|  |  |
| --- | --- |
| Obraz zawierający tekst, Czcionka, logo, Grafika  Opis wygenerowany automatycznie | **Wydział Finansów i Zarządzania Kierunek: Informatyka** |

Daniel Dura

(numer albumu: 76190)

Infrastruktura serwerowa i tworzenie jej za pomocą kodu

Inżynierska praca projektowa

Opiekun merytoryczny:

mgr inż. Wojciech Barczyński

Wrocław 2023

STRESZCZENIE

Głównym celem pracy było utworzenie infrastruktury aplikacji chmurowej AWS przy pomocy kodu oraz porównanie takiego rozwiązania z tradycyjnym podejściem do tworzenia zasobów lokalnie. W pierwszym rozdziale przedstawiono wstęp oraz cel pracy inżynierskiej. W drugim oraz trzecim zawarte są informacje oraz pojęcia o podejściu do tworzenia infrastruktury on-premise i chmurowej. Czwarty rozdział opisuje czym jest infrastruktura jako kod, z jakich narzędzi korzysta, wyjaśnia czym jest Terraform i jaka jest jego historia powstania, a także przedstawia jego open-source’owy odpowiednik – OpenTofu. W piątym rozdziale skupiono się na pokazaniu infrastruktury stworzonej przez kod Terraform w chmurze AWS, jego strukturę oraz wygląd. Szósty rozdział koncentruje się na porównaniu cenowym obu rodzajów podejścia do tworzenia zasobów komputerowych – chmurowego oraz on-premise. Ostatni, siódmy rozdział, przedstawia wnioski i podsumowanie pracy.

Spis treści

[1. Wstęp 5](#_Toc153490216)

[1.1. Cel Pracy 5](#_Toc153490217)

[2. Przedstawienie i omówienie infrastruktury on-premise 6](#_Toc153490218)

[2.1. Co to jest infrastruktura on-premise, jakie są jej zalety i wady? 6](#_Toc153490219)

[2.2. Rodzaje infrastruktury on-premise 7](#_Toc153490220)

[2.3. Wirtualizacja 8](#_Toc153490221)

[2.4. Zabezpieczenia sieci, systemów operacyjnych i aplikacji 10](#_Toc153490222)

[2.5. Zarządzanie i monitorowanie 11](#_Toc153490223)

[2.6. Branże, które wykorzystują głównie lokalną infrastrukturę i dlaczego? 11](#_Toc153490224)

[3. Infrastruktura chmurowa 13](#_Toc153490225)

[3.1. Czym jest chmura? 13](#_Toc153490226)

[3.2. DevOps 14](#_Toc153490227)

[3.3. Liderzy rynku chmurowego 14](#_Toc153490228)

[3.4. Pojęcia związane z Cloud Computing 16](#_Toc153490229)

[3.5. Multicloud 18](#_Toc153490230)

[3.6. Zalety i wady 18](#_Toc153490231)

[3.7. Narzędzia wykorzystywane w chmurze 19](#_Toc153490232)

[4. Infrastruktura jako kod 20](#_Toc153490233)

[4.1. Czym jest Infrastruktura jako kod (IaC)? 20](#_Toc153490234)

[4.2. Popularne narzędzia 21](#_Toc153490235)

[4.3. Czym dokładnie jest Terraform i jak działa? 22](#_Toc153490236)

[4.4. Historia Terraform 23](#_Toc153490237)

[4.5. OpenTofu 24](#_Toc153490238)

[5. Budowa infrastruktury aplikacji chmurowej AWS przy użyciu Terraform 25](#_Toc153490239)

[5.1. Wykorzystane serwisy AWS 26](#_Toc153490240)

[AWS S3 26](#_Toc153490241)

[Czym jest Terraform State i jak wpływa na pracę? 27](#_Toc153490242)

[AWS DynamoDB 28](#_Toc153490243)

[Amazon Virtual Private Cloud 29](#_Toc153490244)

[AWS Load Balancer 30](#_Toc153490245)

[Amazon EC2 & Auto Scalling Group 30](#_Toc153490246)

[Amazon RDS 31](#_Toc153490247)

[5.2. Kod Terraform 32](#_Toc153490248)

[Schemat kodu 32](#_Toc153490249)

[Objaśnienie kodu w folderze „main” 33](#_Toc153490250)

[Pokazanie i wyjaśnienie kodu z folderu „modules” 38](#_Toc153490251)

[Moduł Virtual Private Cloud (VPC) 39](#_Toc153490252)

[Moduł Security Group 43](#_Toc153490253)

[Moduł RDS (Amazon RDS) 46](#_Toc153490254)

[Moduł NAT 48](#_Toc153490255)

[Moduł KEY 51](#_Toc153490256)

[Moduł EC2 Auto Scalling Group (ASG) 52](#_Toc153490257)

[Moduł App Load Balancer (ALB) 56](#_Toc153490258)

[6. Porównanie kosztowe serwerów on-premise z serwerami chmurowymi AWS 58](#_Toc153490259)

[7. Zakończenie 59](#_Toc153490260)

[7.1. Podsumowanie 59](#_Toc153490261)

[Literatura 60](#_Toc153490262)

# 1. Wstęp

W XXI wieku środowisko informatyczne dynamicznie ewoluuje, napotykając na wyzwania związane z rosnącym zapotrzebowaniem na elastyczną, skalowalną i zautomatyzowaną infrastrukturę. W odpowiedzi na te wyzwania narodziła się koncepcja Infrastruktury jako kod (IaC), rewolucyjnej metodyki, która nie tylko zrewolucjonizowała sposób, w jaki postrzegamy i zarządzamy zasobami informatycznymi, ale także odmieniła sposób, w jaki kształtujemy środowisko serwerowe.

## 1.1. Cel Pracy

Celem pracy jest porównanie różnych rodzajów infrastruktur występujących w świecie IT oraz wykorzystanie możliwości technologii Infrastructure as Code, które umożliwiają tworzenie, zarządzanie i utrzymanie infrastruktury za pomocą dedykowanych narzędzi i języków programowania. Zostanie zaprojektowany kod Terraform mający za zadanie zbudować infrastrukturę potrzebną do stworzenia aplikacji webowej w chmurze. Dzięki Terraformowi stworzę środowisko chmurowe składające się z skalujących się serwerów Linux w zależności od obciążenia, baz danych, nadam adresacje sieciową, utworzę przepływ sieciowy między podsieciami oraz wykorzystam resztę kluczowych serwisów chmury Amazon Web Services. Na koniec porównam kosztowo obydwa rozwiązania, a także postaram się ocenić, które jest tym lepszym.

