Zarządzanie Systemami Rozproszonymi
Laboratoria z Microsoft Azure i Terraform

mgr inż. Jakub Woźniak

1 Wprowadzenie do chmury publicznej

1.1 Historia Rozwoju: Od Kolokacji do Chmury

Na początku ery internetu firmy inwestowały we własną infrastrukturę sprzętową, którą umieszczano w centrach danych (kolokacja). Organizacje wynajmowały fizyczne miejsce na serwery, jednak zarządzanie sprzętem i oprogramowaniem pozostawało po ich stronie. Takie rozwiązanie było kosztowne, nieelastyczne i wymagało specjalistycznych zespołów.
Pojawienie się wirtualizacji zrewolucjonizowało rynek i dało początek **chmurze obliczeniowej**. Pierwszym krokiem był model **Infrastructure as a Service (IaaS)**. W 2006 roku Amazon uruchomił usługę EC2, umożliwiając firmom wynajem mocy obliczeniowej na żądanie. Wkrótce pojawiły się inne modele:

- **Platform as a Service (PaaS)** – np. Azure App Service, eliminujący konieczność zarządzania systemem operacyjnym.
- **Software as a Service (SaaS)** – np. Microsoft 365, w pełni gotowe oprogramowanie dostępne dla użytkowników.

Dziś chmura publiczna, jak **Microsoft Azure** czy **AWS**, zapewnia elastyczność, skalowalność i dostęp do zaawansowanych usług przy minimalnych inwestycjach w infrastrukturę.

1.2 Model Współdzielonej Odpowiedzialności (Shared Responsibility)

W chmurze publicznej odpowiedzialność za bezpieczeństwo i zarządzanie zasobami jest podzielona między dostawcę usługi a klienta. Jest to tzw. **Shared Responsibility Model**.

Podział odpowiedzialności w zależności od modelu usług

- **IaaS (Infrastructure as a Service):** Dostawca (np. Microsoft Azure) odpowiada za infrastrukturę fizyczną, ale użytkownik zarządza systemem operacyjnym, aktualizacjami i aplikacjami.

- **PaaS (Platform as a Service):** Dostawca zarządza infrastrukturą i systemem operacyjnym, a użytkownik odpowiada za aplikacje oraz ich konfigurację.
- **SaaS (Software as a Service):** Całość (od sprzętu po aplikacje) jest zarządzana przez dostawcę, a użytkownik odpowiada jedynie za dane i konfigurację kont.

Przykład: W usłudze *Azure App Service*, użytkownik implementuje aplikację, a dostawca zajmuje się infrastrukturą i systemem operacyjnym.

1.3 Demokratyzacja Technologii

Chmura publiczna odegrała kluczową rolę w **demokratyzacji technologii**. Usługi takie jak **Azure Cognitive Services** czy **AWS Rekognition** umożliwiają dostęp do zaawansowanych funkcji sztucznej inteligencji (AI) dla firm każdej wielkości.

Dlaczego to ważne?

Alternatywą dla chmury były wcześniej drogie, ręcznie implementowane rozwiązania, które wymagały zespołów specjalistów, infrastruktury i dużych inwestycji czasowych. Dzięki chmurze:

- Start-up może wykorzystać **Azure Computer Vision** do analizy obrazu bez trenowania własnych modeli AI.
- Firmy płacą tylko za rzeczywiste zużycie zasobów, co radykalnie obniża koszty i próg wejścia.

Demokratyzacja technologii przyspiesza innowacje i otwiera rynek dla małych organizacji, które wcześniej nie miały dostępu do zaawansowanych rozwiązań. Dzięki chmurze wystarczy pomysł, aby korzystać z narzędzi na poziomie globalnych korporacji.

2 Wprowadzenie do Microsoft Azure

Microsoft Azure to publiczna platforma chmurowa oferująca szeroki zakres usług, takich jak obliczenia, przechowywanie danych, sieci, bazy danych i wiele innych. W tej sekcji studenci poznają podstawowe pojęcia związane z platformą Azure oraz nauczą się korzystać z interfejsu użytkownika, aby stworzyć pierwsze zasoby.

