Instalacja Terraform albo OpenTofu i stan w bezpiecznym miejscu

Poziom: podstawy **Dotyczy: Terraform 1.16.x, OpenTofu 1.12.x jako alternatywa**

Czas: ~30 min

Efekt: zainstalowany `terraform` albo `tofu` na stacji administratora i na `app01`, pierwszy `terraform init` przechodzący bez błędu i stan zapisywany w backendzie zdalnym, nie na lokalnym dysku ani w repozytorium Git.

Skróty użyte na tej stronie (6)

Skrót	Co znaczy
API	Application Programming Interface: sposób, w jaki programy rozmawiają ze sobą
AWS	Amazon Web Services: chmura Amazonu
BSL	Business Source License: licencja, na ktora HashiCorp zmienil w 2023 roku Terraform, ograniczajaca prawo do oferowania narzedzia jako konkurencyjnej platnej uslugi
CI	Continuous Integration: automatyczne budowanie i testowanie kodu po każdej zmianie; w ITIL także configuration item, czyli element konfiguracji: każdy składnik usługi zapisany w CMDB
CLI	Command Line Interface: praca przez wpisywanie komend w terminalu
MPL	Mozilla Public License: otwarta licencja, na której HashiCorp wydawał swoje programy do sierpnia 2023 roku i na której jest dziś Talos Linux

Zanim zaczniemy

> **Z ŻYCIA WZIĘTE**
>
> Zespół Oli dostaje zielone światło na pilotaż nowej usługi wewnętrznej i potrzebuje trzech maszyn wirtualnych na klastrze `hv01`/`hv02`, identycznie skonfigurowanych, do piątku. Ręczne klikanie przez konsolę Hyper-V Managera trzy razy z rzędu, z tymi samymi ustawieniami sieci

> i dysków, to dokładnie ten rodzaj powtarzalnej pracy, po który sięgamy po Terraform, nie po Ansible: te maszyny jeszcze nie istnieją.

Komendy wykonujemy z CLI bash (na Rocky Linux) albo z PowerShell (na Windows), z uprawnieniami wystarczającymi do instalacji pakietów. Instalujemy narzędzie na dwóch maszynach z tabeli w 08-swiat-zielonej: stacji administratora do pracy ręcznej i na `app01` (10.20.10.41), skąd później uruchamia je też Semaphore albo pipeline CI.

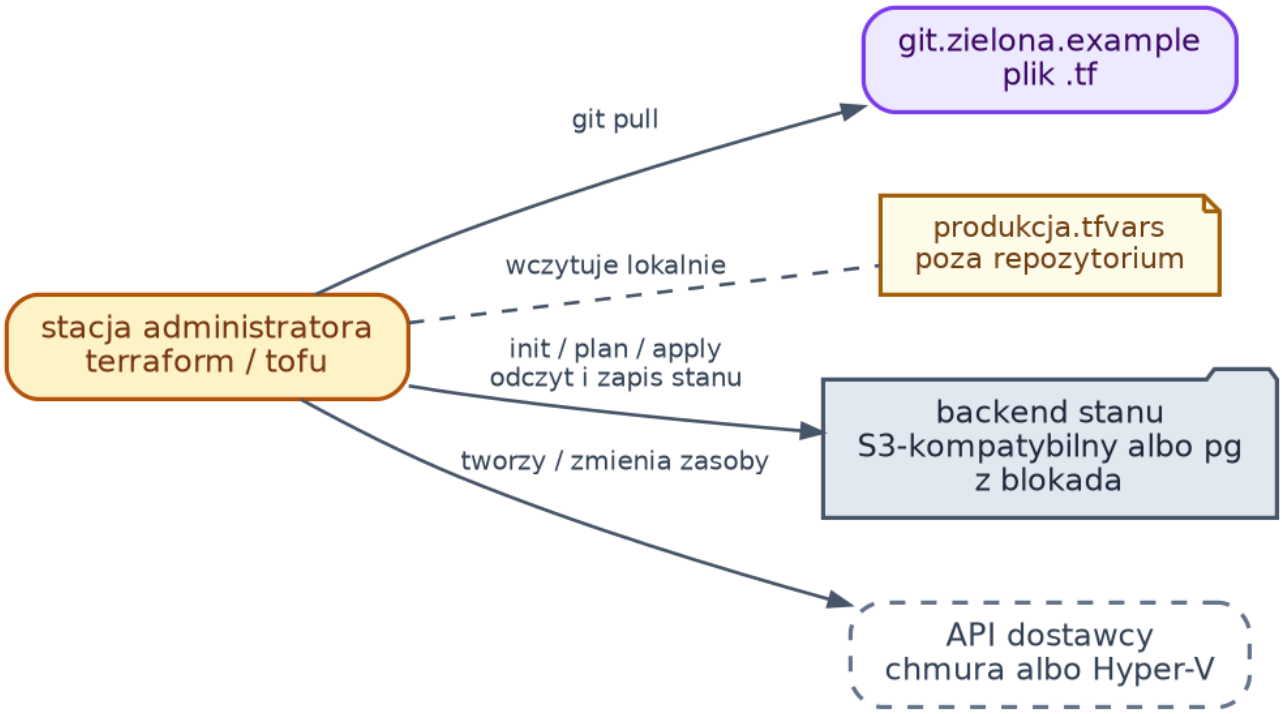
Wymaganie	Uwaga
Uprawnienia administracyjne na maszynie instalującej	`sudo` na Linuksie, konto administratora na Windows
Dostęp do internetu albo do wewnętrznego lustra pakietów	pobieramy binarkę z repozytorium producenta
Miejsce na backend stanu	Postgres na `pg01` albo magazyn S3-kompatybilny, patrz niżej

