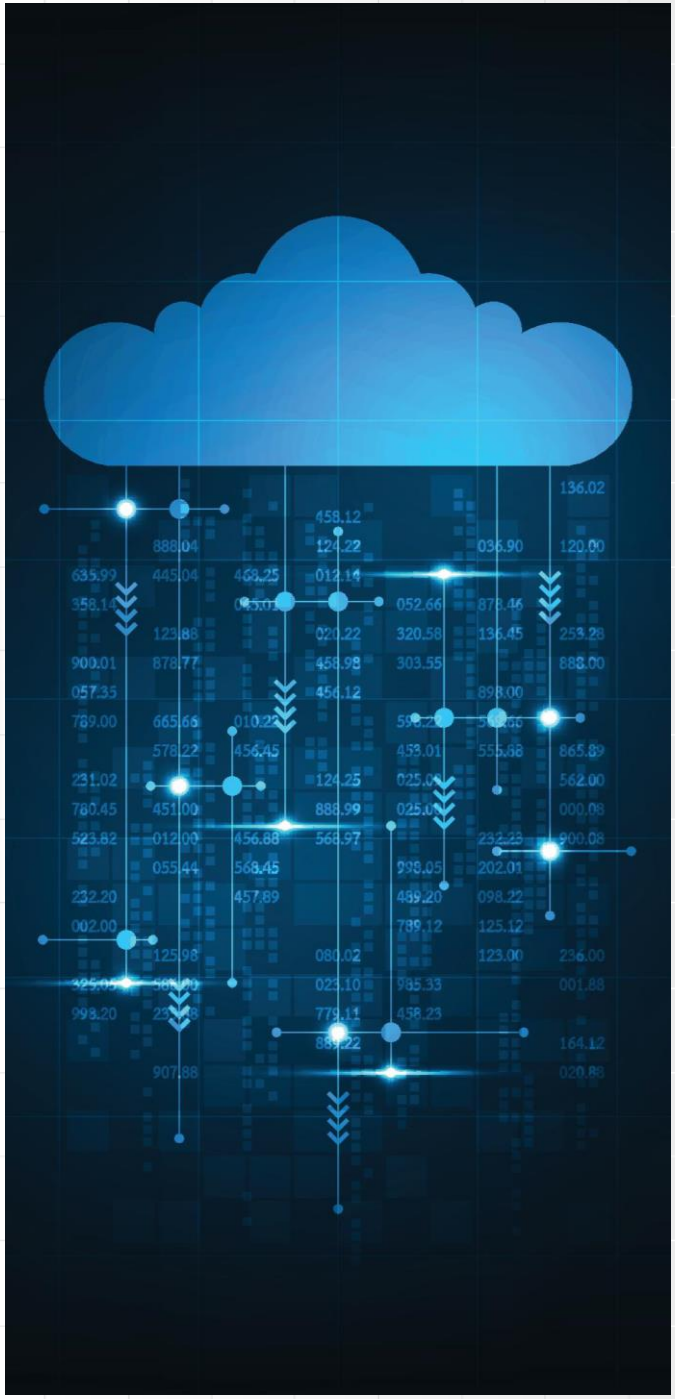


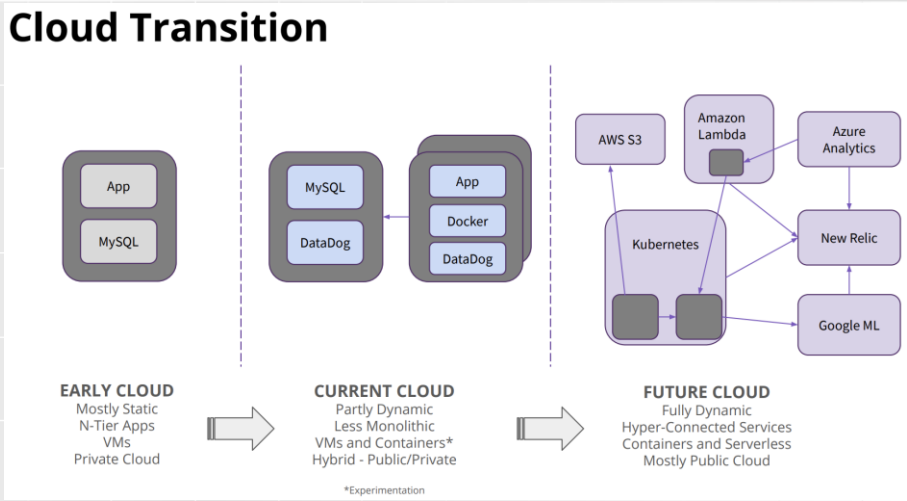
Programowanie w chmurze

Rafał Palak

Politechnika Wrocławska

Infrastruktura jako kod (IaC) [1]

- Zarządzanie i udostępnianie infrastruktury za pomocą kodu, a nie procesów ręcznych
- Tworzone są pliki konfiguracyjne zawierające specyfikacje infrastruktury, co ułatwia edycję i dystrybucję konfiguracji
- Za każdym razem tworzone jest to samo środowisko
- Kodyfikując i dokumentując specyfikacje konfiguracji, IaC wspomaga zarządzanie konfiguracją i pomaga uniknąć nieudokumentowanych, doraźnych zmian konfiguracyjnych