# 2. Przedstawienie i omówienie infrastruktury on-premise

## 2.1. Co to jest infrastruktura on-premise, jakie są jej zalety i wady?

Infrastruktura on-premise jest podejściem do zarządzania zasobami IT   
w organizacji, w której wszystkie systemy informatyczne, serwery, urządzenia sieciowe, oprogramowanie oraz dane są przechowywane, zarządzane i utrzymywane na własnych serwerach wewnętrznych. Jest to przeciwieństwo rozwiązań chmurowych, w których zasoby IT są przechowywane i utrzymywane przez zewnętrznego dostawcę usług w chmurze.

Zalety:

* Pełna kontrola nad danymi firmy,
* Bezpieczeństwo dostosowane bezpośrednio pod organizację, większa kontrola nad bezpieczeństwem zasobów firmy,
* Infrastruktura on-premise umożliwia pełne dostosowanie systemów i oprogramowania do indywidualnych potrzeb organizacji,
* Brak opóźnień związanych z przesyłem danych przez internet.

Wady:

* Utrzymanie własnej infrastruktury IT wiąże się z większymi kosztami początkowymi oraz bieżącymi, takimi jak zakup sprzętu, oprogramowania, opłaty licencyjne czy koszty utrzymania i energii elektrycznej,
* Skalowalność wymagająca dodatkowych zakupów (w tym czasy oczekiwania na dostawy, licencje i inne tym podobne),
* Utrzymanie i wsparcie w 100% należy do właściciela,
* Odporność na awarie, zazwyczaj niższa niż w Data Center uznanego vendora chmurowego.

Podsumowując, taki rodzaj infrastruktury IT może być korzystnym rozwiązaniem dla organizacji, które wymagają pełnej kontroli nad swoimi danymi i systemami, jednak niesie to ze sobą wyższe koszty oraz posiadanie wykwalifikowanej kadry potrafiącej się odnaleźć   
w różnych sytuacjach i awariach związanymi z systemami.

## 2.2. Rodzaje infrastruktury on-premise

Rozwiązania lokalne obejmują różne rodzaje sprzętu i systemów, które umożliwiają działanie środowiska informatycznego w organizacji. Głównie są to:

* Serwery - służące do przechowywania, przetwarzania i zarządzania danymi oraz aplikacjami. Podstawa każdej infrastruktury lokalnej i zwykle są zlokalizowane w dedykowanych pomieszczeniach. Serwery mogą pełnić różne role, takie jak serwery plików, serwery baz danych, serwery aplikacji czy serwery pocztowe.
* Macierze dyskowe - dedykowane systemy przechowywania danych, które są używane do archiwizacji, tworzenia kopii zapasowych i zarządzania dużymi ilościami informacji. Magazyny danych mogą być oparte na technologii HDD, SSD lub taśmowej, a dane mogą być przechowywane w sposób zdecentralizowany lub scentralizowany,   
  w zależności od potrzeb organizacji.
* Urządzenia sieciowe - sprzęt odpowiedzialny za komunikację pomiędzy różnymi elementami infrastruktury on-premise. Do głównych urządzeń sieciowych należą:

1. Switche - odpowiadają za łączenie różnych urządzeń w sieci LAN oraz za przekazywanie danych między nimi.
2. Routery: Routery są odpowiedzialne za kierowanie ruchu sieciowego między różnymi sieciami, w tym między siecią lokalną a Internetem. Pozwalają na łączenie ze sobą różnych urządzeń oraz zarządzanie ruchem danych w sieci.
3. Zapory sieciowe - monitorują ruch sieciowy i chronią sieć organizacji przed nieautoryzowanym dostępem i innymi zagrożeniami. Zapory sieciowe mogą być zarówno sprzętowe, jak i programowe.
4. Access pointy - umożliwiają bezprzewodowe łączenie się z siecią lokalną. Są niezbędne do tworzenia bezprzewodowej infrastruktury sieciowej.

* Systemy zasilania awaryjnego (UPS) i chłodzenia
* Narzędzia zarządzania i monitorowania
* Stacje robocze

Te urządzenia tworzą sieć lokalną LAN, czyli kluczowy element, łączący wszystkie urządzenia, takie jak serwery, stacje robocze i urządzenia sieciowe. Typowe sieci LAN oparte są na kablu Ethernet lub bezprzewodowej technologii Wi-Fi. Architektura sieci może obejmować różne warstwy i podsegmenty, takie jak sieci VLAN, aby zwiększyć bezpieczeństwo i wydajność. Ważnym aspektem sieci jest również łącze internetowe, które umożliwia dostęp do zasobów online oraz zdalne zarządzanie infrastrukturą.

Wszystkie te elementy tworzą kompleksową infrastrukturę, która umożliwia organizacjom przechowywanie, przetwarzanie, zarządzanie i wymianę informacji oraz zasobów w sposób kontrolowany i bezpieczny. W zależności od potrzeb i wymagań organizacji, konfiguracja   
i skala infrastruktury on-premise może się różnić, jednak wymienione powyżej komponenty są kluczowe dla zapewnienia efektywnego środowiska IT.

## 2.3. Wirtualizacja



**Rysunek 1. Porównanie infrastruktury fizycznej z infrastrukturą z wykorzystaniem wirtualizacji**(źródło: https://www.veeam.com/blog/why-virtual-machine-backups-different.html, data dostępu: 07.06.2023)

Wirtualizacja jest technologią, która pozwala na tworzenie wirtualnych instancji serwerów na jednym fizycznym urządzeniu. Proces ten polega na wykorzystaniu oprogramowania zwanego Hypervisorem (Wirtualizatorem) do tworzenia, zarządzania   
i izolowania wirtualnych środowisk serwerowych na jednym fizycznym serwerze. Prościej mówiąc, wirtualizator dzieli serwer na małe części, z których każda wykorzystuje ustaloną wartość jego zasobów. Każde z tych wirtualnych środowisk, zwanych maszynami wirtualnymi, działa niezależnie od siebie, jakby były oddzielnymi serwerami fizycznymi. Obecnie na rynku najpopularniejsze są rozwiązania firm VMware oraz Microsoft (Hyper-V).[[1]](#footnote-1)