Terraform czy OpenTofu

Terraform i OpenTofu czytają te same pliki `.tf` i obsługują te same komendy, bo OpenTofu powstał jako fork Terraform sprzed zmiany licencji, opisanej na stronie Automatyzacja: jak przestaliśmy klikać te same rzeczy na dwudziestu serwerach. Wybór między nimi to dziś głównie kwestia licencji i tego, kto ma nadzór nad projektem, nie różnic w codziennej pracy.

	Terraform	OpenTofu
Licencja	Business Source License, ograniczająca oferowanie go jako konkurencyjnej usługi	Mozilla Public License 2.0, w pełni otwarta
Kto nadzoruje	HashiCorp (IBM)	fundacja Linux Foundation
Zgodność plików `.tf`	punkt odniesienia	wsteczna zgodność z Terraform sprzed forka, nowe funkcje mogą się rozjeżdżać z czasem
Aktualny stan	sprawdzamy w dokumentacji, licencja i warunki użycia bywają zmieniane	sprawdzamy w dokumentacji projektu na opentofu.org

Zielona nie ma dziś powodu, żeby świadczyć komercyjną usługę konkurencyjną wobec HashiCorp, więc licencja BSL sama w sobie jej nie przeszkadza. Poniżej pokazujemy instalację obu, wybieramy jedno.



Stacja administratora uruchamia terraform albo tofu, wczytuje pliki z Gitea i sekrety z tfvars, zapisuje stan w backendzie z blokadą, woła API dostawcy (kliknięcie otwiera pełny rozmiar)

Przebieg

Instalacja Terraform na Rocky Linux z repozytorium HashiCorp

```
sudo yum install -y yum-utils shadow-utils
sudo yum-config-manager --add-repo https://rpm.releases.hashicorp.com/RHEL/hashicorp.repo
sudo yum -y install terraform
```

Sprawdzenie wersji:

```
terraform -version
```

Przykład:

```
Terraform v1.16.4
on linux_amd64
```

Instalacja OpenTofu na Rocky Linux, oficjalnym skryptem

```
curl --proto '=https' --tlsv1.2 -fsSL https://get.opentofu.org/install-opentofu.sh -o install-opentofu.sh
chmod +x install-opentofu.sh
./install-opentofu.sh --install-method rpm
rm -f install-opentofu.sh
```

> Skrypt instalacyjny warto przejrzeć przed uruchomieniem, tak jak podpowiada sama dokumentacja OpenTofu: `less install-opentofu.sh` zanim damy mu prawo wykonania.

