Programowanie w chmurze. Laboratorium 5. Sieci i maszyny wirtualne w chmurze (IaaS). Tworzenie wirtualnych maszyn oraz wirtualnych sieci (VPC) w Amazon EC2. Terraform.

Wprowadzenie :

Infrastruktura jako usługa (IaaS) to model przetwarzania w chmurze, który daje firmom dostęp do zasobów obliczeniowych, takich jak serwery, pamięć masowa i sieci, za pośrednictwem Internetu. W tym modelu dostawca usług chmurowych zarządza całą infrastrukturą, a klienci płacą tylko za to, co wykorzystują. IaaS oferuje wiele korzyści, w tym oszczędności, elastyczność i szybkie wdrażanie. Eliminuje potrzebę inwestowania we własną infrastrukturę, co pozwala firmom obniżyć koszty. Zasoby można łatwo skalować w górę lub w dół w zależności od potrzeb, co zapewnia dużą elastyczność. IaaS umożliwia również szybkie tworzenie i testowanie nowych aplikacji.

Do popularnych dostawców IaaS należą Amazon Web Services (AWS), Microsoft Azure i Google Cloud Platform. Wybór odpowiedniego rozwiązania IaaS zależy od potrzeb firmy, takich jak rodzaj aplikacji, wymagania dotyczące wydajności i budżet.

Ważnymi elementami IaaS są sieci w chmurze i maszyny wirtualne. Sieci w chmurze umożliwiają centralne zarządzanie infrastrukturą sieciową, w tym wirtualnymi routerami i zaporami sieciowymi. Maszyny wirtualne pozwalają na uruchamianie wielu izolowanych środowisk na jednym serwerze fizycznym. Należy jednak pamiętać o kwestiach bezpieczeństwa. Dostawcy IaaS oferują zaawansowane zabezpieczenia, ale firmy powinny również wdrożyć własne środki bezpieczeństwa.

Terraform to narzędzie typu open source stworzone przez firmę HashiCorp, które służy do automatycznego zarządzania infrastrukturą IT w chmurze i poza nią.

Najważniejsze cechy Terraform:

- **Infrastructure as Code (IaC)** — pozwala opisać całą infrastrukturę (serwery, bazy danych, sieci, zasoby chmurowe itp.) w plikach konfiguracyjnych tekstowych.

- **Deklaratywne podejście** — definiujesz *jak* ma wyglądać infrastruktura, a Terraform sam oblicza i wykonuje potrzebne zmiany, aby doprowadzić ją do tego stanu.
- **Wielochmurowość** — działa z wieloma dostawcami usług chmurowych (AWS, Azure, Google Cloud, DigitalOcean i inne) oraz lokalnymi środowiskami.
- **Automatyzacja** — pozwala na łatwe tworzenie, modyfikowanie i usuwanie zasobów bez ręcznej konfiguracji.
- **Stan infrastruktury** — Terraform przechowuje informacje o obecnym stanie infrastruktury, dzięki czemu wie, co zmienić, dodać lub usunąć.

W tej karcie pracy docelowo stworzymy prostą aplikację internetową w technologii React, umożliwiającej odtwarzanie radia internetowego (Antyradio) z funkcją pauzy i wznowienia odtwarzania. Aplikacja dodatkowo będzie wyświetlać aktualną datę i godzinę. To co wykonamy oprócz tego to utworzenie i skonfigurowanie instancji EC2 (Ubuntu) + Nginx + Node.js. Możemy całość zautomatyzować przez Terraform.

Niniejszą kartę pracy należy wykonać w grupie 2-3 osobowej.

UZYSKIWANIE DOSTĘPU DO AWS

AWS oferuje darmowy okres próbny przez 12 miesięcy dla nowych użytkowników, który obejmuje wiele popularnych usług, takich jak EC2 (Elastic Compute Cloud), S3 (Simple Storage Service) i inne. Warto zapoznać się z warunkami korzystania z darmowego konta, aby uniknąć nieprzewidzianych opłat po przekroczeniu limitów.

Aby uzyskać dostęp do Amazon Web Services (AWS) i skorzystać z darmowego okresu próbnego, należy wykonać kilka prostych kroków w celu utworzenia konta.