2.1 Podstawowe pojęcia

- **Subscription** – Plan subskrypcyjny określający limit zasobów i sposób ich płatności.

- **Tenant** – Logiczna przestrzeń w Microsoft Entra ID (Azure Active Directory), umożliwiająca zarządzanie użytkownikami i zasobami.
- **Resource Group (RG)** – Kontener, który grupuje powiązane zasoby, takie jak maszyny wirtualne, sieci i bazy danych.

2.2 Interfejs użytkownika

Azure oferuje dwa główne sposoby interakcji:

- **Azure Portal** – Graficzny interfejs użytkownika dostępny w przeglądarce.
- **Azure CLI** – Narzędzie wiersza poleceń umożliwiające automatyzację i skryptowanie.

2.3 Ćwiczenie 1: Tworzenie Resource Group

Cel: Stworzenie pierwszej Resource Group w Azure Portal.

1. Zaloguj się do Azure Portal (`https://portal.azure.com`).
2. Kliknij `Create a resource`, a następnie wybierz `Resource Group`.
3. Wprowadź nazwę grupy, np. `example-rg`, oraz wybierz region, np. `West Europe`.
4. Kliknij `Review + Create`, a następnie `Create`.

2.4 Ćwiczenie 2: Tworzenie maszyny wirtualnej w Resource Group

Cel: Utworzenie maszyny wirtualnej z systemem Ubuntu w stworzonej Resource Group.

1. W Resource Group `example-rg`, kliknij `+ Add`, a następnie wybierz `Virtual Machine`.
2. Wprowadź następujące dane:
   - **Name:** `example-vm`.
   - **Region:** użyj innych regionów niż Europe West, może być przepełniony.
   - **Image:** `Ubuntu Server 24.04 LTS`.
   - **Size:** poszukaj czegoś taniego, najlepiej `B1s`.
   - **Authentication type:** Hasło lub klucz SSH.
3. Kliknij `Review + Create`, a następnie `Create`.
4. Po utworzeniu, połącz się z maszyną wirtualną za pomocą SSH:

```
ssh azureuser@<public-ip>
```

5. Sprawdź działanie maszyny wirtualnej, np. wykonując polecenie `uptime`.
6. Usuń wszystkie zasoby, aby uniknąć kosztów. Najlepiej usunąć całą `Resource Group`.

3 Wprowadzenie do Terraform

Terraform to narzędzie typu Infrastructure as Code (IaC), które umożliwia deklaratywne zarządzanie infrastrukturą w chmurze, w tym w Microsoft Azure. W tej sekcji studenci poznają podstawy działania Terraform oraz nauczą się przygotowywać środowisko do pracy z tym narzędziem.

3.1 Podstawowe informacje o Terraform

- **Czym jest Terraform?**
  - Narzędzie open-source opracowane przez firmę HashiCorp.
  - Pozwala na deklaratywne definiowanie infrastruktury w plikach konfiguracyjnych.
  - Wspiera wiele dostawców chmurowych, takich jak Azure, AWS, Google Cloud.
- **Zalety Terraform:**
  - Powtarzalność konfiguracji – eliminacja błędów manualnych.
  - Możliwość wersjonowania infrastruktury.
  - Szybkie skalowanie i zarządzanie zasobami.

3.2 Instalacja Terraform

Kroki instalacji Terraform:

1. Pobierz Terraform z oficjalnej strony: `https://www.terraform.io/downloads`.
2. Rozpakuj archiwum i skopiuj plik wykonywalny do katalogu dostępnego w `PATH`, np. `/usr/local/bin`.
3. Sprawdź instalację, uruchamiając polecenie:

   ```
   terraform version
   ```

3.3 Przygotowanie środowiska do pracy z Azure

Terraform wymaga integracji z Microsoft Azure, co można osiągnąć za pomocą narzędzia Azure CLI. Poniższe kroki można pominąć przez wybranie Azure Cloud Shell (ikonka terminala po prawej stronie wyszukiwarki w Azure Portal). Wszystkie polecenia są już zainstalowane w Cloud Shell. Pamiętaj o konfiguracji subskrypcji, jeśli masz więcej niż jedną.

