INFRASTRUKTURA JAKO KOD

Z wykorzystaniem narzędzia terraform

Radosław Andrusiak

DevOps Engineer | Daftcode

Agenda

1. Wprowadzenie
2. Zapoznanie się ze specyfikacją aplikacji demo
3. Omówienie konceptu Infrastruktura jako kod
4. Omówienie narzędzia terraform
5. Utworzenie infrastruktury dla aplikacji demo

WPROWADZENIE

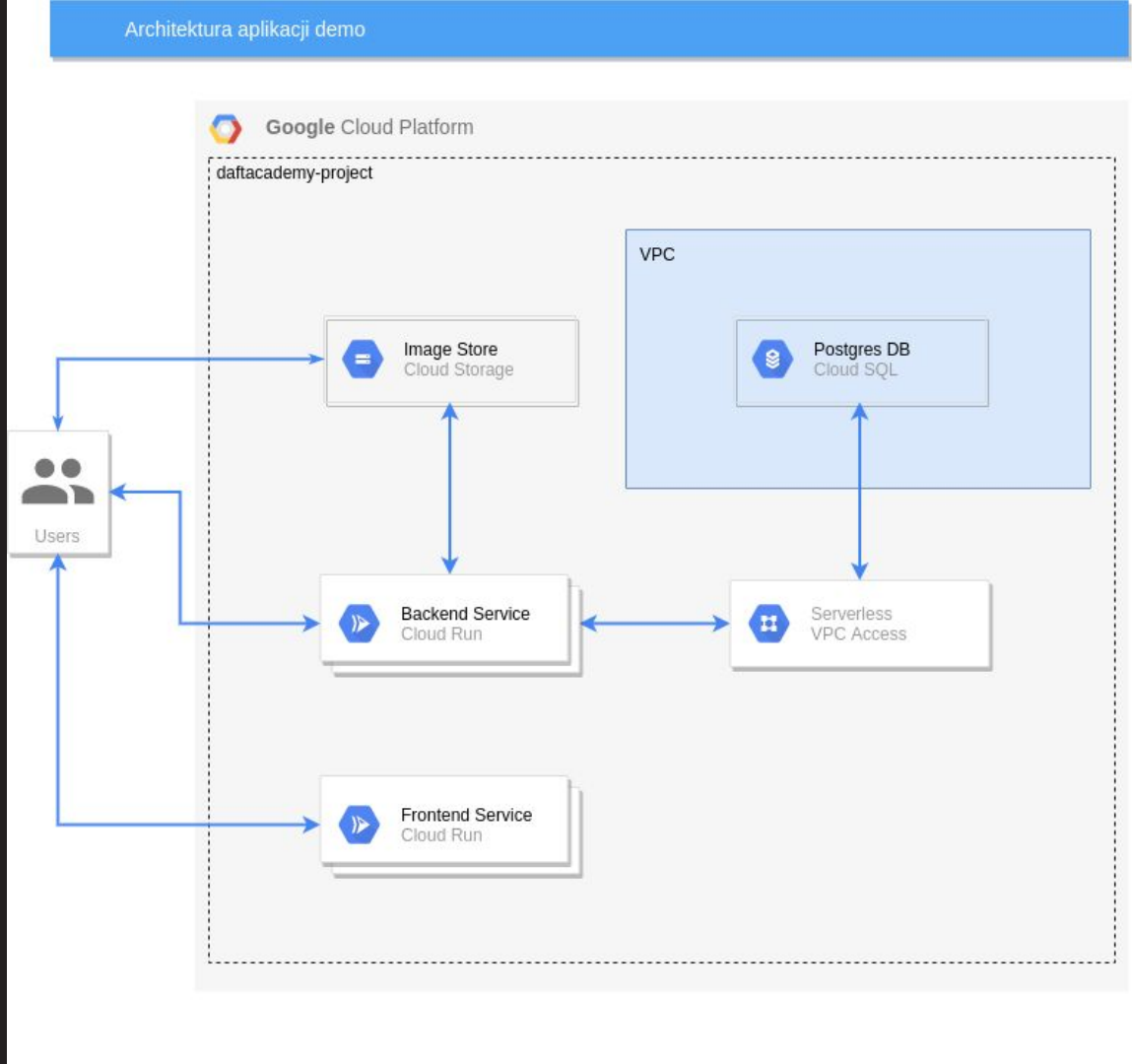
DaftMemes - **specyfikacja**

- Aplikacja umożliwiająca dodawanie, usuwanie oraz przeglądanie memów
- Składa się z dwóch mikroserwisów:
  - Frontend (React)
    - Odpowiada za interakcje z użytkownikiem, komunikuję się z backendem za pomocą REST API
  - Backend (REST API - Golang)
    - Odpowiada za logikę aplikacji tj. dodaje, usuwa oraz zwraca informacje o memach
    - Do działania potrzebuje
      - bazy danych PostgreSQL - w której będzie przechowywać informacje o memach
      - Object storage - w którym będzie mógł przechowywać przesłane obrazy
- Oba serwisy są skonteneryzowane

DaftMemes - **wymagania**