1. Przejdź na stronę główną AWS i kliknij przycisk „Utwórz konto AWS”.
2. Wprowadź swój adres e-mail, nazwę konta oraz hasło. Upewnij się, że hasło jest silne, aby zapewnić bezpieczeństwo swojego konta.
3. W kolejnym kroku należy podać swoje dane osobowe, takie jak imię, nazwisko oraz adres. Te informacje będą używane do celów rozliczeniowych.
4. Konieczne jest podanie dany karty płatniczej. AWS wymaga tego kroku w celu weryfikacji tożsamości, a także aby móc obciążyć ewentualnymi kosztami, jeśli przekroczysz limity darmowego okresu próbnego. Na początku może być pobrana minimalna kwota (zwykle 1 USD) w celu potwierdzenia karty. Zaleca się użyć karty wirtualnej, ewentualnie takiej którą używamy do innych subskrypcji np. Netflix itp.
5. AWS pozwala na wybór metody weryfikacji tożsamości:
   - Potwierdzenie przez SMS.
   - Potwierdzenie przez połączenie telefoniczne.

   Wybierz preferowaną metodę i postępuj zgodnie z instrukcjami, aby otrzymać kod weryfikacyjny.

6. Po zakończeniu procesu rejestracji należy wybrać plan usług. Dla nowych użytkowników zaleca się wybór podstawowego pakietu, który pozwala na korzystanie z wielu usług AWS w ramach darmowego okresu próbnego przez 12 miesięcy.
7. Podczas tworzenia konta AWS, zostaniesz poproszony o wybór regionu. Region to fizyczna lokalizacja centrów danych AWS. Wybór regionu ma wpływ na koszty, opóźnienia i dostępność usług. Zaleca się wybór regionu, który jest najbliżej Twojej lokalizacji lub lokalizacji Twoich użytkowników, aby zminimalizować opóźnienia.
8. Po zakończeniu wszystkich kroków otrzymasz e-mail z potwierdzeniem aktywacji konta. Po jego aktywacji możesz zalogować się do konsoli AWS i rozpocząć korzystanie z usług.
9. Dodatkowo warto skorzystać z usługi AWS IAM (Identity and Access Management). To usługa, która pomaga w bezpiecznym zarządzaniu dostępem do zasobów AWS. Za pomocą IAM możesz tworzyć użytkowników i grupy, przypisywać im uprawnienia i kontrolować, kto ma dostęp do Twoich zasobów.

TWORZYMY INSTACJĘ EC2 na platformie Amazon Web Services (AWS)

Aby utworzyć instancję EC2 na platformie Amazon Web Services (AWS) oraz uzyskać do niej dostęp za pomocą aplikacji RDP, należy postępować zgodnie z poniższymi krokami.

1. Zaloguj się do swojego konta AWS i w panelu usług wyszukaj "EC2".
2. Kliknij przycisk "Uruchom instancję", aby rozpocząć proces tworzenia nowej instancji.
3. Wypełnij formularz konfiguracji:
   - Nazwa instancji: Możesz nadać jej unikalną nazwę lub pozostawić domyślną.
   - Wybór AMI: Wybierz obraz systemu operacyjnego, klikając na **Ubuntu Server 22.04 LTS** (lub inny Linux)..
   - Typ instancji: Wybierz odpowiedni typ instancji, który odpowiada Twoim wymaganiom obliczeniowym. AWS oferuje szeroki wybór typów instancji EC2, zoptymalizowanych pod kątem różnych zastosowań.
   - Instancje ogólnego przeznaczenia (np. T3, M5): zrównoważone pod względem mocy obliczeniowej, pamięci i zasobów sieciowych, odpowiednie dla szerokiego zakresu aplikacji.
   - Instancje obliczeniowe (np. C5, C6g): oferują wysoką moc obliczeniową, idealne dla zastosowań wymagających dużej mocy procesora, takich jak gry, symulacje naukowe i kodowanie wideo.
   - Instancje zoptymalizowane pod kątem pamięci (np. R5, R6g): posiadają dużą ilość pamięci RAM, przeznaczone dla aplikacji bazodanowych, analizy danych i przetwarzania w pamięci. Wybierz typ instancji, który najlepiej odpowiada Twoim potrzebom i budżetowi. Pamiętaj, że niektóre instancje są objęte darmowym okresem próbnym (np. t2.micro).
4. Ustawienia zabezpieczeń
   - Para kluczy: Wybierz istniejącą parę kluczy lub utwórz nową, klikając "Utwórz nową parę kluczy".
   - Ruch RDP: Upewnij się, że w ustawieniach grupy zabezpieczeń ruch RDP (port 3389) jest dozwolony, aby umożliwić zdalny dostęp do instancji.
   - Zezwól na SSH (port 22, najlepiej tylko z Twojego IP).
   - Zezwól na HTTP (port 80) – jeśli aplikacja będzie serwowana przez Nginx.