Zalety wirtualizacji serwerów:

* Konsolidacja serwerów: Dzięki wirtualizacji możliwe jest skupienie wielu serwerów na jednym sprzęcie. To pozwala na redukcję liczby fizycznych urządzeń w serwerowniach bądź centrach danych, co prowadzi do oszczędności kosztów związanych z zakupem, zarządzaniem i utrzymaniem sprzętu.
* Elastyczność i skalowalność. Wirtualne maszyny mogą być łatwo tworzone, uruchamiane, zatrzymywane i skalowane w zależności od aktualnych potrzeb.   
  To zapewnia elastyczność w zarządzaniu zasobami i umożliwia łatwe dostosowanie się do zmieniających się wymagań biznesowych.
* Izolacja i bezpieczeństwo. W przypadku awarii jednej z maszyn, pozostałe mogą nadal działać niezakłócone. Izolacja ta pomaga również w zwiększaniu bezpieczeństwa, ponieważ atak na jedną maszynę wirtualną nie wpływa na inne.
* Łatwość tworzenia kopii zapasowych i przywracania. Wirtualizacja upraszcza proces tworzenia kopii zapasowych i przywracania staje się bardziej elastyczny i wydajny. Można łatwo wykonać obrazy maszyn wirtualnych i przywrócić je w razie potrzeby, co ułatwia proces zarządzania danymi i ochrony przed utratą danych.

Wady wirtualizacji serwerów:

* Wydajność. W przypadku wymagających aplikacji, które potrzebują dużej ilości zasobów, może być konieczne dokładne dostosowanie i optymalizacja wirtualizatora.
* Awaryjność. W przypadku awarii fizycznego serwera, wszystkie maszyny wirtualne działające na tym serwerze mogą zostać dotknięte. Dlatego ważne jest odpowiednie planowanie i zastosowanie wysokiej dostępności, takiej jak klastry lub replikacja, aby minimalizować ryzyko przerw w działaniu systemów.
* Konieczne jest zapewnienie odpowiedniego monitorowania, konfiguracji, skalowania   
  i zabezpieczenia wirtualnych maszyn.
* Zależność od sprzętu. Serwery, które są wirtualizatorami wymagają odpowiednich podzespołów i infrastruktury do obsługi obciążenia. Aby uzyskać optymalne wyniki, konieczne jest zainwestowanie w odpowiednio wydajne serwery, pamięć masową   
  i sieciowe oraz dbanie o ich zwiększanie ich wydajności w razie wzrostu ilości wirtualnych maszyn.

## 2.4. Zabezpieczenia sieci, systemów operacyjnych i aplikacji

Bezpieczeństwo obejmuje wiele aspektów, w tym zabezpieczanie sieci, systemów operacyjnych i aplikacji. Za zabezpieczenie sieci odpowiedzialnych jest wiele czynników, ale najważniejszymi z nich są:

* Zapory sieciowe monitorujące ruch sieciowy i chroniące sieć przed nieautoryzowanym dostępem oraz innymi zagrożeniami. Zapory sieciowe powinny być konfigurowane zgodnie z zasadą „Zero Trust”,
* Wirtualne sieci prywatne (VPN): VPN pozwala na bezpieczne łączenie się z siecią firmową przez Internet, chroniąc dane przesyłane między zdalnymi użytkownikami   
  a siecią przed przechwytywaniem i manipulacją,
* Segmentacja sieci: Podział sieci na mniejsze segmenty (np. za pomocą sieci VLAN) może zwiększyć bezpieczeństwo, izolując różne części sieci przed potencjalnymi zagrożeniami i atakami,
* Kontrola dostępu.

Do tego dochodzą zabezpieczenia systemów operacyjnych, które wprowadzane są poprzez regularne aktualizacje (a czasami cofanie aktualizacji) wpływające na poprawę luk w systemie, oprogramowanie antywirusowe pomagające chronić system przed znanymi wirusami bądź innymi rodzajami złośliwego oprogramowania. Oprócz tego ważna jest ochrona systemów przed działaniem błędu ludzkiego. To znaczy, że dział IT odpowiedzialny za daną organizację wprowadza zasady bezpieczeństwa ograniczające uprawnienia użytkowników, wymaganie uwierzytelnień i blokadę części akcji jakie użytkownik może podjąć.

W przypadku infrastruktury lokalnej, odpowiedzialność za bezpieczeństwo sieci, systemów operacyjnych i aplikacji spoczywa głównie na organizacji. Dlatego ważne jest, aby wdrożyć odpowiednie strategie i narzędzia zabezpieczające oraz regularnie monitorować i aktualizować środowisko, aby zapewnić jego bezpieczeństwo i zgodność z obowiązującymi regulacjami. Istotnymi czynnościami jest audytowanie i monitorowanie rozwiązań IT, zabezpieczenie danych fizycznie (zabezpieczenie serwerowni w całości spada na organizacje, która dane zasoby posiada) oraz zapewnienie ciągłości działania poprzez zatrudnienie odpowiednich specjalistów potrafiących szybko reagować na incydenty związane z bezpieczeństwem, minimalizując ryzyko utraty danych, przestoju i uszkodzeń reputacji. Do czynności związanych dochodzą również dodatkowe zadania mające na celu wdrożenie polityk i procedur bezpieczeństwa związanych z urządzeniami znajdującymi się w budynku, a także zarządzanie dostępem, szyfrowanie danych i ich utylizacja.