- Każdy z serwisów musi mieć możliwość autoskalowania horyzontalnego
- Każdy z serwisów musi być dostępny przez pojedynczy endpoint (adres IP, domena).
- Baza danych powinna być dostępna tylko z poziomu prywatnej sieci
- Baza danych powinna być wysoce dostępna, posiadać mechanizm failover oraz wykonywać kopie bezpieczeństwa co najmniej raz dziennie (tylko dla środowiska produkcyjnego)
- Dla aplikacji powinny być przygotowane 2 środowiska
    - Środowisko produkcyjne
    - Środowisko testowe

Draft architektury



Lista zadań do wykonania

1. Utworzyć dedykowaną sieć
2. Utworzyć bazę danych PostgreSQL
3. Utworzyć object storage - (Google Cloud Storage)
4. Utworzyć serwisy backend oraz fronted z użyciem usługi Cloud Run

Czas zabierać się **do pracy!**

W jaki sposób podejdziemy do wykonania zadania?

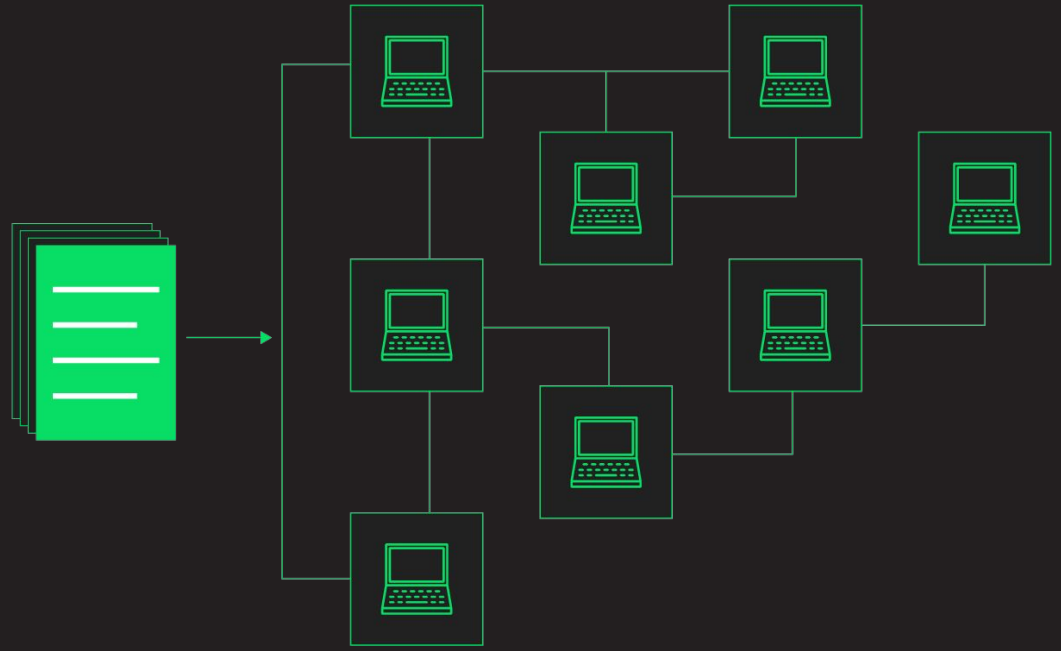
Możemy utworzyć infrastrukturę ręcznie przez konsole GCP ale, takie podejście może się sprawdzić jedynie w przypadku nauki, testów lub zarządzania małym środowiskiem*

