stronie (3)

Skrót	Co znaczy
API	Application Programming Interface: sposób, w jaki programy rozmawiają ze sobą
AWX	otwarta wersja Ansible Tower: przeglądarkowy panel do uruchamiania Ansible
SSH	Secure Shell: szyfrowane zdalne logowanie do serwera

Z życia wzięte

Z ŻYCIA WZIĘTE

Nowy handlowiec potrzebuje w środę stanowiska w Swarzędzu, a Tomek próbuje to załatwić jednym playbookiem Ansible: kontem w domenie, kontem pocztowym i maszyną wirtualną naraz. Ansible owszem, tworzy konto i skrzynkę, ale maszyny wirtualnej nie da się w ten sposób sensownie opisać ani odtworzyć, gdyby trzeba było postawić drugą taką samą. Po rozdzieleniu zadania: Terraform tworzy maszynę, a Ansible dopiero potem ją konfiguruje, cały proces staje się powtarzalny.

Zarządzanie konfiguracją a provisioning

> **Pojęcie: zarządzanie konfiguracją (configuration management).** Utrzymywanie pożądanego stanu **już istniejących** maszyn — pakiety, które mają być zainstalowane, pliki konfiguracyjne, które mają mieć określoną treść, usługi, które mają działać. Zarządzanie konfiguracją zakłada, że serwer już istnieje (fizyczny, wirtualny, wynajęty u dostawcy chmury) i zajmuje się wyłącznie tym, co się na nim dzieje od środka.

> **Pojęcie: provisioning.** Tworzenie **samych zasobów** infrastruktury — maszyn wirtualnych, sieci, dysków, kont w chmurze — zanim w ogóle zacznie się mówić o tym, co ma być na nich zainstalowane. Provisioning odpowiada na pytanie „ile serwerów, jakiej wielkości, w jakim regionie", nie na pytanie „co na nich uruchomić".

To rozróżnienie tłumaczy, dlaczego w jednym środowisku spotyka się zwykle dwa narzędzia naraz, nie jedno — jedno tworzy maszynę, drugie ją potem konfiguruje. Próba użycia narzędzia do provisioningu jako narzędzia do zarządzania konfiguracją (albo odwrotnie) zwykle kończy się prowizorką gorszą niż użycie dwóch właściwych narzędzi razem.

Push a pull

> **Pojęcie: model push.** To centralny węzeł zarządzający inicjuje połączenie do zarządzanej maszyny i wykonuje na niej zmiany — maszyna zarządzana nie musi mieć żadnego stale działającego agenta, wystarczy, że jest osiągalna (najczęściej po SSH) w momencie, gdy węzeł zarządzający chce coś zmienić.

> **Pojęcie: model pull.** To zarządzana maszyna, z zainstalowanym na stałe agentem, sama łączy się okresowo do centralnego serwera polityk i pobiera aktualną definicję pożądanego stanu, po czym stosuje ją lokalnie — węzeł centralny nigdy nie inicjuje połączenia do maszyny, tylko czeka, aż ta się zgłosi.

Różnica ma konsekwencje praktyczne. Model push nie wymaga niczego zainstalowanego trwale na zarządzanej maszynie i pozwala uruchomić zmianę na żądanie, w dowolnym momencie — ale wymaga, żeby węzeł zarządzający miał w danej chwili dostęp sieciowy do wszystkich maszyn

naraz. Model pull sam naprawia rozjazd konfiguracji nawet bez ręcznej interwencji (agent i tak zgłosi się o kolejną porcję polityk zgodnie z harmonogramem) i lepiej znosi maszyny czasowo odcięte od sieci, ale zmiana nie jest natychmiastowa — czeka na najbliższy cykl agenta.

Tabela narzędzi

Narzędzie	Model	Język opisu	Agent czy bezagentowe	Typowe zastosowanie
Ansible	push	YAML (playbooki)	bezagentowe (SSH)	konfiguracja serwerów, wdrożenia aplikacji, zadania ad-hoc
AWX / Semaphore	push (nad Ansible)	YAML + UI/API	bezagentowe (SSH, przez Ansible)	uruchamianie playbooków z harmonogramem, uprawnieniami i historią z poziomu UI
Rudder	pull	reguły i dyrektywy (własny DSL, technique)	agent na każdym zarządzanym węźle	ciągłe wymuszanie zgodności konfiguracji, raportowanie odchyleń
Terraform	— (provisioning, nie zarządzanie konfiguracją)	HCL (deklaratywny)	bezagentowe (wywołania API dostawcy)	tworzenie i wersjonowanie infrastruktury — maszyn, sieci, kont w chmurze



