Zarządzanie Systemami Rozproszonymi
Laboratoria z Terraform

mgr inż. Jakub Woźniak

1 Wprowadzenie do zaawansowanych funkcji Terraform

Terraform to narzędzie służące do zarządzania infrastrukturą w chmurze z wykorzystaniem podejścia Infrastructure as Code (IaC). Zaawansowane funkcje Terraform, takie jak wykorzystanie modułów, zmiennych oraz zdalnego stanu, pozwalają na budowę skalowalnych i czytelnych środowisk w chmurze.

1.1 Moduły

Moduł w Terraform to zestaw plików, w których można zdefiniować powiązane ze sobą zasoby i ich konfiguracje. Dzięki modułom:

- Ułatwia się powtórne wykorzystanie kodu.
- Redukuje się złożoność dzięki podziałowi infrastruktury na mniejsze części.
- Łatwiej utrzymać spójność oraz standardy w różnych częściach projektu.

1.2 Zmiennie

Zmienna (`variable`) jest wykorzystywana w Terraform do parametryzacji modułów i skryptów. Można określać typy zmiennych oraz domyślne wartości. Przekazywanie wartości do zmiennych odbywa się najczęściej przez pliki `.tfvars` lub parametry CLI.

1.3 Zdalny stan (Remote State)

Stan Terraform to plik (`.tfstate`), w którym przechowywana jest informacja o wdrożonych zasobach. Domyślnie stan zapisywany jest w pliku lokalnym, ale w przypadku zespołowej pracy nad infrastrukturą niezbędne jest przeniesienie stanu do lokalizacji zdalnej (np. Azure Blob Storage), aby uniknąć konfliktów i umożliwić współdzieloną edycję.

1.4 Dlaczego warto stosować modularność w Terraform

- **Reużywalność kodu**: Te same wzorce infrastruktury można wykorzystać w wielu projektach.
- **Łatwość utrzymania**: Modyfikacje w module są automatycznie dostępne we wszystkich miejscach, w których moduł jest używany.
- **Skalowalność**: Infrastruktura może rozrastać się w uporządkowany sposób.

W kolejnych ćwiczeniach zostaną przedstawione przykłady praktycznego wykorzystania modułów do tworzenia infrastruktury chmurowej na platformie Azure. Każde ćwiczenie rozpoczyna się wstępem teoretycznym, który omawia kluczowe pojęcia.

2 Ćwiczenie 1: Moduł Resource Group

2.1 Wprowadzenie teoretyczne

Resource Group w Azure to podstawowa jednostka logiczna, w której przechowywane są zasoby. Poprzez stworzenie prostego modułu do zarządzania grupą zasobów (Resource Group) możemy zademonstrować strukturę modułu w Terraform:

- `main.tf` – zawiera główną definicję modułu, w tym zasoby.
- `variables.tf` – przechowuje definicje zmiennych.
- `outputs.tf` – definiuje wartości wyjściowe (outputs) udostępniane przez moduł.

2.2 Kroki do wykonania zadania

1. Utworzyć nowy folder o nazwie `rg_module`.
2. W tym folderze przygotować plik `main.tf`, który będzie definiował zasób Resource Group.
3. Dodać plik `variables.tf` z definicjami zmiennych takich jak `rg_name` i `rg_location`.
4. Dodać plik `outputs.tf` z wyjściem udostępniającym np. nazwę Resource Group.
5. Wywołać moduł `rg_module` z poziomu głównego pliku Terraform (`main.tf` w katalogu głównym projektu), przekazując odpowiednie wartości do zmiennych modułu.

2.3 Oczekiwany rezultat

Po zakończeniu ćwiczenia powinna zostać utworzona nowa Resource Group w wybranym regionie Azure. Projekt Terraform będzie też zawierał pierwszy działający moduł.

2.4 Przykładowy kod

Listing 1: Tworzenie Resource Group w Terraform

```
resource "azurerm_resource_group" "example" {
  name     = var.rg_name
  location = var.rg_location
}
```

3 Ćwiczenie 2: Moduł Virtual Network

3.1 Wprowadzenie teoretyczne

Virtual Network (VNet) w Azure umożliwia tworzenie prywatnej przestrzeni adresowej dla zasobów w chmurze. W module umieścimy definicję VNet oraz Subnetów, pozwalając na zdefiniowanie:

- Przestrzeni adresowej VNet (np. `10.0.0.0/16`).
- Liczby Subnetów oraz ich wielkości.