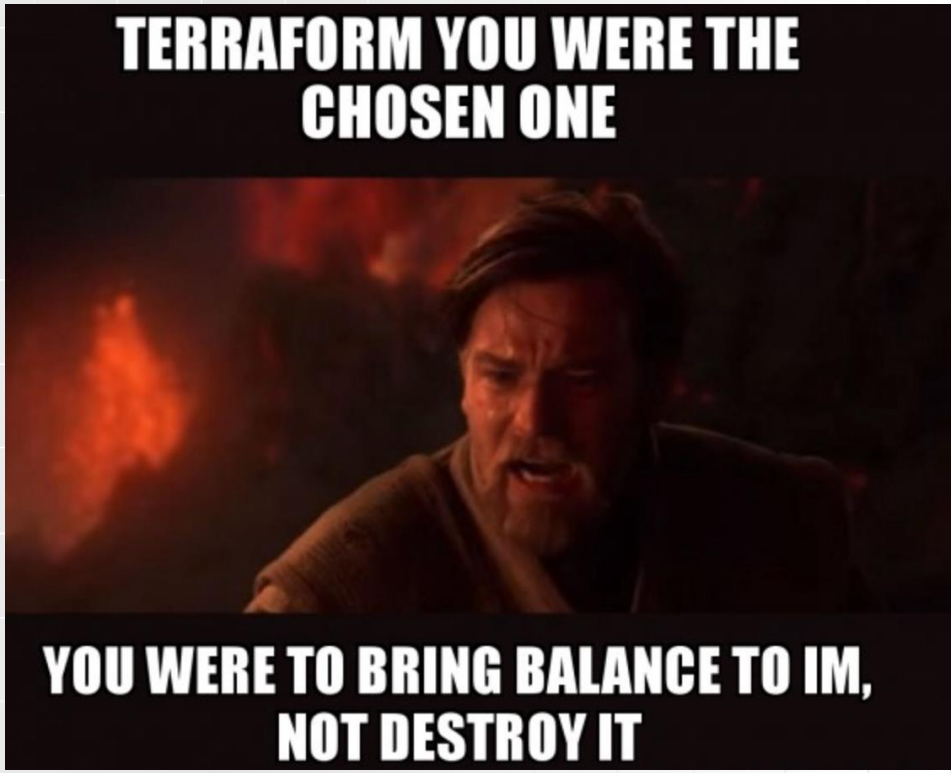


Infrastruktura jako kod (IaC)[2]

- Kontrola wersji jest ważną częścią IaC,
- Pliki konfiguracyjne powinny być traktowane tak jak każdy inny plik kodu źródłowego oprogramowania.
- Wdrożenie infrastruktury jako kodu oznacza również, że możesz podzielić swoją infrastrukturę na modułowe komponenty, które następnie można łączyć na różne sposoby poprzez automatyzację
- Bez IaC coraz trudniej jest zarządzać skalą dzisiejszej infrastruktury
- Poprawia spójność oraz redukuje błędy i ręczną konfigurację

