Poza Terraformem

Narzędzia wspomagające pracę z kodem Terraforma

Maciej Rostański

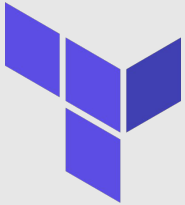
Tematyka prezentacji

01

Automatyczna dokumentacja

02

Statyczna analiza



03

Wycena zmian w infrastrukturze

04

Zmiany w chmurze poza kodem

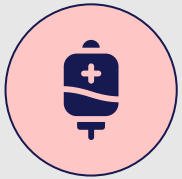
O mnie

Inżynier IT → Wykładowca → DevOps → Cloud Architect

Pracuję z:

- IaC (Terraform)
- Środowiskami chmurowymi i rozproszonymi (AWS, GCP, K8S)
- CI/CD (GitLab, GitHub, Azure DevOps)
- Wspaniałymi ludźmi :D

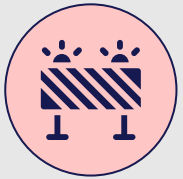
Terraform



Pomoc

Ten sam (lub podobny) kod Terraforma można używać w wielu projektach, jak również współdzielić jak biblioteki

Duża społeczność i bardzo wiele przykładów użycia w artykułach, blogach i serwisach typu stack overflow



Ochrona

Ochrona przed nieostrożnymi działaniami oraz błędami podczas postępowania manualnie

Możliwość przywrócenia infrastruktury do postaci zapisanej w kodzie i pamiętanej jako stan

Problemy pracy z prawdziwym kodem Terraforma

Chaos

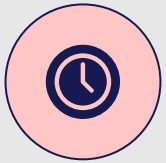
Każdy system z biegiem czasu użytkowania zwiększa swoją entropię ;)

Postęp

Zmiany w dostępnej infrastrukturze (providers) wymuszają stałą i planowaną pracę z kodem

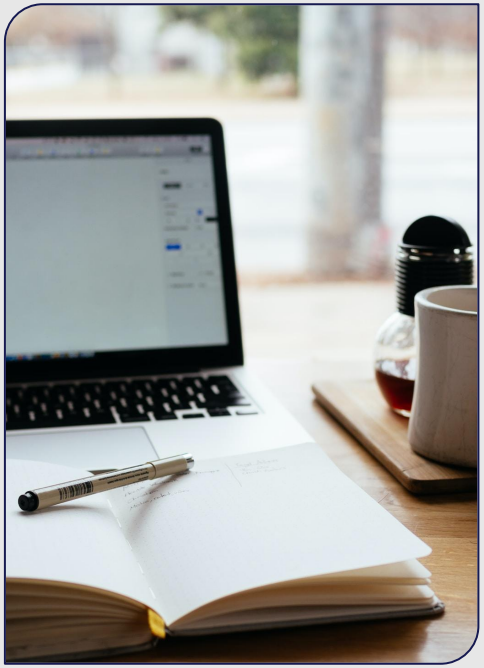
Ludzie

Zmiany wprowadzane przez szeroki zespół noszą znamiona “różnego wytężenia siły”



Czas

Wprowadzanie zmian wymaga więcej czasu niż interwencje manualne



01

Automatyczna dokumentacja

Terraform-docs


https://terraform-docs.io

Inżynier IT → Wykładowca → DevOps → Cloud Architect

Pracuję z:

- IaC (Terraform)
- Środowiskami chmurowymi i rozproszonymi (AWS, GCP, K8S)
- CI/CD (GitLab, GitHub, Azure DevOps)
- Wspaniałymi ludźmi :D