## 2.5. Zarządzanie i monitorowanie

Zarządzanie i monitorowanie infrastruktury IT to proces, który obejmuje kompleksowe nadzorowanie wszystkich elementów infrastruktury IT znajdującej się na terenie przedsiębiorstwa. W skład tej infrastruktury mogą wchodzić serwery, stacje robocze, urządzenia sieciowe, systemy operacyjne, aplikacje biznesowe, bazy danych i wiele innych.   
W jego skład wchodzą:

* Systemy monitoringu: To narzędzia, które umożliwiają śledzenie stanu poszczególnych elementów infrastruktury. Systemy te umożliwiają wykrycie problemów z wydajnością, awarii czy ataków na system. Narzędzia te analizują również wydajność poszczególnych elementów infrastruktury i pozwalają na przewidywanie potencjalnych problemów związanych z ich pracą.
* Systemy zarządzania konfiguracją, które umożliwiają kontrolowanie i zarządzanie konfiguracją urządzeń i aplikacji, takie jak serwery, routery, czy bazy danych. Systemy te pozwalają na automatyzację procesów związanych z wdrażaniem i aktualizacją oprogramowania, co pomaga w zapewnieniu ciągłości działania systemów.
* Narzędzia do zdalnego zarządzania, umożliwiają zdalne zarządzanie i monitorowanie infrastruktury bez konieczności fizycznej obecności w biurze. Narzędzia te umożliwiają wykonywanie zadań administracyjnych, takich jak aktualizacje, tworzenie kopii zapasowych czy rozwiązywanie problemów z wydajnością.
* Narzędzia do automatyzacji procesów, umożliwiają automatyzację procesów biznesowych, takich jak tworzenie raportów, procesów sprzedaży czy wdrażania aktualizacji systemów. Narzędzia te pozwalają na zwiększenie wydajności i skuteczności procesów, a jednocześnie zmniejszenie ryzyka ludzkich błędów.

## 2.6. Branże, które wykorzystują głównie lokalną infrastrukturę i dlaczego?

W dzisiejszych czasach wiele firm korzysta z różnych form infrastruktury IT, w tym zarówno lokalnych, jak i chmurowych. Jednak istnieją branże, w których infrastruktura   
on-premise jest nadal szeroko stosowana. Taki kierunek zostaje obrany głównie dla bezpieczeństwa i utrzymania większej prywatności danych znajdujących się na urządzeniach firmy (na przykład know-how produktu). Oto kilka przykładów branż oraz firm, które głównie wykorzystują lokalną infrastrukturę IT:

* Branża finansowa: Banki, instytucje ubezpieczeniowe i firmy inwestycyjne często mają rozległe systemy informatyczne zbudowane na infrastrukturze lokalnej. W związku z wysokimi wymogami dotyczącymi bezpieczeństwa i poufności danych, wiele z tych firm preferuje utrzymanie swojej infrastruktury wewnętrznie.
* Opieka zdrowotna: Szpitale, kliniki i firmy farmaceutyczne często wykorzystują infrastrukturę lokalną ze względu na wrażliwość danych medycznych i potrzebę szybkiego dostępu do nich. W przypadku ochrony prywatności pacjentów, wiele z tych organizacji preferuje przechowywanie danych w swoich własnych centrach danych.
* Przemysł produkcyjny: Firmy z sektora produkcyjnego, takie jak producenci samochodów, elektroniki czy inne duże fabryki, często mają złożone systemy informatyczne związane z zarządzaniem łańcuchem dostaw, śledzeniem produkcji   
  i optymalizacją procesów. Wiele z nich preferuje infrastrukturę lokalną ze względu na kontrolę nad danymi i wydajność działania systemów..
* Organizacje rządowe: Wiele rządowych instytucji i agencji preferuje utrzymanie infrastruktury IT wewnętrznie ze względu na bezpieczeństwo danych i kontrolę   
  nad systemami. Dotyczy to takich dziedzin jak obrona narodowa, służby wywiadowcze, podatki czy obszar zdrowia publicznego.
* Edukacja: Szkoły, uniwersytety i inne instytucje edukacyjne często wykorzystują lokalną infrastrukturę do zarządzania danymi uczniów, kadry naukowej i infrastrukturą e-learningową. Z uwagi na ochronę danych osobowych uczniów i pracowników, wielu z tych uczelni preferuje utrzymanie infrastruktury wewnętrznie.

Warto jednak zauważyć, że trendem ostatnich lat jest również migracja wielu firm   
do chmury ze względu na elastyczność, skalowalność i koszty. Dlatego w dzisiejszych tak naprawdę najpopularniejszym rozwiązaniem jest hybryda świata chmurowego z światem urządzeń on-premise, nawet tylko poprzez korzystanie z pakietu Microsoft 365 w przedsiębiorstwie (to też rozwiązanie chmurowe gdzie chociażby nie musimy utrzymywać serwera poczty lokalnie).

# 3. Infrastruktura chmurowa

Wraz z rozwojem internetu, wzrostem mocy obliczeniowej i pamięci masowej oraz postępem technologicznym, firma Amazon wprowadziła w 2006 roku usługę Elastic Compute Cloud (EC2) jako część swojej platformy Amazon Web Services (AWS). Na początku stosowali te usługi tylko wewnętrznie na potrzeby firmy, lecz po czasie wpadli na pomysł, że mogliby udostępniać swoje zasoby i pobierać za to opłatę. Był to przełomowy moment, który dał początek szeroko stosowanej infrastrukturze chmurowej.