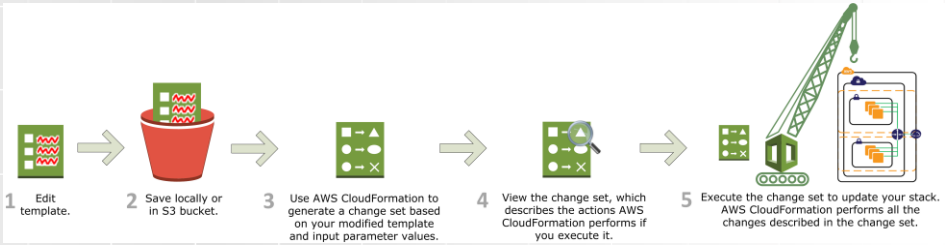
Teraform

- Jedno z popularniejszych **narzędzi do zarządzania infrastrukturą jako kod** (Infrastructure as Code, IaC)
- Pozwala **opisać strukturę swojej infrastruktury** za pomocą języka konfiguracyjnego **HashiCorp Configuration Language** (HCL) lub **JSON**
- **Projekt** o niesamowitej elastyczności, **obsługujący wszystkie najpopularniejsze platformy chmurowe**



CloudFormation

- Działa tylko dla AWS
- Jest w **pełni zintegrowany z AWS** pozwalając na większą kontrolę
- Pozwala opisać strukturę swojej infrastruktury za pomocą **YAML** and **JSON**
- **Oferuje rollback** pozwalający przywrócenie aplikacji do poprzedniego stanu
- **Ścisły związek tego narzędzia z AWS** umożliwia wdrażanie infrastruktury w kilku regionach i kontach przy użyciu tego samego szablonu CloudFormation.
- Odpowiedniki u konkurencji: **Google Cloud Deployment Manager** oraz **Azure Resource Manager**



Wrocław
University
of Science
and Technology

Teraform - podstawy

Inicjalizacja Teraforma - terraform init[1]

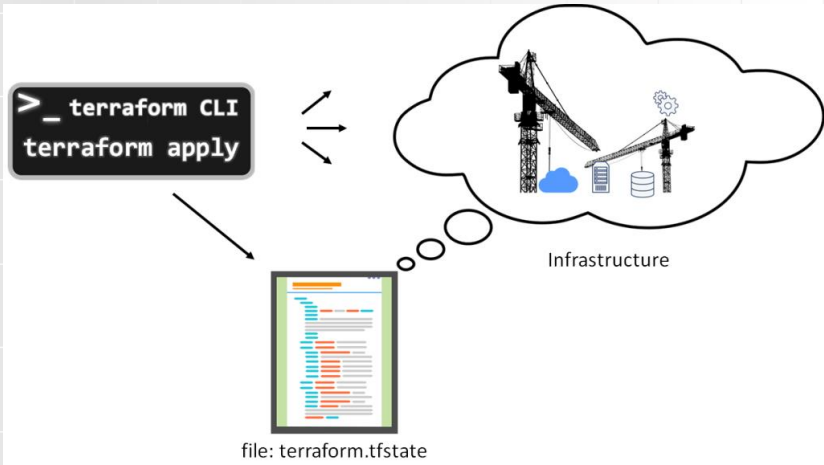
- **Pobieranie Dostawców (Providers)**
  - Teraform **używa różnych dostawców do interakcji z różnymi usługami** chmurowymi lub technologiami.
  - Teraform **automatycznie pobiera niezbędne wtyczki dostawców**, które są wymienione w konfiguracji.
- **Utworzenie Katalogu .terraform**
  - Teraform **tworzy katalog .terraform**, w którym **przechowuje pobrane wtyczki dostawców oraz inne tymczasowe pliki**, które są używane do **zarządzania stanem twojej infrastruktury**.