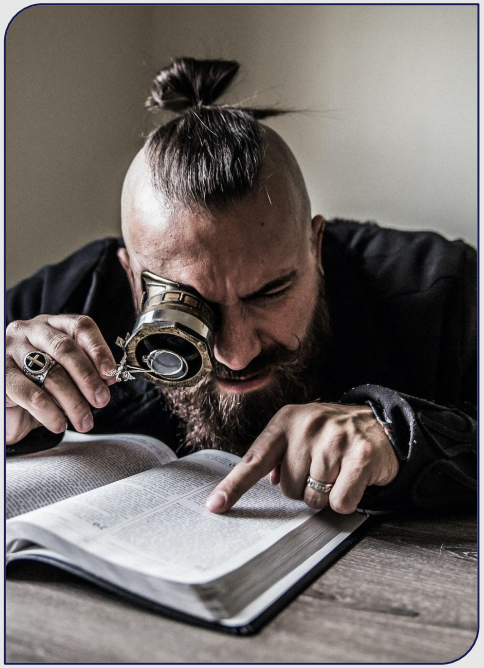
Wykorzystanie terraform-docs

Prawidłowo zaimplementowane działanie terraform-docs:

- Instrukcja użytkowania modułu
- Sprawdzenie poprawności dokumentacji
- Współpraca pomiędzy zespołami (True DevOps)

Wymagania:

- Dobrze przygotowany template
- Jasna i spójna instrukcja używania
- Kultura pracy zespołu :D



02

Statyczna analiza kodu

Narzędzia analizy kodu

Przejrzenie kodu pod kątem dobrych praktyk i zasad bezpieczeństwa

Przykłady:

- otwarcie grupy bezpieczeństwa na cały świat (wszystkie IP)
- brak konfiguracji backupu dla cloud bucketu
- nie skonfigurowane (czyli wyłączone) logi sieci VPC

Co najmniej kilka podobnych narzędzi:

- tfsec
- checkov (Bridgecrew)
- snyk

Wykorzystanie narzędzia checkov

Funkcjonalność:

https://www.checkov.io

- Analiza całego kodu albo planu (zbioru zmian)
- Podpowiedzi prawidłowej konfiguracji
- Wsparcie dla reguł własnych
- Możliwość dynamicznego ignorowania naruszeń

Obserwacje:

- Akceptacja naruszeń reguł wymaga analizy ryzyka!
- Pewne reguły będą właściwe dla jednych zasobów, a dla innych - nie
- Jest output junitXML albo GitLab SAST JSON
- I znów ważna kultura pracy zespołu :D

03

Wycena zmian infrastruktury

Wycena zmian w infrastrukturze

Przełożenie kodu (zasobów) na elementy o określonych kosztach godzinowych (miesięcznych) LUB ilościowych

Możliwości:

- Dostarczenie informacji o koszcie infrastruktury zapisanej w kodzie
- Dostarczenie informacji o koszcie planowanych zmian
- Porównanie kosztów dwóch infrastruktur (w zależności od danych wejściowych)

Problemy:

- Szacunkowe wykorzystanie zasobów płatnych ilościowo
- Koszt zasobów skalowanych dynamicznie
- Koszt bardziej skomplikowanych usług

Działanie infracost


https://www.infracost.io

Składniki rozwiązania:

- Klient CLI + API
- Opcjonalnie: własne proxy API
- Opcjonalnie: konfiguracja wykorzystania zasobów