W przypadku, gdy chcemy zarządzać infrastrukturą w sposób spójny i powtarzalny oraz chcemy w łatwy sposób skalować nasze środowiska musimy **zautomatyzować** cały proces.

* mimo wszystko nawet przy małym środowisku rekomenduje, żeby zautomatyzować zarządzanie infrastrukturą

Infrastruktura jako kod

Jest to **koncept** w którym traktujemy naszą infrastrukturę jak kod aplikacji tzn. Jesteśmy w stanie opisać komponenty tj. serwery, usługi, sieć itd. za pomocą kodu



Infrastruktura jako kod - **zalety**

- \+ Wersjonowanie za pomocą systemu kontroli wersji
- \+ Code Review
- \+ Reużywalność
- \+ Powtarzalność
- \+ Parametryzacja
- \+ Integracja z systemami CI/CD, które np. przetestują zmiany w naszej infrastrukturze
- \+ Kod jest formą dokumentacji

Infrastruktura jako kod - **narzędzia**

Cloud Deployment Manager

Puppet

Ansible

Terraform

I wiele, wiele innych

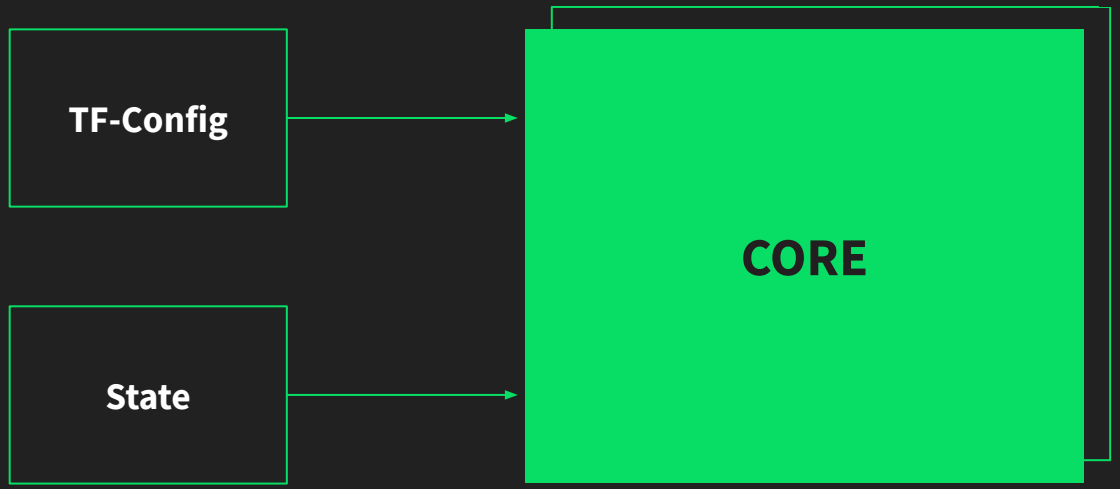
Terraform

Narzędzie open-source firmy hashicorp

Deklaratywny język

Wspiera wiele chmur obliczeniowych i technologii

Terraform - **jak działa?**



PROVIDERS:

- AWS | GCP | [IaaS]
- Kubernetes | [PaaS]
- Slack

Terraform - **Hashicorp Configuration Language**

Składnia języka posiada dwie konstrukcje: argumenty oraz bloki.