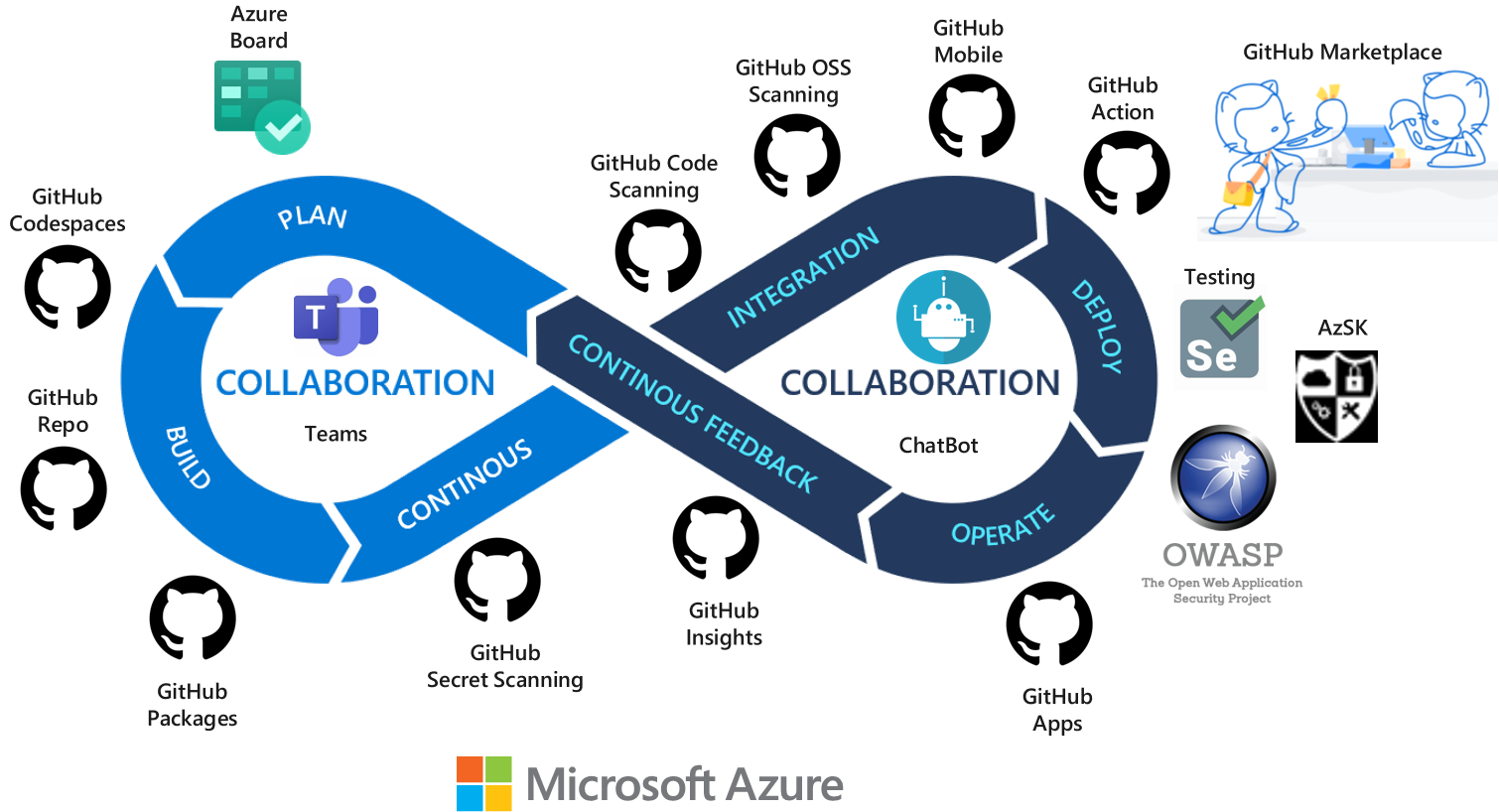
Na dzień dzisiejszy praktycznie każdy dostawców chmurowych oferuję po kilkadziesiąt lub kilkaset dostępnych usług w swojej chmurze. Zakres takich rozwiązań rozciąga się   
od tworzenia maszyn wirtualnych, składowania danych, baz danych relacyjnych i nierelacyjnych po komunikacje z satelitami w czasie rzeczywistym (AWS Ground Station) czy analizę treści obrazu bądź rozpoznowania emocji na podstawie obrazu (Azure Cognitive Services).

## 3.1. Czym jest chmura?

Chmura to bardzo ogólne pojęcie. Do jej bardzo dobrego opisania można przytoczyć definicję z książki napisanej przez Arthura Mateosa oraz Jothy Rosenberga:

„Chmura obliczeniowa to w tej chwili najgorętsze słowo wytrych w informatycznym świecie. […] Coraz więcej dostawców, analityków i użytkowników zgadza się, że na najwyższym poziomie chmurę obliczeniową można zdefiniować jako usługi (serwisy) obliczeniowe oferowane przez zewnętrzne podmioty i dostępne na życzenie w dowolnym momencie, skalujące się dynamicznie w odpowiedzi na zmieniające się zapotrzebowanie. Przetwarzanie w chmurze to odejście od dotychczasowej normy wytwarzania systemów informatycznych, ich obsługi i zarządzania nimi. Z punktu widzenia ekonomii przejście na chmurę to nie tylko możliwość uzyskania o wiele większych przychodów — to także większa elastyczność i lekkość projektu.” [[2]](#footnote-2)

## 3.2. DevOps



**Rysunek 2. Przykładowa struktura DevOps z narzędziami usług Azure DevOps oraz Github**  
(źródło: https://learn.microsoft.com/pl-pl/azure/cloud-adoption-framework/ready/considerations/devops-principles-and-practices, data dostępu: 06.12.2023)