- (Opcjonalnie) Zezwól na port 3000, jeśli chcesz testować aplikację bezpośrednio na tym porcie.
- Dysk: Domyślny (8 GB EBS) wystarczy.

5. Dostosuj opcje przechowywania danych zgodnie z potrzebami. AWS oferuje dwa główne typy przechowywania danych dla instancji EC2:
   - Amazon Elastic Block Store (EBS): trwałe i elastyczne rozwiązanie do przechowywania danych, które można dołączać i odłączać od instancji. Dane na woluminach EBS są przechowywane niezależnie od instancji, dzięki czemu są zachowywane nawet po zatrzymaniu lub zakończeniu instancji.
   - Instance Store: tymczasowe rozwiązanie do przechowywania danych, które jest fizycznie dołączone do hosta instancji. Dane na Instance Store są tracone po zatrzymaniu lub zakończeniu instancji. Wybierz typ przechowywania, który najlepiej odpowiada Twoim potrzebom. EBS jest zalecane dla większości zastosowań, ponieważ oferuje trwałość i elastyczność."
6. Następnie kliknij "Uruchom instancję" po prawej stronie, aby zakończyć proces. Po chwili Twoja instancja będzie gotowa do użycia. Jeśli potrzebujesz stałego publicznego adresu IP dla swojej instancji EC2, możesz przypisać do niej Elastic IP. Elastic IP to statyczny adres IP, który pozostaje przypisany do Twojego konta, nawet jeśli zatrzymasz lub uruchomisz ponownie instancję.

UZYSKIWANIE DOSTĘPU DO INSTANCJI EC2 ZA POMOCĄ SSH

Połączenie z instancją (SSH)

1. Pobierz plik klucza .pem, jeśli tworzysz nową parę.
2. Połącz się z terminala:

ssh -i /ścieżka/do/klucza.pem ubuntu@<publiczny_IP_instancji>

KONFIGURACJA WIRTUALNYCH SIECI W AMAZON EC2

Aby skonfigurować wirtualne sieci w Amazon EC2, kluczowym elementem jest zrozumienie Virtual Private Cloud (VPC) oraz jego komponentów.

1. Stwórzmy Wirtualną Sieć (VPC) w AWS. Najpierw zaloguj się na swoje konto AWS i przejdź do konsoli zarządzania.
2. W konsoli AWS, wyszukaj „VPC” i kliknij na usługę.
3. Wprowadź nazwę dla swojego VPC oraz wybierz zakres adresów CIDR (np. 10.0.0.0/16). To określi zakres adresów IP, które będą dostępne w Twojej sieci.Kliknij „Utwórz”.
4. Utwórz podsieci (Subnet)
   - W sekcji „Podsieci” kliknij „Utwórz podsieć”.
   - Wybierz wcześniej utworzone VPC.

- Wprowadź nazwę podsieci oraz zakres adresów CIDR (np. 10.0.1.0/24).
- Powtórz ten krok dla innych podsieci, jeśli to konieczne.

5. Przejdź do sekcji „Bramy internetowe” i kliknij „Utwórz bramę internetową”. Nazwij bramę i kliknij „Utwórz”.
6. Po utworzeniu, przypisz bramę do swojego VPC, aby umożliwić dostęp do Internetu.
7. W sekcji „Tabele routingu” kliknij „Utwórz tabelę routingu”. Wybierz swoje VPC i nadaj nazwę tabeli. Dodaj trasę do tabeli, wskazując na bramę internetową jako cel dla adresu 0.0.0.0/0 (wszystkie adresy IP), co pozwoli na dostęp do Internetu. Przypisz tę tabelę do odpowiednich podsieci.
8. Przejdź do sekcji „Grupy zabezpieczeń” i kliknij „Utwórz grupę zabezpieczeń”. Nadaj nazwę grupie i przypisz ją do swojego VPC.
9. Skonfiguruj reguły przychodzące i wychodzące, aby zezwolić na ruch RDP (port 3389) lub SSH (port 22), w zależności od systemu operacyjnego instancji EC2, którą planujesz uruchomić.