Inicjalizacja Teraforma - terraform init [2]

- **Ustawienie Backendu**
  - Jeśli w twojej konfiguracji jest zdefiniowany **backend** (służący do przechowywania stanu infrastruktury), terraform init skonfiguruje go odpowiednio.
  - **Backendy mogą być używane do przechowywania informacji o stanie infrastruktury w sposób zdalny**, co ułatwia współpracę i zarządzanie infrastrukturą w zespołach.
- **Inicjalizacja Stanu**
  - Teraform **inicjalizuje stan infrastruktury**, **tworząc nowy plik stanu** lub **konfigurując dostęp do istniejącego stanu** w zależności od konfiguracji.

Stan [1]

- Odnosi się do **aktualnej konfiguracji i właściwości zasobów infrastruktury**, które są zarządzane przez Teraform
- **Plik stanu** jest **używany do mapowania zasobów w konfiguracji** Teraforma do **rzeczywistych zasobów w usłudze chmurowej**
- Pozwala Teraformowi na **śledzenie informacji o zarządzanych zasobach**, co jest kluczowe dla odpowiedniego **planowania i wprowadzania zmian** w infrastrukturze
- Pozwala Teraform **wiedzieć, które zasoby istnieją i jakie są ich aktualne właściwości**

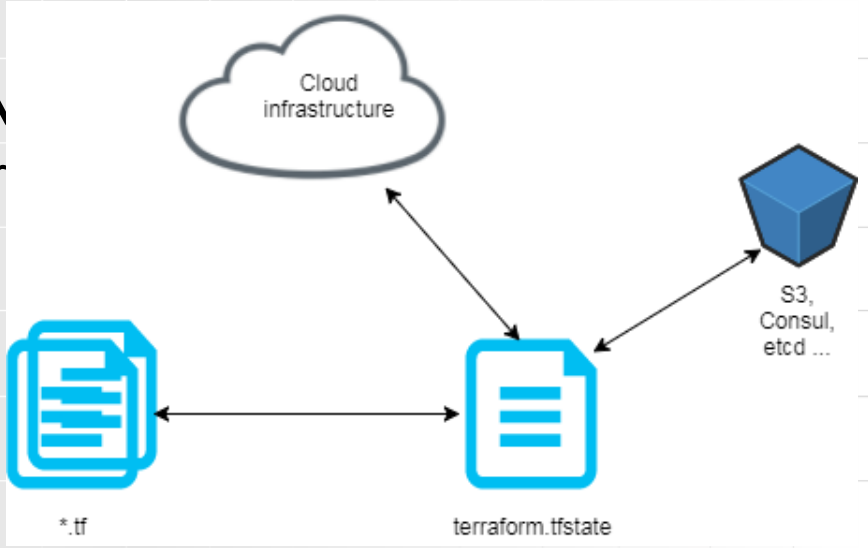


Stan [2]

- Kluczowy dla **uniknięcia konfliktów i niespójności**, gdy **różne osoby lub systemy wprowadzają zmiany w infrastrukturze**
- Pozwala Teraformowi **optymalizować operacje**, tylko **wprowadzając zmiany tam, gdzie są one naprawdę potrzebne,** zamiast rekreować wszystko od zera za każdym razem
- Używany do **wyświetlania wartości wyjściowych** (outputs) po zastosowaniu zmian, co **pozwala na łatwe przekazywanie informacji między różnymi modułami i konfiguracjami** Teraforma.

Stan [3]

- **Teraform umożliwia zarządzanie stanem lokalnie lub zdalnie**
- Przechowywanie **stanu w sposób zdalny** jest zalecane dla **większych zespołów i infrastruktur**, ponieważ zapewnia lepszą kontrolę i bezpieczeństwo
- **Plik stanu** jest zazwyczaj **plikiem JSON** który można przeglądać i zarządzać pr użyciu różnych poleceń Teraform, takich jak **terraform state list** czy **terraform state show**.