Binarka instaluje się pod nazwą `tofu`, nie `terraform`:

```
tofu -version
```

Przykład:

```
OpenTofu v1.12.3
on linux_amd64
```

Instalacja na Windows, przez winget

Na stacji administratora z Windows 11:

```
winget install Hashicorp.Terraform
```

Weryfikacja tym samym poleceniem `terraform -version` co na Linuksie, z terminala PowerShell.

Pierwsze polecenia

Pełny opis `init`, `plan`, `apply` i pracy ze stanem stoi na stronie Terraform: podstawy, identycznie dla `terraform` i dla `tofu`, bo różni je tylko nazwa binarki. Tutaj tylko pierwsze sprawdzenie, że instalacja działa, w katalogu z jednym prostym plikiem `.tf`:

```
terraform init
terraform validate
```

Dlaczego stan nie leży w Git

Plik stanu **Pojęcie: state** opisuje strona Terraform: podstawy. W skrócie: zawiera surowe dane o utworzonych zasobach, często z wartościami uznawanymi za wrażliwe w postaci jawnej, i wymaga blokady przy pracy zespołowej, której Git w ogóle nie oferuje. Dwie osoby z zespołu Zielonej, każda z osobnym lokalnym stanem tego samego projektu, prędzej czy później nadpiszą sobie nawzajem zmiany.

Backend: Postgres na pg01

Zielona ma już działający serwer PostgreSQL na `pg01`, więc backend `pg` nie wymaga stawiania niczego nowego, tylko jednej bazy:

```
CREATE DATABASE terraform_state;
CREATE USER terraform WITH ENCRYPTED PASSWORD 'HasloDoBazy';
GRANT ALL PRIVILEGES ON DATABASE terraform_state TO terraform;
```

```
terraform {
  backend "pg" {
    conn_str = "postgres://terraform:HasloDoBazy@pg01:5432/terraform_state"
  }
}
```

Blokada w backendzie `pg` działa od razu, przez mechanizm blokad samego PostgreSQL, bez dodatkowej konfiguracji.

Backend: magazyn S3-kompatybilny

Tam, gdzie Zielona wolałaby magazyn obiektowy zamiast bazy (na przykład na `bkp01`, obok reszty kopii zapasowych), backend `s3` obsługuje też magazyny inne niż Amazon, przez wskazanie własnego adresu:

```
terraform {
  backend "s3" {
    bucket         = "NazwaBucketa-terraform-state"
    key            = "NazwaProjektu/terraform.tfstate"
```

```
    region         = "eu-central-1"
    use_path_style = true
    use_lockfile   = true
    skip_region_validation      = true
    skip_credentials_validation = true
    skip_requesting_account_id  = true

    endpoints = {
      s3 = "https://bkp01.zielona.example:9000"
    }
  }
}
```

`use_lockfile` włącza blokadę stanu wprost w magazynie S3-kompatybilnym, bez osobnej usługi do samych blokad, którą starsze poradniki opisują jako wymaganą. Wartość `skip_requesting_account_id` i pozostałe flagi `skip_*` wyłączają sprawdzenia specyficzne dla prawdziwego AWS, których magazyn inny niż Amazon i tak by nie obsłużył.

Sekrety poza repozytorium

Plik `.tfvars` z hasłami i kluczami dostawcy nie trafia do repozytorium Gitea razem z resztą kodu `.tf`:

```
produkcja.tfvars
```

```
echo "*.tfvars" >> .gitignore
```

Zastosowanie wartości z takiego pliku przy `apply` opisuje strona Terraform: podstawy (`-var-file`); tutaj pilnujemy tylko, żeby sam plik nigdy nie trafił do `git add`.