3.2 Kroki do wykonania zadania

1. Utworzyć folder `vnet_module`.
2. W pliku `main.tf` zdefiniować zasób `azurerm_virtual_network` oraz `azurerm_subnet`.
3. W pliku `variables.tf` zdefiniować zmienne dla nazwy VNet, przestrzeni adresowej oraz parametrów Subnetów.
4. W `outputs.tf` udostępnić ID VNet oraz Subnetów.
5. Wywołać moduł `vnet_module` z głównego `main.tf`, przekazując odpowiednie wartości zmiennych.

3.3 Oczekiwany rezultat

Po zakończeniu ćwiczenia powinna zostać utworzona sieć wirtualna z Subnetami według zadanych parametrów. Nowy moduł umożliwi łatwą konfigurację warstwy sieciowej dla kolejnych zasobów.

4 Ćwiczenie 3: Moduł Virtual Machine

4.1 Wprowadzenie teoretyczne

Virtual Machine (VM) w Azure można skonfigurować za pomocą Terraform, definiując m.in.:

- Rozmiar maszyny (np. `Standard_B1s`).
- System operacyjny (np. Linux Ubuntu).
- Sieć, do której VM jest podłączona.

W tym ćwiczeniu utworzymy moduł zawierający definicję maszyny wirtualnej z dynamiczną konfiguracją (np. listą maszyn, które mają być zbudowane w pętli `for_each` lub `count`).

4.2 Kroki do wykonania zadania

1. Utworzyć folder `vm_module`.
2. W pliku `main.tf` zdefiniować zasób `azurerm_virtual_machine` z wykorzystaniem `for_each` lub `count`.
3. Zdefiniować odpowiednie zmienne w `variables.tf` (np. lista maszyn, parametry systemu operacyjnego).
4. Zdefiniować wartości wyjściowe w `outputs.tf` (np. adresy IP utworzonych VM).
5. Wywołać moduł `vm_module` z głównego `main.tf`, tworząc jedną lub wiele maszyn wirtualnych.

4.3 Oczekiwany rezultat

Po zakończeniu ćwiczenia w wybranej sieci wirtualnej zostaną utworzone maszyny wirtualne zdefiniowane w module, a wszystkie parametry (np. rodzaj systemu czy liczba maszyn) będą kontrolowane za pomocą zmiennych.

5 Ćwiczenie 4: Połączenie modułów – Kompleksowa infrastruktura

5.1 Wprowadzenie teoretyczne

W tym ćwiczeniu zintegrujemy wszystkie dotychczasowe moduły w jednym projekcie. Celem jest stworzenie kompletnej infrastruktury obejmującej:

- Resource Group.

- Virtual Network z Subnetami.
- Maszyny wirtualne.
- Load Balancer (opcjonalnie), aby rozkładać ruch do maszyn.

5.2 Kroki do wykonania zadania

1. W głównym pliku `main.tf` (poza modułami) zdefiniować nową Resource Group lub wykorzystać istniejącą z poprzednich ćwiczeń.
2. Wywołać moduł VNet z podaniem przestrzeni adresowej i Subnetów.
3. Wywołać moduł VM, aby utworzyć klaster maszyn wirtualnych w Subnetach.
4. Opcjonalnie dodać moduł Load Balancer, przekazując mu informacje o maszynach.

5.3 Oczekiwany rezultat

W efekcie powstaje zestaw zasobów, które mogą wspólnie stanowić środowisko testowe lub podstawę produkcyjnej infrastruktury.

6 Ćwiczenie 5: Zarządzana baza danych PostgreSQL

6.1 Wprowadzenie teoretyczne

Zarządzane bazy danych w Azure (np. `azurerm_postgresql_server` i `azurerm_postgresql_database`) pozwalają na łatwą konfigurację, automatyczne aktualizacje i wysoki poziom dostępności. Terraform umożliwia definiowanie parametrów instancji bazy i kontrolowanie dostępu sieciowego.