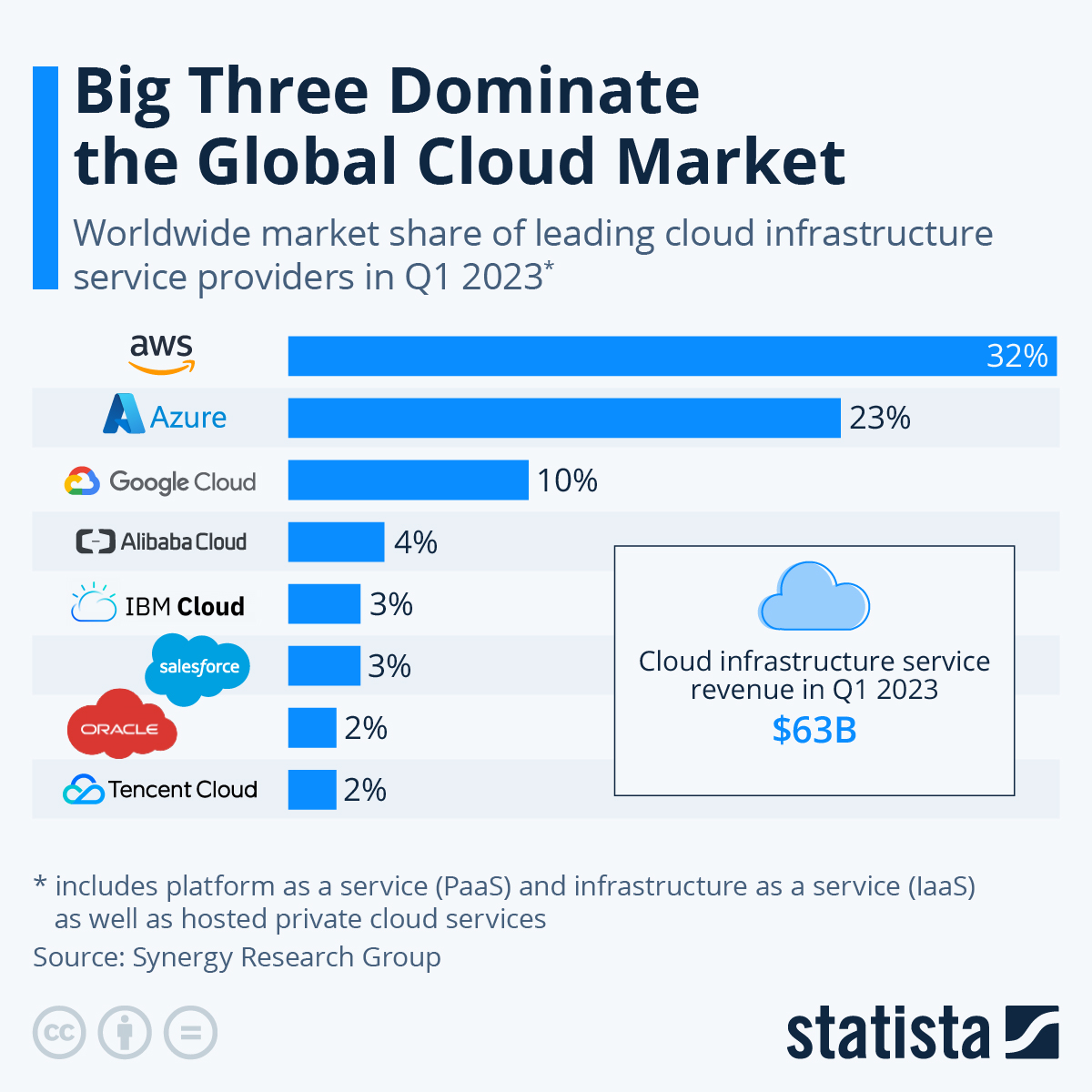
DevOps jest metodyką, który wpływa na cykl życia aplikacji w trakcie jej różnych faz. Pochodzi od połączenia słów „development” oraz „operations”. Opiera się głównie na współpracy międzyludzkiej (ludzi od wytwarzania oprogramowania oraz ludzi od operacji np. inżynierów systemowych), która jest wspieranie przez zautomatyzowane procesy oraz zadania. Jej głównymi filarami są - chmura, narzędzia IaC, konteneryzacja oraz narzędzia CI/CD. DevOps zapewnia niższe koszty oraz większa wydajność oraz efektywność zespołów. W przyszłości, przy udoskonelaniu tego procesu znacząco skraca czas potrzebny do finalizacji projektu oraz zmniejsza liczbę popełnionych błędów.[[3]](#footnote-3)

## 3.3. Liderzy rynku chmurowego

AWS rozpoczął „wyścig zbrojeń” po którym inne wielkie firmy z sektora technologicznego dostrzegły potencjał w udostępnianiu własnych zasobów na zewnątrz.   
W 2008 roku Microsoft przedstawił Azure Cloud, a następnie 1 lutego 2010 roku udostępnił go   
do użytku komercyjnego. Następna była firma Google, która w maju 2010 roku udostępniła swój pierwszy serwis do użytku – Google Cloud Storage.

Na dzień dzisiejszy te 3 firmy posiadają większościowy udział w rynku infrastruktury chmurowej, którego wartość w 2022 roku została oszacowana na 545 miliardów dolarów.

W pierwszym kwartale 2023 roku, gdzie wydatki na infrastrukturę chmurową wyniosły 63 miliardy dolarów, AWS, Azure oraz Google Cloud Platform posiadały łącznie 65% udziałów w tej kwocie.



**Rysunek 3. Rynek wydatków na usługi chmurowe Q1 2023**  
(źródło: https://www.statista.com/chart/18819/worldwide-market-share-of-leading-cloud-infrastructure-service-providers/, data dostępu: 08.06.2023)

Innymi, mniejszymi vendorami są Oracle Cloud, IBM Cloud czy też Alibaba Cloud bądź Salesforce Cloud, którzy skupiają się raczej na specyficznych rozwiązaniach w poszczególnych specjalizacjach. Oczywiście mają też dostępne podstawowe usługi takie jak na przykład wirtualne maszyny, ale względem „Wielkiej Trójki” nie stanowią na tym polu konkurencji.

## 3.4. Pojęcia związane z Cloud Computing

Chmura jest bardzo ogólnym, zbiorczym pojęciem zawierającym w sobie szeroko pojęte dostarczanie zasobów informatycznych na żądanie z opłatą naliczaną tylko w czasie ich wykorzystywania. W rzeczywistości chmurę dzielimy na 3 rodzaje:

* Chmura prywatna/wewnętrzna (private cloud), która jest przeznaczona dla jednej organizacji. Zbudowana z zasobów należących fizycznie lub dzierżawionych przez użytkownika. Korzystają z niej tylko i wyłącznie członkowie danej organizacji. Przykładem takiego rozwiązania jest VMware vCloud.
* Chmura publiczna (public cloud), w której zasoby są udostępniane publicznie przez dostawców chmurowych dla różnych klientów. Dostępna dla każdego (przedsiębiorstwo bądź użytkownik prywatny) i zazwyczaj w obecnym świecie wykorzystywana przez każdego kto nawet o tym nie wie – Microsoft 365 bądź Google Gmail. Zasoby te są skalowalne i dostępne na żądanie. Na przykład Amazon Web Services (AWS) lub Microsoft Azure.
* Chmura hybrydowa (hybrid cloud) to kombinacja publicznej i prywatnej chmury, która umożliwia elastyczność i skalowalność. Organizacje mogą korzystać z prywatnej chmury dla bardziej poufnych danych i używać publicznej chmury dla bardziej elastycznych obciążeń lub uruchamiać zasoby prywatne w chmurze publicznej. Takie rozwiązania umożliwia Azure Stack lub AWS Outposts.