- **Argumenty** - przypisują wartość do określonej nazwy.

```
project_name = "daftacademy-tf-intro"
```

- **Bloki** - zazwyczaj przedstawiają konfigurację jakiegoś obiektu.

```
<BLOCK TYPE> "<BLOCK LABEL>" "<BLOCK LABEL>" {
  # Block body
  <IDENTIFIER> = <EXPRESSION> # Argument
}
```

Terraform - **HCL**

- **Podstawowe struktury danych**
  - String
  - Number
  - Boolean
  - etc.
- **Komentarze**
  - Jednowierszowe - # lub //
  - Wielowierszowe - /* */
- **Interpolacja ciągów znaków**
- **Instrukcje warunkowe**
- **Wbudowane funkcje**
- **Pętle**

Terraform
- **podstawowe bloki**

- **Resource** - zasób, który zostanie utworzony w chmurze
- **Variable** - dane wejściowe
- **Output** - dane wyjściowe
- **Local** - zmiennie lokalne (mogą zawierać wyrażenia)
- **Data** - zbieranie danych o istniejących zasobach
- **Module** - kontener dla wielu zasobów, które są używane razem

Terraform
- **podstawowe komendy**

- **terraform init** - min. pobiera zależności
- **terraform refresh** - aktualizuje “tfstate” na podstawie aktualnego stanu infrastruktury
- **terraform plan** - tworzy plan wykonania
- **terraform apply** - wykonuje plan
- **terraform destroy** - usuwa zasoby/infrastrukturę
- **terraform fmt** - formatuje kod zgodnie z przyjętą konwencją

Terraform
- **przykładowy plik konfiguracyjny**

```
terraform {
  required_providers {
    google = {
      source  = "hashicorp/google"
      version = "3.69.0"
    }
  }
  required_version = "~> 0.15"
}

provider "google" {
  project = "daftacademy-tf-intro"
  region  = "europe-central2"
  zone    = "europe-central2-a"
}

resource "google_compute_network" "vpc_network" {
  name                    = "terraform-network"
  auto_create_subnetworks = false
}

resource "google_compute_subnetwork" "main" {
  name          = "main"
  network       = google_compute_network.vpc_network.id
  ip_cidr_range = "10.10.0.0/16"
}
```

Demo

Lista zadań do wykonania

1. Utworzyć dedykowaną sieć
2. Utworzyć bazę danych PostgreSQL
3. Utworzyć object storage - (Google Cloud Storage)
4. Utworzyć serwisy backend oraz fronted z użyciem usługi Cloud Run

0. Inicjalizacja projektu

Tworzymy nowy katalog lub pobieramy poniższe repozytorium:

git clone git@github.com:randrusiak/daftacademy-gcp-terraform.git

1. Konfiguracja sieci

ZADANIA

01

Utworzenie VPC

02

Rezerwacja puli prywatnych adresów IP dla usług

03

Konfiguracja VPC Peering

04

Utworzenie VPC Access Connector

2. Tworzenie bazy danych PostgreSQL

ZADANIA

01

Utworzenie instancji PostgreSQL

02

Utworzenie użytkownika

03

Bezpieczne przekazanie hasła dla tworzonego użytkownika

04

Utworzenie bazy danych

3. Tworzenie object storage (GCS)

ZADANIA

01 Utworzenie bucketa

02 Nadanie uprawnień tylko do odczytu dla wszystkich użytkowników

4. Tworzenie serwisów frontend/backend

ZADANIA

Backend

1. Utworzenie dedykowanego konta serwisowego
2. Nadanie uprawnień zapis-odczyt do bucketa
3. Utworzenie serwisu w Cloud Run
4. Nadanie publicznego dostępu do serwisu

Frontend

1. Utworzenie serwisu Cloud Run
2. Nadanie publicznego dostępu do serwisu

Dodatkowe zadania dla chętnych

01 Wykorzystanie GCS jako remote backend (tfstate)

02 Przeniesienie obrazów backend/frontend do prywatnego container registry (w swoim projekcie)

03 Utworzenie kolejnego środowiska (produkcyjnego) na podstawie przygotowanej konfiguracji z wykorzystaniem terraform workspaces

Dziękuję!