Planowanie Zmian – terraform plan

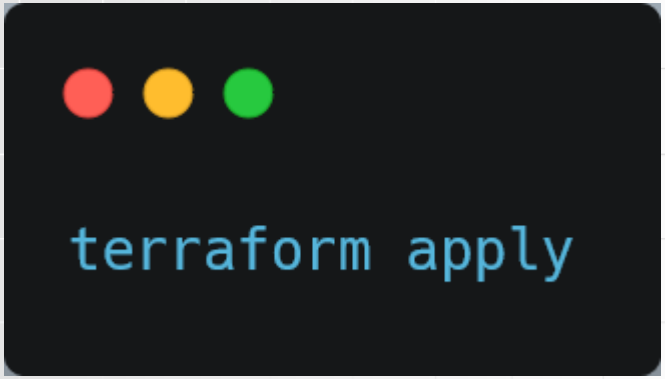
- Jedno z **najważniejszych narzędzi** w ekosystemie Teraforma
- Umożliwia **przeprowadzenie analizy różnic między aktualnym stanem infrastruktury** (zapisany w pliku stanu) a **stanem opisanym w plikach konfiguracyjnych**.
- Na podstawie **analizy**, Teraform **generuje plan działania**, który pokazuje, które **zasoby zostaną utworzone**, **zmodyfikowane** lub **usunięte**.
- Plan jest **wyświetlany w konsoli**, dając użytkownikowi **pełny przegląd proponowanych zmian przed ich faktycznym wprowadzeniem**.

Planowanie Zmian – dlaczego?

- **Bezpieczeństwo** - Umożliwia **bezpieczne sprawdzenie, jakie zmiany zostaną wprowadzone**, zanim faktycznie zostaną zastosowane. Dzięki temu **pozwala uniknąć niechcianych efektów ubocznych**.
- **Dokładność** - Pomaga w **identyfikowaniu błędów w konfiguracji** lub **niezgodności z rzeczywistą infrastrukturą** przed próbą wprowadzenia zmian.
- **Kontrola** - Daje **możliwość recenzji i zatwierdzenia zmian** przez innych członków zespołu lub przez procesy automatycznego wdrażania (CI/CD).
- **Dokumentacja** - Generowany plan **może służyć jako dokumentacja zmian**, która może być archiwizowana dla celów audytu lub analizy.

Wprowadzenie zmian – terraform apply [1]

- Używane do **wprowadzenia zmian w infrastrukturze** zgodnie z **aktualną konfiguracją opisaną w plikach** Teraform
- **Najpierw generuje plan zmian**, który pokazuje, jakie **zasoby zostaną utworzone**, **zmodyfikowane** lub **usunięte**.
- Po wygenerowaniu planu, **polecenie pyta użytkownika o zatwierdzenie wprowadzenia zmian**. Można **zobaczyć plan i zdecydować, czy chcesz go zastosować**.



Wprowadzenie zmian – terraform apply [2]

- Po **zatwierdzeniu, terraform apply** faktycznie **wykonuje zmiany w infrastrukturze**: **tworzy**, **modyfikuje** lub **usuwa zasoby zgodnie z planem**.
- Po **zakończeniu operacji**, **stan infrastruktury** jest **aktualizowany**, aby odzwierciedlić wprowadzone zmiany.

Niszczenie zasobów – terraform destroy

- **Usuwa wszystkie zasoby opisane w plikach konfiguracyjnych Teraforma**, które zostały wcześniej utworzone przez terraform apply.
- Podobnie jak w przypadku terraform apply, polecenie destroy również **pyta o potwierdzenie przed wykonaniem**.
- Pozwala zobaczyć, jakie **zasoby zostaną usunięte i zdecydować**, czy kontynuować
- **Aktualizacja Stanu Po zakończeniu operacji, plik stanu jest aktualizowany, aby odzwierciedlić brak zasobów.**

Niszczenie zasobów – dlaczego?