Dla prywatnych podsieci, czyli takich, które nie mają bezpośredniego dostępu do Internetu, można skonfigurować NAT Gateway. NAT Gateway to usługa, która umożliwia instancjom w prywatnych podsieciach łączenie się z Internetem, jednocześnie chroniąc je przed bezpośrednim dostępem z zewnątrz. NAT Gateway tłumaczy prywatne adresy IP instancji na publiczny adres IP NAT Gateway, co pozwala im na wysyłanie żądań do Internetu.

TWORZYMY APLIKACJĘ TYPU INTERAKTYWNY ODTWARZACZ RADIOWY

---

1. Utwórz nowy projekt React za pomocą:

npx create-react-app interaktywny-odtwarzacz-radiowy
cd interaktywny-odtwarzacz-radiowy

2. Następnie utwórz następujące pliki:

src/App.js - główny komponent aplikacji
src/RadioPlayer.js - komponent odtwarzacza radiowego src/App.css - plik ze stylami CSS

3. Stwórzmy komponent RadioPlayer. W plik src/RadioPlayer.js i umieść w nim następujący kod:

```
import React, { useState, useRef, useEffect } from 'react';

const stations = {
  Antyradio: 'http://redir.atmcdn.pl/sc/o2/Eurozet/live/antyradio.livx',
  RMF_FM: 'https://www.rmfon.pl/n/rmf_fm.pls',
  Radio_ZET: 'https://stream.radiozet.pl/zet.aac'
};

const RadioPlayer = () => {
  const [isPlaying, setIsPlaying] = useState(false);
  const [volume, setVolume] = useState(1);
```

```
const [currentStation, setCurrentStation] = useState(Object.keys(stations)[0]);
const [currentDateTime, setCurrentDateTime] = useState(new Date());
const audioRef = useRef(null);

useEffect(() => {
  audioRef.current = new Audio(stations[currentStation]);
  audioRef.current.volume = volume;

  const timer = setInterval(() => {
    setCurrentDateTime(new Date());
  }, 1000);

  return () => clearInterval(timer);
}, []);

useEffect(() => {
  if (audioRef.current) {
    audioRef.current.pause();
    audioRef.current = new Audio(stations[currentStation]);
    audioRef.current.volume = volume;
    if (isPlaying) {
      audioRef.current.play();
    }
  }
}, [currentStation]);

const togglePlayPause = () => {
  if (isPlaying) {
    audioRef.current.pause();
  } else {
    audioRef.current.play();
  }
  setIsPlaying(!isPlaying);
};

const handleVolumeChange = (e) => {
  const newVolume = parseFloat(e.target.value);
  setVolume(newVolume);
  if (audioRef.current) {
    audioRef.current.volume = newVolume;
  }
};
```

```
 return (
  <div className="radio-player">
   <h2>Odtwarzacz Radiowy</h2>
   <select value={currentStation} onChange={(e) => setCurrentStation(e.target.value)}>
    {Object.keys(stations).map((station) => (
     <option key={station} value={station}>
      {station}
     </option>
    ))}
   </select>
   <button onClick={togglePlayPause}>
    {isPlaying ? 'Pauza' : 'Odtwórz'}
   </button>
   <div>
    <label>Głośność: </label>
    <input type="range" min="0" max="1" step="0.01" value={volume}
onChange={handleVolumeChange} />
   </div>
   <div className="date-time">
    <p>Data: {currentDateTime.toLocaleDateString()}</p>
    <p>Godzina: {currentDateTime.toLocaleTimeString()}</p>
   </div>
  </div>
 );
};

export default RadioPlayer;
```

Komponent RadioPlayer jest sercem aplikacji. Oto jego główne funkcje:

- Stan odtwarzania: Wykorzystuje hook useState do śledzenia, czy radio jest odtwarzane.
- Aktualna data i czas: Używa useState i useEffect do aktualizacji daty i czasu co sekundę.
- Odtwarzanie audio: Używa useRef do utworzenia referencji do obiektu Audio, który odtwarza strumień radiowy.
- Przycisk odtwarzania/pauzy: Pozwala użytkownikowi kontrolować odtwarzanie.