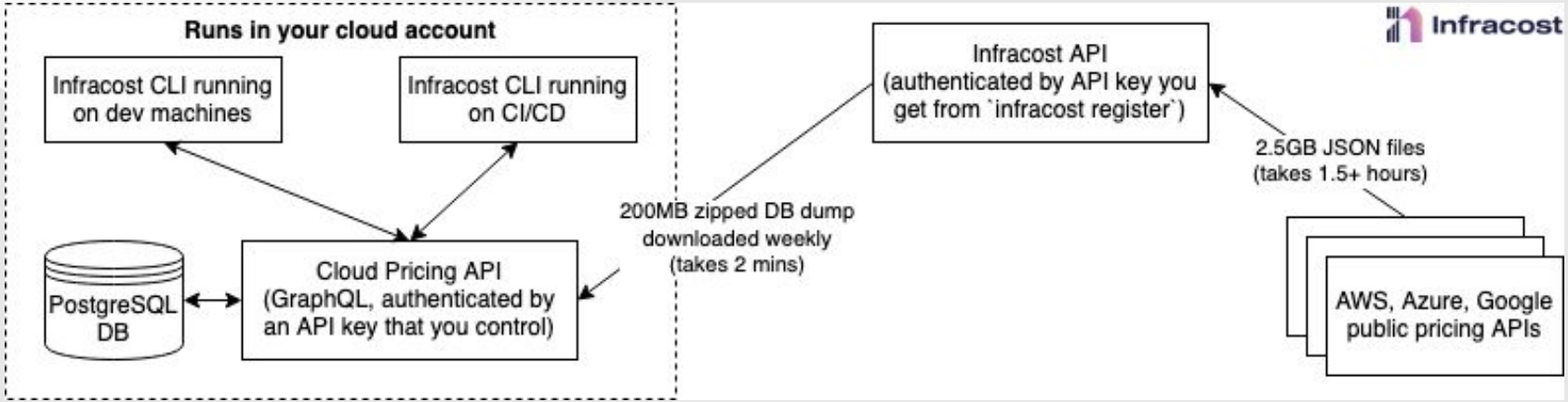
Realia:

- Rozwiązanie rozwijające się - także siłami kontrybutorów
- ... i chyba głównie kontrybutorów
- Bardzo duży potencjał
- ... ale słaba dokumentacja i case studies

Cloud Pricing API

https://www.infracost.io

(infracost proxy)



04

Informacja o zmianach poza kodem

Informacja o zmianach poza kodem

Przejrzenie stanu Terraforma (zasobów) i porównanie ze stanem faktycznym danego obszaru infrastruktury chmurowej

Przypadki użycia:

- Wykrycie zmian dokonanych ręcznie nie objętych kodem
- Detekcja dodatkowych elementów w obszarze objętym opieką
- Sporządzenie listy obiektów jeszcze nie objętych stanem Terraforma

Po co?

- Wspomaganie/raportowanie podczas migracji
- Ewidencja w przypadku współpracy dwóch podmiotów nad jedną infrastrukturą

Driftctl w akcji

Obserwacje ogólne

- Wyjście driftctl może być w postaci:
  - JSON (do dalszego przetworzenia)
  - HTML (artefakt to wizualizacji)
- Analiza driftctl to proces ciągły. Należy mieć procedury na postępowanie w przypadku wykrycia niepożądanych zasobów!

Uwaga na:

- Zużycie API providera (dokładamy kolejne wywołania API przed planem i apply)
- Nie może być kilku konfiguracji providera w kodzie (np. wiele regionów)

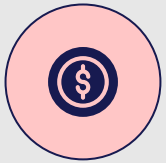
05

Podsumowanie

TESTIMONIALS

tfdocs ★★★★★

checkov ★★★★☆



infracost ★★☆☆☆

driftctl ★★★☆☆

"Terraform-docs jest nieocenionym, bardzo konfigurowalnym narzędziem, upraszczającym mnóstwo pracy nad dokumentacją"

Maciej R, DevOps Engineer

"Checkov świetnie uzupełnia pracę lidera kodu, automatycznie wskazując ewentualne niedoróbki w kodzie"

Maciek, Team Leader

"Duży potencjał, ale sprawia wrażenie projektu studenckiego, a nie narzędzia, które można używać na produkcji"

Maciej (Cloud Architect)

"Rzadko potrzebne narzędzie, choć w tych konkretnych sytuacjach, do których jest przeznaczony, będzie niezastąpiony"

MR, Azure DevOps Contractor

Dzięki!

https://github.com/terraform-training/terraform-tools

Jakieś przemyślenia?

mrostanski@gmail.com
https://www.linkedin.com/in/maciej-rostanski/