Weryfikacja

Sprawdzamy, że backend faktycznie przyjął połączenie i zablokował stan podczas operacji:

```
terraform init
```

Przykład:

```
Initializing the backend...

Successfully configured the backend "pg"! Terraform will automatically
use this backend unless the backend configuration changes.

Terraform has been successfully initialized!
```

Sprawdzamy, że na dysku lokalnym nie ma pliku `terraform.tfstate` z prawdziwymi danymi: powinien tam zostać co najwyżej pusty wpis wskazujący na backend, nie sam stan.

```
ls -la terraform.tfstate 2>/dev/null || echo "brak lokalnego pliku stanu, tak jak powinno byc"
```

Wycofanie zmian

Odinstalowanie Terraform:

```
sudo dnf remove terraform -y
sudo rm -f /etc/yum.repos.d/hashicorp.repo
```

Odinstalowanie OpenTofu (metoda rpm):

```
sudo dnf remove tofu -y
```

Na Windows:

```
winget uninstall Hashicorp.Terraform
```

Sam backend (baza `terraform_state` na `pg01` albo bucket na `bkp01`) usuwamy osobno, dopiero gdy mamy pewność, że żaden projekt już go nie używa.

Gdy coś nie zadziała

Objaw	Przyczyna	Co robimy
`terraform init` zgłasza błąd połączenia z backendem `pg`	`pg01` nieosiągalny z maszyny albo złe hasło w `conn_str`	sprawdzamy `psql -h pg01 -U terraform -d terraform_state`, poprawiamy `conn_str`

Objaw	Przyczyna	Co robimy
`Error: Backend configuration changed` przy kolejnym `init`	zmieniono blok `backend` bez migracji istniejącego stanu	`terraform init -migrate-state`, po dokładnym przeczytaniu, co migracja przeniesie
Dwie osoby uruchamiają `apply` naraz i druga dostaje błąd blokady	tak ma działać: backend odrzuca drugą operację, dopóki pierwsza się nie skończy	czekamy na zakończenie pierwszego `apply`, nie próbujemy wymuszać zwolnienia blokady bez pewności, że pierwsza operacja faktycznie padła
`winget install Hashicorp.Terraform` nie znajduje pakietu	winget nie ma jeszcze zaktualizowanych źródeł pakietów	`winget source update`, ponawiamy instalację

Ćwiczenia

> Zadania sprawdzające.

1. Wyjaśniamy, dlaczego plik stanu Terraform nie powinien trafić do tego samego repozytorium Git co pliki `.tf`.

Odpowiedź

Plik stanu zawiera surowe dane o utworzonych zasobach, często z wartościami wrażliwymi w postaci jawnej, i wymaga blokady przy pracy zespołowej, której Git nie zapewnia. Dwie osoby pracujące na osobnych kopiach lokalnego stanu nieuchronnie się rozjadą.

1. Zespół ma już działający serwer PostgreSQL i nie chce stawiać dodatkowej usługi tylko pod stan Terraform. Wskazujemy backend, który to rozwiąże, i piszemy jego blok konfiguracji.

Odpowiedź

Backend `pg`, wskazujący istniejący serwer, na przykład `pg01`:

```
terraform {
  backend "pg" {
    conn_str = "postgres://terraform:HasloDoBazy@pg01:5432/terraform_state"
  }
}
```

1. Opisujemy różnicę między Terraform a OpenTofu, jaką napotkalibyśmy w codziennej pracy z plikami `.tf`.

Odpowiedź

Praktycznie żadnej na starcie: OpenTofu powstał jako fork Terraform sprzed zmiany licencji i czyta te same pliki `.tf` tymi samymi komendami, tylko pod nazwą binarki `tofu`. Różnica dotyczy licencji i tego, kto nadzoruje projekt, a z czasem może dotyczyć też nowych funkcji, które w jednym z narzędzi pojawią się wcześniej niż w drugim.

Powiązane: Automatyzacja: jak przestaliśmy klikać te same rzeczy na dwudziestu serwerach · Terraform: podstawy · Ansible, AWX, Semaphore, Rudder, Terraform: co do czego

Źródła: developer.hashicorp.com/terraform/install · opentofu.org/docs/intro/install/rpm · opentofu.org, historia projektu i licencja MPL 2.0 · developer.hashicorp.com/terraform/language/backend/pg · developer.hashicorp.com/terraform/language/backend/s3

Revision #22
Created 2026-09-28 17:15:26 UTC by Tomasz Konieczny
Updated 2026-10-26 14:13:44 UTC by Tomasz Konieczny