4. Opracujmy teraz kod, który tworzy podstawową strukturę aplikacji radiowej, definiując jej główne sekcje: nagłówek, zawartość z odtwarzaczem oraz stopkę. Zmodyfikuj plik src/App.js, aby zawierał następującą strukturę:

```
import React, { useEffect, useState } from 'react';
import RadioPlayer from './RadioPlayer';
import './App.css';
import PrivacyPopup from './PrivacyPopup';
```

```
function App() {
  const [location, setLocation] = useState(null);
  const [browserInfo, setBrowserInfo] = useState(null);

  useEffect(() => {
    if (navigator.geolocation) {
      navigator.geolocation.getCurrentPosition(
        (pos) => setLocation(pos.coords),
        (err) => console.warn("Geolokalizacja odrzucona.")
      );
    }

    setBrowserInfo({
      appName: navigator.appName,
      userAgent: navigator.userAgent,
      platform: navigator.platform,
    });
  }, []);

  return (
    <div className="app">
      <header className="header">
        <h1>Radio Internetowe</h1>
      </header>
      <main className="main-content">
        <RadioPlayer />
        {location && (
          <div>
            <p>Twoja lokalizacja: {location.latitude}, {location.longitude}</p>
          </div>
        )}
        {browserInfo && (
          <div>
            <p>Przeglądarka: {browserInfo.appName}</p>
            <p>System: {browserInfo.platform}</p>
            <p>User Agent: {browserInfo.userAgent}</p>
          </div>
        )}
      </main>
      <footer className="footer">
        <p>&copy; 2025 Radio Internetowe. Wszelkie prawa zastrzeżone.</p>
```

```
      </footer>
      <PrivacyPopup />
    </div>
  );
}

export default App;
```

5. App.css – czyli styl css dla tej strony może wyglądać tak:

```
body {
  margin: 0;
  padding: 0;
  font-family: Arial, sans-serif;
  background-color: black;
  color: white;
}

.app {
  display: flex;
  flex-direction: column;
  min-height: 100vh;
}

.header {
  background-color: #222;
  padding: 20px;
  text-align: center;
}

.main-content {
  flex-grow: 1;
  padding: 20px;
  display: flex;
  flex-direction: column;
  align-items: center;
}

.radio-player {
  margin-bottom: 20px;
}
```

```
.date-time {
  text-align: center;
}

.footer {
  background-color: #222;
  padding: 10px;
  text-align: center;
}

button {
  background-color: #4CAF50;
  border: none;
  color: white;
  padding: 10px 24px;
  margin: 10px;
  font-size: 16px;
  cursor: pointer;
}

select, input[type="range"] {
  margin: 10px;
}
```

Plik App.css zawiera style, które nadają aplikacji spójny wygląd:

- Czarne tło z białym tekstem
- Responsywny układ z użyciem flexbox
- Stylizowane przyciski i sekcje

6. Dodaj możliwość wyboru innej stacji radiowej (np. stwórz listę rozwijaną z kilkoma opcjami) oraz zaimplementuj regulację głośności za pomocą suwaka. Nie musi to być zaawansowany poziom
7. Napisz skrypt, który pozwala uzyskać Twoją lokalizację oraz podstawowe informacje na temat przeglądarki. Informacje te zostaną wyświetlone na Twojej stronie z radiem internetowym. Najprostsza implementacja może wykorzystywać wbudowane API przeglądarki, takie jak navigator.geolocation do uzyskania lokalizacji oraz właściwości obiektu navigator do pobrania informacji o przeglądarce. Co ważne korzystanie z geolokalizacji wymaga zgody użytkownika, a niektóre przeglądarki mogą blokować tę funkcję ze względów prywatności.
8. Dodatkowo dodaj politykę prywatności i zgodę użytkownika na zbieranie danych geolokalizacyjnych oraz przetwarzanie plików cookie(w postaci powiadomienia popup z możliwością zamknięcia). Użyj dostępnych generatorów treści:
   https://jakwylaczyccookie.pl/darmowy-notyfikator-cookie/
   https://cyberfolks.pl/generator-polityki-prywatnosci/
   https://jakwylaczyccookie.pl/generator-polityki-cookie/