1. Zainstaluj Azure CLI, postępując zgodnie z instrukcją na stronie: `https://learn.microsoft.com/en-us/cli/azure/install-azure-cli`.
2. Zaloguj się do Azure za pomocą polecenia:

   ```
   az login
   ```

3. Skonfiguruj domyślną subskrypcję (opcjonalnie):

   ```
   az account set --subscription <subscription-id>
   ```

3.4 Pierwsze kroki z Terraform

Tworzenie projektu Terraform:

1. Utwórz nowy katalog projektu:

   ```
   mkdir terraform-azure && cd terraform-azure
   ```

2. Utwórz plik `main.tf` z następującą konfiguracją:

   ```
   provider "azurerm" {
     features {}
   }

   resource "azurerm_resource_group" "example" {
     name     = "example-rg"
     location = "West Europe"
   }
   ```

3. Zainicjalizuj Terraform w katalogu projektu:

   ```
   terraform init
   ```

4. Sprawdź, jakie zmiany zostaną wprowadzone:

   ```
   terraform plan
   ```

5. Utwórz Resource Group, wykonując polecenie:

   ```
   terraform apply
   ```

6. Po zakończeniu pracy usuń zasoby:

   ```
   terraform destroy
   ```

4 Zmienne i zaawansowane przykłady w Terraform

Terraform umożliwia dynamiczne zarządzanie konfiguracją za pomocą zmiennych. Dzięki temu można zwiększyć elastyczność i powtarzalność kodu. W tej sekcji studenci nauczą się definiować zmienne, używać ich w zasobach oraz tworzyć bardziej zaawansowane konfiguracje.

4.1 Definiowanie zmiennych w Terraform

Zmienne w Terraform:

- Zmienne są definiowane w pliku `variables.tf`.
- Umożliwiają dynamiczne przypisywanie wartości do konfiguracji.
- Mogą być przekazywane jako parametry wiersza poleceń, pliki `.tfvars` lub mieć wartości domyślne.

Przykład definicji zmiennej:

```
variable "resource_group_name" {
  default = "example-rg"
}

variable "location" {
  default = "East US"
}
```

Użycie zmiennych w konfiguracji:

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

4.2 Ćwiczenie: Tworzenie Resource Group za pomocą zmiennych

1. Utwórz plik `variables.tf` i dodaj następujące zmienne:

   ```
   variable "resource_group_name" {
     default = "example-rg"
   }

   variable "location" {
     default = "East US"
   }
   ```

2. Zaktualizuj plik `main.tf`, aby używał zmiennych:

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

3. Zainicjalizuj Terraform:

   ```
   terraform init
   ```

4. Przeanalizuj plan:

   ```
   terraform plan
   ```

5. Utwórz Resource Group:

   ```
   terraform apply
   ```

4.3 Zaawansowane przykłady: Tworzenie maszyn wirtualnych z dynamiczną konfiguracją

Cel: Wykorzystanie zmiennych do tworzenia kilku maszyn wirtualnych w oparciu o dynamiczną listę.
Przykład konfiguracji: Zaczynamy od stworzenia warstwy sieciowej w Microsoft Azure. Użyjemy do tego `Resource Group` z poprzednich ćwiczeń oraz `Virtual Network` i `Subnet`.

```
resource "azurerm_virtual_network" "example" {
  name                = "example-network"
  address_space       = ["10.0.0.0/16"]
  location            =
     azurerm_resource_group.example.location
  resource_group_name =
     azurerm_resource_group.example.name
}
resource "azurerm_subnet" "example" {
  name                 = "internal"
  resource_group_name  =
     azurerm_resource_group.example.name
  virtual_network_name =
     azurerm_virtual_network.example.name
  address_prefixes     = ["10.0.2.0/24"]
}
```