6.2 Kroki do wykonania zadania

1. Utworzyć folder `postgresql_module`.
2. W pliku `main.tf` zdefiniować:
   - `azurerm_postgresql_server` – definiujący serwer bazy danych.
   - `azurerm_postgresql_database` – tworzący konkretną bazę danych.
   - Ewentualne reguły firewall lub sieciowe dla bazy.
3. Zdefiniować zmienne w `variables.tf` (hasło, SKU bazy, nazwa bazy, itp.).
4. Zdefiniować wartości wyjściowe w `outputs.tf` (np. connection string).

5. Powiązać moduł bazy danych z maszynami wirtualnymi, aby aplikacja mogła łączyć się z bazą.

6.3 Oczekiwany rezultat

Po zakończeniu ćwiczenia powinna zostać utworzona zarządzana baza danych PostgreSQL, a maszyny wirtualne będą miały możliwość połączenia z nią.

7 Ćwiczenie 6: Zdalny stan Terraform

7.1 Wprowadzenie teoretyczne

Konfiguracja **zdalnego stanu** umożliwia bezpieczne przechowywanie pliku stanu w chmurze. Dzięki temu wiele osób może pracować jednocześnie z tym samym stanem, a Terraform zapewnia mechanizmy blokowania zapisu i rozwiązywania konfliktów.

7.2 Kroki do wykonania zadania

1. W pliku `backend.tf` (lub w `terraform{...}` sekcji głównego `main.tf`) skonfigurować `backend` typu `azurerm`.
2. Utworzyć kontener w Azure Blob Storage, w którym będzie przechowywany stan (`tfstate`).
3. Skonfigurować `storage_account_name`, `container_name` oraz `key` (nazwa pliku stanu).
4. Przetestować współdzielenie stanu, uruchamiając polecenia `terraform plan` i `terraform apply` z różnych stanowisk.

7.3 Oczekiwany rezultat

Po zakończeniu ćwiczenia plik stanu będzie przechowywany w usłudze Azure Blob Storage, co umożliwi zespołowe zarządzanie zasobami i wyeliminuje konflikty związane z lokalnymi kopiami stanu.

8 Ćwiczenie końcowe: Własny projekt zespołowy

8.1 Wprowadzenie teoretyczne

W ćwiczeniu końcowym uczestnicy łączą wszystkie elementy poznane wcześniej, tworząc kompleksowy projekt infrastruktury w oparciu o moduły i zdalny stan.

8.2 Kroki do wykonania zadania

1. Zaplanować architekturę aplikacji w zespole (Resource Group, VNet, VM, Baza PostgreSQL, Load Balancer lub inne).
2. Utworzyć folder projektu z głównym plikiem `main.tf` i plikami `tfvars`, które zawierają wartości dla zmiennych.
3. Wykorzystać utworzone wcześniej moduły (Resource Group, VNet, VM, PostgreSQL).
4. Skonfigurować `backend` Terraform do zdalnego stanu, jeśli to możliwe.
5. Zbudować i przetestować całą infrastrukturę, weryfikując poprawne działanie aplikacji.

8.3 Oczekiwany rezultat

Rezultatem będzie w pełni funkcjonalna infrastruktura zbudowana we współpracy zespołowej, używająca wielokrotnie zaprojektowanych modułów i udostępniająca kompletne środowisko do uruchomienia aplikacji.

9 Podsumowanie i wnioski

W trakcie ćwiczeń zaprezentowano, w jaki sposób Terraform umożliwia organizowanie rozbudowanej infrastruktury w sposób modularny, skalowalny i przejrzysty. Kluczowymi elementami były moduły, zmienne oraz zdalny stan, które wspólnie pomagają w utrzymaniu dużych projektów i zespołowej pracy nad kodem.
Rozszerzeniem przedstawionego podejścia może być dodanie kolejnych usług, takich jak:

- **Load Balancer** (lub Application Gateway) z bardziej zaawansowaną konfiguracją reguł ruchu.
- **AKS** (Azure Kubernetes Service), aby konteneryzować aplikacje i zarządzać ich cyklem życia.

Niniejszy skrypt stanowi wprowadzenie do zarządzania systemami rozproszonymi z wykorzystaniem Terraform w chmurze Azure i może służyć za punkt wyjścia do dalszych eksperymentów i rozbudowy infrastruktury.