INSTALACJA ŚRODOWISKA I WDROŻENIE REACT

---

1. Wykonaj poniższe polecenia na instancji EC2:

```
# Aktualizacja systemu i instalacja narzędzi
sudo apt-get update -y
sudo apt-get install -y git nginx curl

# Instalacja Node.js i npm
curl -fsSL https://deb.nodesource.com/setup_20.x | sudo -E bash -
sudo apt-get install -y nodejs

# Pobierz swoją aplikację React (publiczne repozytorium)
git clone https://github.com/TWOJE-REPO/interaktywny-odtwarzacz-radiowy.git
/home/ubuntu/app

cd /home/ubuntu/app
npm install
npm run build

# Konfiguracja Nginx do serwowania aplikacji React
sudo rm -rf /var/www/html/*
sudo cp -r build/* /var/www/html/

sudo bash -c 'cat > /etc/nginx/sites-available/default <<EOL
server {
 listen 80 default_server;
 server_name _;
 root /var/www/html;
 index index.html;
 location / {
  try_files \$uri /index.html;
 }
}
EOL'

sudo systemctl restart nginx
```

Po chwili Twoja aplikacja React będzie dostępna pod publicznym adresem IP instancji EC2 na porcie 80.

AUTOMATYZACJA Z UŻYCIEM TERRAFORM

Możemy zautomatyzować proces uruchomienia aplikacji React, korzystając z pliku main.tf i bloku user_data, który uruchomi skrypt instalujący wymagane pakiety oraz deployujący aplikację.

1. Przykładowy kod Terraform:

```
resource "aws_security_group" "react_app_sg" {
 name        = "react_app_sg"
 description = "Security group for React app"
 vpc_id      = aws_vpc.main.id

 ingress {
  from_port   = 22
  to_port     = 22
  protocol    = "tcp"
  cidr_blocks = ["TWOJ_PUBLICZNY_IP/32"]
 }

 ingress {
  from_port   = 80
  to_port     = 80
  protocol    = "tcp"
  cidr_blocks = ["0.0.0.0/0"]
 }

 egress {
  from_port   = 0
  to_port     = 0
  protocol    = "-1"
  cidr_blocks = ["0.0.0.0/0"]
 }
}

resource "aws_instance" "react_app" {
 ami           = "ami-xxxxxxxx" # Ubuntu 22.04 LTS w Twoim regionie
 instance_type = "t2.micro"
 subnet_id     = aws_subnet.public.id
 vpc_security_group_ids = [aws_security_group.react_app_sg.id]
 associate_public_ip_address = true
 key_name      = "twoja-para-kluczy"

 user_data = <<-EOF
  #!/bin/bash
  apt-get update -y
```

```
  apt-get install -y git nginx curl
  curl -fsSL https://deb.nodesource.com/setup_20.x | bash -
  apt-get install -y nodejs
  git clone https://github.com/TWOJE-REPO/interaktywny-odtwarzacz-radiowy.git /home/ubuntu/app
  cd /home/ubuntu/app
  npm install
  npm run build
  rm -rf /var/www/html/*
  cp -r build/* /var/www/html/
  cat > /etc/nginx/sites-available/default <<EOL
  server {
   listen 80 default_server;
   server_name _;
   root /var/www/html;
   index index.html;
   location / {
    try_files \$uri /index.html;
   }
  }
  EOL
  systemctl restart nginx
 EOF
}
```

- Zmień "ami-xxxxxxxx" na ID oficjalnego Ubuntu w Twoim regionie.
- Podmień repozytorium na własne.
- Ustaw własny adres IP dla SSH w security group

2. Po wdrożeniu aplikacji (ręcznie lub przez Terraform), wejdź na: http://<publiczny_IP_instancji>
3. Jeśli chcesz, by aplikacja była dostępna na innym porcie (np. 3000), otwórz ten port w security group i uruchom aplikację React bezpośrednio (npm start).