Przetestuj konfigurację, wykonując polecenie `terraform apply`. Zauważ referencję zasobu `Subnet` do nazwy `Virtual Network`.
Następnie utworzymy interfejsy sieciowe dla 3 maszyn wirtualnych. Użyjemy do tego parametru `count`, który pozwala na tworzenie wielu instancji zasobu na podstawie jednej definicji. Iterator `count.index` pozwala na dynamiczne przypisanie adresów IP i rozróżnienie instancji.

```
resource "azurerm_network_interface" "example" {
  name                = "example-nic-${count.index}"
  location            =
     azurerm_resource_group.example.location
  resource_group_name =
     azurerm_resource_group.example.name

  ip_configuration {
    name                          = "internal"
    subnet_id                     =
       azurerm_subnet.example.id
    private_ip_address_allocation = "Dynamic"
  }
}
```

Zaaplikuj konfigurację i obserwuj jak tworzą się interfejsy sieciowe. Następnie przy pomocy dokumentacji dostępnej na `https://registry.terraform.io` znajdź dokumentację dla providera `azurerm` i poszukaj jak napisać definicję maszyny wirtualnej. Przy pisaniu definicji uwzględniej parametr `count` i odnieś się do interfejsów sieciowych przez odpowiedni parametr, np. `network_interface_ids`.

4.4 Wykorzystanie zmiennych w sieciach i Storage Account

Przykład zmiennych dla sieci:

```
variable "vnet_address_space" {
  default = ["10.0.0.0/16"]
}

resource "azurerm_virtual_network" "vnet" {
  name                = "example-vnet"
  address_space       = var.vnet_address_space
  location            =
     azurerm_resource_group.example.location
  resource_group_name =
     azurerm_resource_group.example.name
}
```

Ćwiczenie:

1. Skonfiguruj zmienne dla sieci (`address_space`) i Subnet (`address_prefixes`).
2. Użyj zmiennych w definicji zasobów.
3. Przeprowadź symulację zmian (`terraform plan`) i wdrożenie (`terraform apply`).

5 Relacje między zasobami w Terraform – Storage Account i VM

Terraform umożliwia tworzenie zależności między zasobami w sposób deklaratywny. Dzięki temu można łatwo integrować różne usługi w chmurze. W tej sekcji studenci nauczą się, jak połączyć Storage Account z maszyną wirtualną w ramach jednej infrastruktury.

5.1 Przykład relacji: Storage Account i maszyna wirtualna

Cel: Utworzenie Storage Account, a następnie skonfigurowanie maszyny wirtualnej, aby korzystała z klucza dostępu do tego Storage Account.

5.2 Konfiguracja Storage Account

```
resource "azurerm_storage_account" "storage" {
  name                     = "examplestorage"
  resource_group_name      =
      azurerm_resource_group.example.name
  location                 =
      azurerm_resource_group.example.location
  account_tier             = "Standard"
  account_replication_type = "LRS"

  tags = {
    environment = "dev"
  }
}
```

5.3 Konfiguracja maszyny wirtualnej z dostępem do Storage Account

```
resource "azurerm_virtual_machine" "example" {
  name                  = "example-vm"
  location              =
      azurerm_resource_group.example.location
  resource_group_name   =
      azurerm_resource_group.example.name
  network_interface_ids =
      [azurerm_network_interface.example.id]
  vm_size               = "Standard_B1s"

  os_disk {
```

```
    caching              = "ReadWrite"
    storage_account_type = "Standard_LRS"
  }

  source_image_reference {
    publisher = "Canonical"
    offer     = "UbuntuServer"
    sku       = "18.04-LTS"
    version   = "latest"
  }

  admin_username = "azureuser"
  admin_password = "P@ssw0rd1234!"

  tags = {
    environment = "dev"
  }

  provisioner "local-exec" {
    command = <<EOT
      echo
          "STORAGE_KEY=${azurerm_storage_account.storage.primary_access_key}"
          >> /etc/environment
    EOT
  }
}
```

5.4 Ćwiczenie: Utworzenie infrastruktury i weryfikacja połączenia

1. Utwórz plik `main.tf` zawierający Resource Group, Storage Account oraz maszynę wirtualną.
2. Użyj klucza dostępu do Storage Account w konfiguracji maszyny wirtualnej:
   - Klucz `primary_access_key` jest odczytywany dynamicznie z zasobu `azurerm_storage_account`.
   - Zmienna środowiskowa `STORAGE_KEY` zostaje zapisana w systemie maszyny wirtualnej.
3. Wykonaj `terraform apply`, aby wdrożyć infrastrukturę.
4. Połącz się z maszyną wirtualną za pomocą SSH:

   ```
   ssh azureuser@<public-ip>
   ```

5. Zweryfikuj obecność zmiennej środowiskowej:

```
cat /etc/environment | grep STORAGE_KEY
```

5.5 Przykład z użyciem dynamicznych zmiennych dla Storage Account

Aby ułatwić skalowanie i elastyczność, można zdefiniować zmienną na poziomie nazwy Storage Account:

```
variable "storage_account_name" {
  default = "examplestorage"
}

resource "azurerm_storage_account" "storage" {
  name                     = var.storage_account_name
  resource_group_name      =
     azurerm_resource_group.example.name
  location                 =
     azurerm_resource_group.example.location
  account_tier             = "Standard"
  account_replication_type = "LRS"
}
```

Ćwiczenie: Zaktualizuj konfigurację Terraform tak, aby nazwa Storage Account była przekazywana jako zmienna. Przetestuj wprowadzone zmiany, uruchamiając:

```
terraform plan
terraform apply
```

6 Konfiguracja PostgreSQL i integracja z maszyną wirtualną

W tej sekcji studenci nauczą się tworzyć zarządzaną bazę danych PostgreSQL w Microsoft Azure oraz łączyć ją z maszyną wirtualną. W ramach zadania studenci skonfigurują serwer PostgreSQL, dostosują reguły zapory sieciowej oraz przetestują połączenie z poziomu maszyny wirtualnej.

6.1 Tworzenie zarządzanego serwera PostgreSQL

Cel: Utworzenie zarządzanej bazy danych PostgreSQL w Azure.

```
resource "azurerm_postgresql_server" "postgresql" {
  name                = "example-postgresql"
```

```
  location            =
      azurerm_resource_group.example.location
  resource_group_name =
      azurerm_resource_group.example.name
  administrator_login = "psqladmin"
  administrator_login_password = "P@ssw0rd1234!"
  sku_name            = "GP_Gen5_2"
  version             = "11"

  storage_mb = 5120
  ssl_enforcement_enabled = false
  auto_grow_enabled = true
}

resource "azurerm_postgresql_database" "exampledb" {
  name                = "exampledb"
  resource_group_name =
      azurerm_resource_group.example.name
  server_name         =
      azurerm_postgresql_server.postgresql.name
  charset             = "UTF8"
  collation           = "English_United States.1252"
}
```

6.2 Konfiguracja reguł zapory sieciowej PostgreSQL

Cel: Skonfigurowanie dostępu do serwera PostgreSQL dla maszyny wirtualnej.

```
resource "azurerm_postgresql_firewall_rule" "allow_vm" {
  name                = "allow-vm"
  resource_group_name =
      azurerm_resource_group.example.name
  server_name         =
      azurerm_postgresql_server.postgresql.name
  start_ip_address    =
      azurerm_public_ip.vm.public_ip_address
  end_ip_address      =
      azurerm_public_ip.vm.public_ip_address
}
```

Wyjaśnienie:

- Reguła zapory zezwala na połączenie do PostgreSQL z określonego adresu IP.
- W tym przypadku IP publiczne maszyny wirtualnej (`azurerm_public_ip.vm.public_ip_address`) jest używane jako źródło.

6.3 Integracja maszyny wirtualnej z PostgreSQL

Cel: Skonfigurowanie maszyny wirtualnej do połączenia z PostgreSQL.

```
resource "azurerm_virtual_machine" "example" {
  name                = "example-vm"
  location            =
     azurerm_resource_group.example.location
  resource_group_name   =
     azurerm_resource_group.example.name
  network_interface_ids =
     [azurerm_network_interface.example.id]
  vm_size             = "Standard_B1s"

  os_disk {
    caching              = "ReadWrite"
    storage_account_type = "Standard_LRS"
  }

  source_image_reference {
    publisher = "Canonical"
    offer     = "UbuntuServer"
    sku       = "18.04-LTS"
    version   = "latest"
  }

  admin_username = "azureuser"
  admin_password = "P@ssw0rd1234!"

  provisioner "remote-exec" {
    inline = [
      "sudo apt update",
      "sudo apt install postgresql-client -y",
      "echo 'export
         DB_HOST=${azurerm_postgresql_server.postgresql.fqdn}'
         >> ~/.bashrc",
      "echo 'export DB_USER=psqladmin' >> ~/.bashrc",
      "echo 'export DB_PASSWORD=P@ssw0rd1234!' >>
         ~/.bashrc",
      "source ~/.bashrc"
    ]
  }
}
```