Oprócz rodzajów chmur w chmurze wyróżniamy również kilka modeli dostarczania usług dla klientów. Są to:

* Infrastructure as a Service (IaaS). Model IaaS polega na dostarczeniu do klienta wirtualnej infrastruktury IT. Klient otrzymuje dostęp do różnych zasobów informatycznych, na przykład tworzenia wirtualnych maszyn lub sieci. Dostawca jest odpowiedzialny jedynie za utrzymanie infrastruktury fizycznej, a zarządzanie zasobami pozostawia w gestii klienta. Przykładami są chmury AWS, Google Cloud Platform bądź Microsoft Azure.
* Platform as a Service (PaaS). W modelu PaaS, klient otrzymuje środowisko deweloperskie, które obejmuje system operacyjny, serwer aplikacji, bazy danych   
  i narzędzia deweloperskie. Klient może tworzyć, testować i wdrażać aplikacje bez konieczności zarządzania infrastrukturą pod nimi. Na przykład AWS Elastic Beanstalk lub Heroku.
* Software as a Service (SaaS). W modelu SaaS dostawca chmury dostarcza oprogramowanie jako usługę. Oznacza to, że klient otrzymuje gotowe do użycia aplikacje, które są dostępne na przykład przez przeglądarkę internetową. Klient nie musi zarządzać infrastrukturą, platformą ani oprogramowaniem, ponieważ wszystko to jest zarządzane przez dostawcę. Model ten sprzedawany jest najczęściej w postaci licencji na oprogramowanie. Przykład to Microsoft 365 lub Dropbox.[[4]](#footnote-4)

Obraz zawierający tekst, zrzut ekranu, numer, Czcionka

Opis wygenerowany automatycznie

**Rysunek 4. On-Premises vs IaaS vs PaaS vs SaaS**

(źródło: https://www.eginnovations.com/blog/saas-vs-paas-vs-iaas-examples-differences-how-to-choose/, data dostępu: 08.06.2023)

Oprócz tych 3 podstawowym modeli dostarczania usług powstały jeszcze usługi typu „Serverless” (FaaS/Function as a Service). Oferowane są przez największych dostawców chmurowych (AWS Lambda, Azure Functions i Google Cloud Functions). Znaczenie „bezserwerowe” jest mylące, ponieważ w rzeczywistości serwery istnieją, lecz są niewidoczne dla użytkownika. Vendor dla danej usługi przyznaje zasoby i pamięć obliczeniową przez co klient nie musi się tym martwić. W tym modelu za zasoby płacimy tylko i wyłącznie wtedy gdy funkcje aplikacji bezserwerowa są wykorzystywane. Gdy pozostaje bezczynna nie ponosimy żadnych opłat. Jest to bardzo dobre rozwiązanie dla niewielkich i posiadających małe wymagania odnośnie pamięci obliczeniowej aplikacji. Na przykład funkcje związane   
z wysyłaniem powiadomień bądź proste chatboty.

## 3.5. Multicloud

Multicloud to strategia polegająca na budowaniu środowiska chmurowego przy wykorzystaniu kilku dostawców chmurowych. Na przykład do zarządzania sprzętem IT wykorzystujemy Microsoft Azure, do przechowywania i operacji na danych Google Cloud Platform, a do aplikacji webowych Amazon Web Services. Pojęcie multicloud należy rozróżnić od hybrid cloud, które łączy sprzęt lokalny z chmurą. Wykorzystanie różnych dostawców chmurowych pozwala na dywersyfikację ryzyka i uniezależnienie się od awarii u jednego z dostawców (przez co wszystkie procesy firmowe zostaną wstrzymane), a także może pozwolić na spełnienie niektórych wymagań biznesowych (przykładowo Microsoft Azure posiada certyfikaty bezpieczeństwa takie jak ISO 27001, które mogą być wymagane przez przepisy). Do tego dochodzi kwestia pieniędzy, ponieważ te same usługi mogą się znacznie różnić ceną u różnych dostawców, a także ich jakość może być różna.[[5]](#footnote-5)

## 3.6. Zalety i wady

Zalety:

* Skalowalność. Chmura umożliwia elastyczne skalowanie zasobów, takich jak moc obliczeniowa, pamięć masowa i przepustowość sieci. Można szybko zwiększać   
  lub zmniejszać zasoby w zależności od potrzeb, co pozwala lepiej dostosować się   
  do zmieniających się wymagań biznesowych.
* Elastyczność. Infrastruktura chmurowa oferuje szeroki zakres usług i możliwości,   
  co umożliwia dostosowanie się do różnych potrzeb aplikacji i biznesowych. Można łatwo wdrażać nowe aplikacje, testować nowe rozwiązania i szybko reagować na zmiany na rynku.
* Wydajność i dostępność. Dostawcy chmurowi zapewniają wysoką wydajność   
  i dostępność swoich usług. Rozbudowane regiony chmurowe (miejsca centr danych) znajdujące się w wielu miejscach gwarantują minimalne opóźnienia bądź przestoje.
* Koszty. Infrastruktura chmurowa pozwala uniknąć kosztów związanych z zakupem, konserwacją i zarządzaniem własnym sprzętem i infrastrukturą. Zamiast tego, opłaty   
  za korzystanie z chmury są zwykle oparte na modelu pay-as-you-go, gdzie płaci się tylko za wykorzystane zasoby.
* Ułatwione zarządzanie sprzętem. Dostawcy chmurowi zajmują się zarządzaniem infrastrukturą, aktualizacjami, zabezpieczeniami i innymi aspektami technicznymi.

Wady:

* Bezpieczeństwo i prywatność. Dostęp do danych znajdujących się w chmurze publicznej jest ułatwiony. Istnieje ryzyko naruszenia poufności danych lub utraty kontroli nad nimi, dlatego niektóre organizacje mogą obawiać się przeniesienia wrażliwych danych do chmury.
* Zależność od dostawcy. Przeniesienie infrastruktury do chmury oznacza zależność   
  od dostawcy chmurowego. Jeśli wystąpią problemy z dostępnością usług lub zmiany   
  w warunkach umowy, może to wpływać na działanie biznesu.

Należy pamiętać, że wszystko zależy tak naprawdę od specyfiki przedsiębiorstwa. Dla jednej firmy ułatwiony dostęp do danych może być minusem, a dla drugiej ogromnym plusem i uproszczeniem procesów. Czasami koszty zasobów w chmurze mogą przekraczać koszty postawienia własnej infrastruktury, dlatego do każdego przypadku trzeba podejść osobno i oszacować czy migracja do chmury ma sens.

## 3.7. Narzędzia wykorzystywane w chmurze

Narzędzia stosowane w infrastrukturze chmurowej ułatwiają zarządzanie infrastrukturą, automatyzację procesów, skalowanie zasobów, monitorowanie wydajności oraz zapewnienie bezpieczeństwa i zgodności. Oferują możliwość dostosowania się do zmieniających się potrzeb biznesowych, przyspieszenie i ułatwienie pracy oraz usunięcie monotonnych czynności.