4. Dobrze zorganizowany projekt Terraform ułatwia zarządzanie, rozwój i ponowne wykorzystanie kodu. Typowa struktura projektu składa się z kilku kluczowych plików, z których każdy pełni określoną funkcję:

- **main.tf** – zawiera główną logikę, czyli definicje zasobów, wywołania modułów, źródła danych i zmienne lokalne.
- **variables.tf** – definiuje wszystkie zmienne wykorzystywane w projekcie, dzięki czemu konfiguracja jest elastyczna i łatwa do parametryzacji.
- **outputs.tf** – określa dane wyjściowe, czyli wartości zwracane po wdrożeniu infrastruktury (np. adresy IP, identyfikatory zasobów), co ułatwia dalszą automatyzację i integrację.
- **versions.tf** – zawiera wymagania dotyczące wersji Terraform oraz providerów, co zapewnia spójność środowiska.

W większych projektach stosuje się także katalogi na moduły i środowiska (np. modules/, environments/), co pozwala na lepsze zarządzanie kodem i jego wielokrotne użycie

5. **Przykładowy workflow z Terraform**

```
terraform init     # Inicjalizacja projektu
terraform plan     # Sprawdzenie planu wdrożenia
terraform apply    # Wdrożenie infrastruktury
terraform destroy  # Usunięcie zasobów po testach
```

Pamiętaj: Zmień ID obrazu AMI na właściwe dla Twojego regionu oraz dostosuj repozytorium aplikacji do własnych potrzeb

Automatyzacja infrastruktury w chmurze AWS za pomocą Terraform znacząco przyspiesza wdrożenie, minimalizuje ryzyko błędów oraz pozwala łatwo skalować i monitorować aplikacje produkcyjne. Połączenie EC2, ALB, CloudWatch i SNS daje kompletne, profesjonalne środowisko dla nowoczesnych aplikacji webowych.

MONITORING ZUŻYCIA CPU I POWIADOMIENIA EMAIL

Amazon CloudWatch to usługa monitoringu zasobów i aplikacji w chmurze AWS.

- Monitoruje metryki takie jak zużycie CPU, ilość pamięci, sieć, zdrowie instancji itp.
- Pozwala na tworzenie alarmów (CloudWatch Alarms), które reagują, gdy wartość metryki przekroczy ustawiony próg (np. CPU powyżej 70%).
- Alarmy mogą automatycznie wyzwalać akcje, np. wysyłać powiadomienia na SNS lub uruchamiać skalowanie.

Amazon SNS (Simple Notification Service)

to usługa powiadomień, która umożliwia wysyłanie komunikatów (notyfikacji) do wielu odbiorców lub systemów.

- **Temat SNS (SNS Topic)** to kanał komunikacyjny, do którego można się subskrybować (np. za pomocą emaila, SMS, HTTP endpointu).
- Kiedy coś się wydarzy (np. alarm CloudWatch), wiadomość jest publikowana na temat SNS i automatycznie wysyłana do wszystkich subskrybentów.
- W Twoim przypadku subskrypcją jest adres email — po potwierdzeniu subskrypcji, będziesz dostawać powiadomienia o alarmach.

1. Utwórz temat SNS i dodaj subskrypcję e-mailową (potwierdź subskrypcję klikając link w e-mailu).
2. W CloudWatch stwórz alarm dla metryki CPUUtilization instancji EC2.
3. Ustaw próg np. 70% CPU przez 2 okresy po 5 minut.
4. Jako akcję alarmu wybierz wysłanie powiadomienia na temat SNS.
5. Gdy CPU przekroczy próg, otrzymasz e-mail z powiadomieniem

KONFIGURACJA LOAD BALANCERA (ALB)

Application Load Balancer to usługa równoważenia obciążenia ruchu HTTP(S) na wiele instancji EC2.

- Rozdziela ruch sieciowy na dostępne i zdrowe instancje w grupie docelowej (target group).
- Zapewnia wysoką dostępność, skalowalność i poprawia wydajność aplikacji.
- Umożliwia zaawansowane routowanie (np. na podstawie ścieżki URL, nagłówków, itp.).

1. Utwórz Application Load Balancer (ALB).
2. Skonfiguruj grupę docelową (target group) i dodaj do niej instancje EC2 (najlepiej przez ASG).
3. Ustaw listener na porcie 80 (HTTP).
4. Load Balancer równomiernie rozdzieli ruch między instancje, zapewniając wysoką dostępność i skalowalność.

Wyniki i rozwiązania zanotuj w sprawozdaniu!