- **Zwalnianie Zasobów -** Jeżeli **nie potrzeba już pewnej infrastruktury**, możesz ją **łatwo usunąć**, co może być szczególnie ważne w kontekście **zarządzania kosztami**.
- **Czyszczenie Środowiska -** Jest to **przydatne w środowiskach testowych i deweloperskich**, gdzie często tworzy się i niszczy infrastrukturę.
- **Bezpieczeństwo - Dzięki potwierdzeniu przed usunięciem** istnieje dodatkowa **kontrola** nad tym, **co zostanie zniszczone**.
- **Automatyzacja –** Można **zautomatyzować proces niszczenia zasobów** w ramach **workflow CI/CD**, choć **wymaga to ostrożności**, aby **nie usunąć ważnych zasobów**.

Niszczenie zasobów – uwaga!

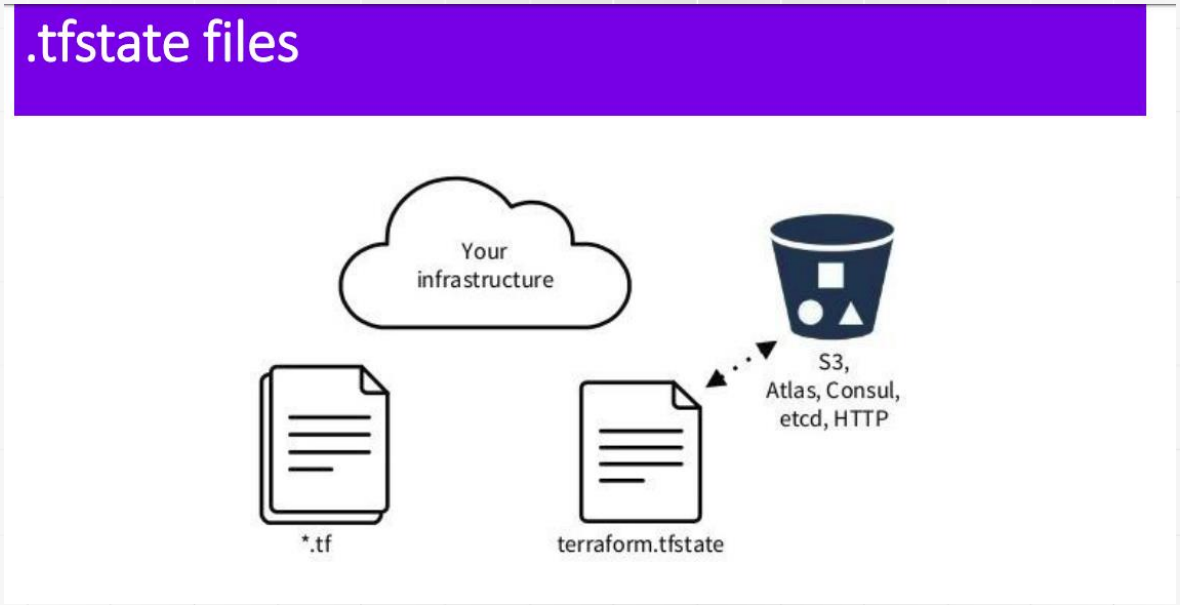
- **W środowiskach produkcyjnych -** ***Zawsze*** upewnij się, że wiesz, **jakie zasoby zostaną usunięte**, i zastanów się nad **konsekwencjami** ich usunięcia.
- Uzywaj **flagi *-target*** do określenia **konkretnych zasobów**, które chcesz zniszczyć, **zamiast niszczyć wszystko**. Na przykład: terraform destroy -target=aws_instance.my_instance.

Stan i jak jest zarządzany

- Zarządzanie stanem jest **jednym z kluczowych aspektów pracy z Teraformem**.
- Stan w Teraformie to **zapis, który przechowuje informacje o zasobach zarządzanych przez Teraform** w danym projekcie.
- **Plik** stanu jest **wykorzystywany przez Teraform do mapowania zasobów** w konfiguracji na **rzeczywiste zasoby w środowisku docelowym**.