Terraform tworzy serwer, Ansible go konfiguruje na żądanie, a agent Rudder na tym samym serwerze cyklicznie pilnuje zgodności z serwerem polityk (kliknięcie otwiera pełny rozmiar)

AWX i Semaphore nie są osobnymi silnikami automatyzacji — to warstwa UI i API nad tym samym Ansible, opisana szerzej na stronie AWX i Semaphore. Terraform stoi w tabeli obok pozostałych

narzędzi, ale odpowiada na inne pytanie — nie „jak skonfigurować", tylko „co w ogóle ma istnieć", zanim jakiekolwiek zarządzanie konfiguracją ma sens. Szczegóły składni i codziennej pracy z każdym z nich rozwijają osobne strony: Ansible: podstawy operacyjne, Terraform: podstawy, Rudder.

Idempotentność jako wspólny mianownik

> **Pojęcie: idempotentność.** Właściwość operacji, dzięki której jej wielokrotne wykonanie daje dokładnie taki sam efekt końcowy, jak wykonanie jej raz — uruchomienie tego samego playbooka albo tej samej reguły Ansible, Rudder czy Terraform drugi raz z rzędu na maszynie, która już jest w pożądanym stanie, nie zmienia niczego, tylko potwierdza zgodność. To jest cecha, której wszystkie cztery narzędzia z tabeli wyżej pilnują z zasady, mimo zupełnie różnych modeli działania — bez niej żadne z nich nie nadawałoby się do bezpiecznego, powtarzalnego stosowania. Pełne omówienie tej właściwości, razem z częstymi pułapkami przy pisaniu nieidempotentnych skryptów, stoi na stronie Idempotentność: skrypt, który wolno uruchomić dwa razy.

Częste błędy

Użycie Terraform do zmiany konfiguracji wewnątrz już istniejącej maszyny. Terraform potrafi wywołać skrypt startowy przy tworzeniu zasobu (`user_data`, `provisioner`), ale nie jest zaprojektowany do ciągłego utrzymywania zgodności konfiguracji już działającej maszyny — do tego służy Ansible albo Rudder, uruchamiane osobno, już po tym, jak Terraform stworzył sam zasób.

Traktowanie AWX jako czegoś innego niż Ansible. Zespół, który nie ma dostępu do serwera AWX, wciąż może wykonać ten sam playbook bezpośrednio komendą `ansible-playbook` z linii poleceń — AWX dokłada UI, harmonogram i historię wykonań, nie nową logikę automatyzacji.

Wybór modelu pull tam, gdzie liczy się natychmiastowość zmiany. Rudder naprawi odchylenie konfiguracji przy najbliższym cyklu agenta, nie w chwili wprowadzenia reguły — do zmian, które muszą wejść w życie od razu (np. pilna łatka bezpieczeństwa), lepiej pasuje model push.

□□ Ćwiczenia

> Zadania sprawdzające.

1. Wyjaśniamy różnicę między zarządzaniem konfiguracją a provisioningiem na przykładzie utworzenia nowego serwera aplikacyjnego w chmurze i zainstalowania na nim aplikacji.
2. Wskazujemy, które z pięciu narzędzi z tabeli pracują w modelu push, a które w modelu pull, i uzasadniamy jedną praktyczną konsekwencję tej różnicy.
3. Zespół chce, żeby odchylenie konfiguracji od polityki bezpieczeństwa było wykrywane i naprawiane automatycznie, nawet bez ręcznej interwencji, kosztem tego, że naprawa nie musi być natychmiastowa. Wskazujemy, które narzędzie z tabeli pasuje do tego wymagania najlepiej.
4. Wyjaśniamy, dlaczego AWX nie pojawia się w tabeli jako osobny „silnik", tylko jako warstwa nad innym narzędziem.

Powiązane: Ansible: podstawy operacyjne · AWX i Semaphore · Terraform: podstawy · Rudder · Idempotentność: skrypt, który wolno uruchomić dwa razy

---

Revision #37
Created 2026-09-25 07:40:04 UTC by Tomasz Konieczny
Updated 2026-10-26 14:13:26 UTC by Tomasz Konieczny