Wyjaśnienie:

- Klient PostgreSQL jest instalowany na maszynie wirtualnej.

- Zmienna środowiskowa `DB_HOST` zawiera pełną nazwę domenową PostgreSQL (`fqdn`).
- Użytkownik i hasło są zapisane jako zmienne środowiskowe.

6.4 Ćwiczenie: Testowanie połączenia z PostgreSQL

1. Po wdrożeniu infrastruktury połącz się z maszyną wirtualną za pomocą SSH:

   ```
   ssh azureuser@<public-ip>
   ```

2. Zweryfikuj zmienne środowiskowe:

   ```
   echo $DB_HOST
   echo $DB_USER
   echo $DB_PASSWORD
   ```

3. Przetestuj połączenie z bazą danych za pomocą klienta PostgreSQL:

   ```
   psql -h $DB_HOST -U $DB_USER -d exampledb
   ```

4. Wylistuj dostępne tabele w bazie danych:

   ```
   \dt
   ```

6.5 Podsumowanie zadania

Cele osiągnięte w tym zadaniu:

- Tworzenie zarządzanej bazy danych PostgreSQL.
- Konfiguracja reguł zapory sieciowej.
- Integracja maszyny wirtualnej z bazą danych PostgreSQL.
- Testowanie połączenia z bazy danych za pomocą klienta `psql`.

7 Podsumowanie

Podczas tych zajęć zostały omówione kluczowe zagadnienia związane z wykorzystaniem Terraform oraz integracją zasobów w chmurze Microsoft Azure. Główne punkty realizowane podczas zajęć to:

- **Podstawy platformy Microsoft Azure:** Tworzenie Resource Group oraz zarządzanie zasobami za pomocą Azure Portal i Azure CLI.
- **Podstawy Terraform:** Wprowadzenie do Infrastructure as Code, instalacja Terraform i jego integracja z Azure.

- **Tworzenie zasobów:** Automatyzacja tworzenia Resource Group, maszyn wirtualnych, sieci i Storage Account przy użyciu Terraform.
- **Relacje między zasobami:** Łączenie zasobów, takich jak Storage Account i maszyny wirtualne, za pomocą referencji.
- **Zarządzana baza danych PostgreSQL:** Konfiguracja serwera PostgreSQL w Azure, reguł zapory sieciowej oraz integracja z maszyną wirtualną.

7.1 Wnioski

Terraform pozwala na deklaratywne zarządzanie infrastrukturą, co znacząco upraszcza procesy wdrożeniowe w systemach rozproszonych. Kluczowe wnioski:

- Automatyzacja tworzenia zasobów w chmurze pozwala na większą efektywność i eliminację błędów manualnych.
- Integracja różnych typów zasobów, takich jak bazy danych i maszyny wirtualne, umożliwia budowę skalowalnej infrastruktury.
- Testowanie połączeń i funkcjonalności zintegrowanych zasobów pozwala na weryfikację poprawności konfiguracji.

7.2 Kroki na przyszłość

Aby poszerzyć wiedzę, warto:

- Eksperymentować z bardziej zaawansowanymi funkcjami Terraform, takimi jak moduły czy zdalny stan.
- Poznać inne usługi Azure, takie jak Load Balancer czy Azure Kubernetes Service (AKS).
- Zapoznać się z technikami monitorowania zasobów w chmurze oraz integracji z narzędziami, takimi jak Prometheus.

Podczas zajęć zdobyto praktyczne umiejętności w zarządzaniu infrastrukturą chmurową, co jest kluczowym elementem nowoczesnych metod zarządzania systemami rozproszonymi.