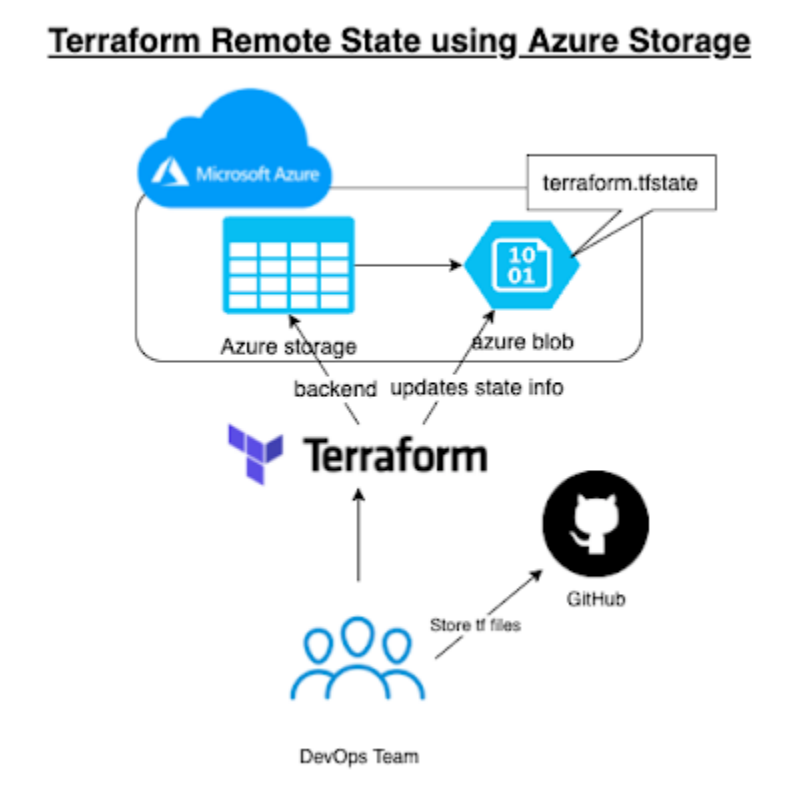
Stan - przechowywanie

- **Lokalnie - Domyślnie**, Teraform przechowuje stan w **pliku** o nazwie **terraform.tfstate** w katalogu roboczym. Jest to **dobre dla prostych, indywidualnych projektów**.
- **Zdalnie -** Dla **zespołów i bardziej złożonych projektów**, przechowywanie stanu zdalnie **jest zalecane**. Pozwala to na **lepsze zarządzanie dostępem**, **bezpieczeństwo** i **historię zmian**.
- Teraform wspiera **wiele opcji backendu** do przechowywania stanu, w tym usługi takie jak AWS S3, Google Cloud Storage, Azure Blob Storage, itd.



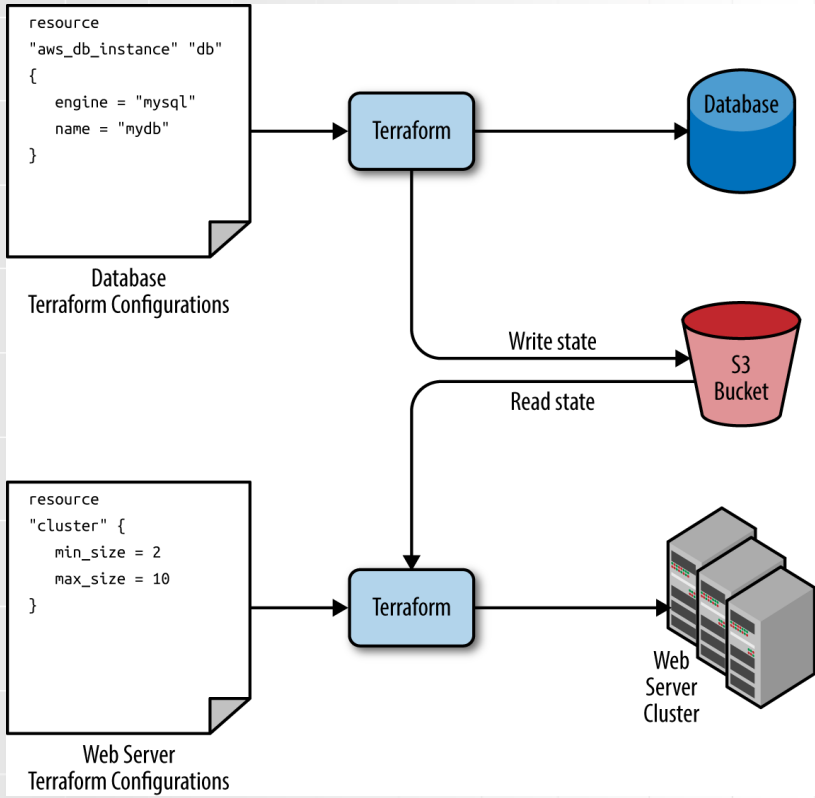
Stan - struktura pliku stanu

- Plik stanu w formacie **JSON przechowuje szeroką gamę informacji o zarządzanych zasobach**
  - **Identyfikatory Zasobów -** Unikalne identyfikatory zasobów w dostawcy usług.
  - **Atrybuty Zasobów -** Aktualne atrybuty każdego zasobu, np. adres IP instancji VM.
  - **Zależności między Zasobami -** Informacje o tym, jak zasoby są od siebie zależne.



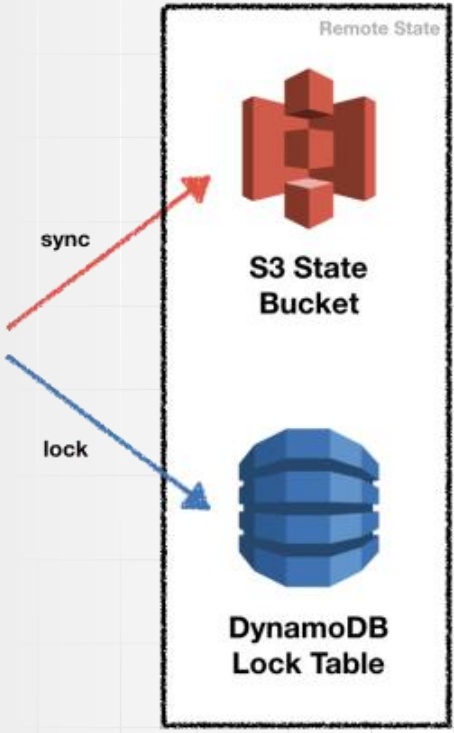
Stan - zarządzanie stanem

- **Zabezpieczenia**: Stan zawiera **wrażliwe dane**, takie jak **hasła** czy **klucze API** w postaci zaszyfrowanej. Należy Zawsze **zabezpieczać dostęp do plików stanu** i **używać zdalnych backendów** z odpowiednimi **środkami kontroli dostępu i szyfrowania**.
- **Blokady Stanu -** Aby zapobiec konfliktom, Teraform **umożliwia "blokowanie" stanu** podczas aplikowania zmian, **zapobiegając jednoczesnym modyfikacjom przez różne procesy**.
- **Zarządzanie Wersjami -** Niektóre backendy (np. S3 z włączonym versioningiem) **umożliwiają przechowywanie wielu wersji pliku stanu**, co **pozwala na łatwe przywracanie do poprzedniego stanu**.
- **Stan jako Środowisko -** Możliwość **zarządzania wieloma środowiskami** (np. produkcja, staging) przez **utworzenie oddzielnych konfiguracji** stanu dla każdego z nich.



Stan – dobre praktyki

- **Unikaj Edycji Ręcznej -** Bezpośrednia edycja plików stanu jest **ryzykowna i może prowadzić do niespójności**. Zaleca się **używanie poleceń Teraforma** do zarządzania stanem.
- **Regularne Przeglądy i Backupy –**Regularne sprawdzanie i backupu pliki stanu, aby uniknąć utraty danych.
- **Automatyzacja -** W miarę możliwości **automatyzacja procesów związane ze stanem**, takie jak **backup** czy **przenoszenie między środowiskami**.
- **<u>Właściwe zarządzanie stanem jest kluczowe dla efektywnego i bezpiecznego korzystania z Teraforma w zarządzaniu infrastrukturą.</u>**



Wrocław
University
of Science
and Technology

Dziękuję za uwagę