Jednymi z najważniejszych narzędzi stosowanych w chmurze są te związane   
z konteneryzacją, czyli Docker i Kubernetes. Docker jest platformą pozwalająca stawiać wiele lekkich kontenerów (odizolowane środowisko zawierające aplikację i wszystkie jej składniki potrzebne do działania np. biblioteki lub pliki konfiguracyjne) wykorzystującą jeden wspólny system. Dzięki temu nie trzeba stawiać wielu wirtualnych maszyn z każdą aplikacją,   
a wystarcza jeden wydajny serwer. Do tego ułatwia procesy aktualizacji systemu i jego skalowanie. Natomiast Kubernetes (K8s) pozwala zarządzać kontenerami jak i ich grupami umożliwiając ich automatyzację, skalowalność i orkiestrację. Potrafi rozkładać obciążenie, samodzielnie naprawiać awarię i skalować kontenery. Dzięki niemu można łatwo monitorować i zarządzać aplikacjami w środowisku chmurowych, zapewniając niezawodność i wysoką dostępność.

Oprócz narzędzi konteneryzacji ważne są też narzędzia związane z automatyzacją procesów i zarządzania infrastrukturą. Najbardziej popularnym rozwiązaniem jest Terraform (stworzony przez Hashicorp), który pozwala tworzyć infrastrukturę za pomocą kodu. Umożliwia to łatwe śledzenie zmian oraz czytelność i elastyczność konfiguracji. Za jego pomocą możemy na przykład jednorazowo stworzyć 100 instancji VM bez konieczności przeklikiwania się przez cały panel. Cała infrastruktura w narzędziu firmy Hashicorp jest prezentowana za pomocą plików konfiguracyjnych, dlatego można śledzić łatwo jej zmiany na przykład w narzędziu kontroli wersji Git. Innym ważnym elementem Terraform jest planowanie i wdrożenie zmian w infrastrukturze. Umożliwia wstępne planowanie zmian, które pokazuje, jakie zasoby zostaną utworzone, zmodyfikowane lub usunięte. Pozwala to na ocenę wpływu zmian przed ich wdrożeniem. Terraform współpracuje z każdym dużym vendorem chmurowym, a kod różni się niewiele od siebie – główne różnice wynikają z nazewnictwa danego dostawcy. Oprócz Terraform dostępne są również wbudowane narzędzia do tworzenia infrastruktury   
za pomocą kodów u każdego z dostawców chmurowych. W Microsoft Azure będzie to na przykład ARM, a w AWS CloudFormation.

# 4. **Infrastruktura jako kod**

## 4.1. Czym jest Infrastruktura jako kod (IaC)?

Infrastruktura jako Kod (Infrastructure as Code) to koncepcja zarządzania infrastrukturą IT przez oprogramowanie i narzędzia programistyczne, zamiast ręcznego procesu konfiguracji sprzętu. Jest to podejście, które zmienia sposób, w jaki inżynierowie IT wdrażają i zarządzają infrastrukturą sieciową i serwerową, opierając się na kodzie i automatyzacji. Pozwala uprościć zarządzanie ogromną ilością zasobów, ale też wymaga dodatkowej wiedzy specjalistycznej   
w porównaniu do standardowego podejścia.

W ramach IaC, infrastrukturę definiuje się za pomocą skryptów lub plików konfiguracyjnych, co umożliwia automatyzację procesów, zwiększa spójność środowiska   
oraz ułatwia reprodukcję, skalowanie i utrzymanie infrastruktury. Celem IaC jest zminimalizowanie interwencji ręcznej, zapewnienie elastyczności i zwiększenie efektywności w zarządzaniu środowiskiem IT.[[6]](#footnote-6)

## 4.2. Popularne narzędzia

W Infrastructure as Code można wykorzystać wiele narzędzi, które pomagają tworzyć infrastrukturę oraz posiadają swoje unikalne cechy oraz przeznaczenie. Są to między innymi:

* Terraform, stworzony przez HashiCorp, jest jednym z najbardziej popularnych narzędzi. Pozwala na deklaratywne definiowanie infrastruktury i zarządzanie nią na wielu platformach, takich jak AWS, Azure, Google Cloud, czy nawet infrastruktura on-premises. Korzysta z języka HashiCorp Configuration Language (HCL).
* Ansible, narzędzie open source, które umożliwia konfigurację i zarządzanie systemami. Choć bardziej znane jest jako narzędzie do automatyzacji konfiguracji systemów, może być również wykorzystywane jako narzędzie IaC, szczególnie do zarządzania konfiguracją maszyn wirtualnych i kontenerów. Wykorzystuje język YAML.
* AWS CloudFormation, pozwala zarządzać zbiorami usług Amazon Web Services za pomocą wcześniej przygotowanych szablonów definiujących odpowiednie zasoby, usługi, ich konfigurację oraz kolejność uruchamiania. Wykorzystuję języki YAML oraz JSON.
* Azure Resource Manager (ARM), odpowiednik AWS CloudFormation dla chmury Microsoftu, czyli Azure Cloud. ARM’owe szablony pozwalają zarządzać zasobami chmurowymi za pomocą języka JSON.
* Google Cloud Deployment Manager, odpowiednik AWS CloudFormation oraz Azure Resource Manager dla Google Cloud Platform. Pozwala zarządzać zasobami za pomocą języków YAML oraz Python, co odróżnia go od pozostałych dwóch konkurentów.
* Pulumi, umożliwia pisanie Pulumi umożliwia pisanie IaC w dowolnym języku programowania, takim jak Python, JavaScript, Go, C#, Java, YAML czy TypeScript. Pozwala na zarządzanie infrastrukturą na różnych platformach chmurowych (AWS, Azure, GCP, Snowflake)
* Chef, narzędzie do automatyzacji oraz zarządzania konfiguracją. Pozwala na definiowanie konfiguracji systemów i aplikacji za pomocą języka Ruby DSL.
* Puppet, również narzędzie do zarządzania konfiguracją i automatyzacji, które umożliwia programistom i administratorom systemów zdefiniowanie i utrzymanie oczekiwanego stanu konfiguracji na serwerach i maszynach w sposób powtarzalny i skonsolidowany. Pozwala na deklaratywne definiowanie stanu systemu. Wykorzystuje język Puppet DSL.
* OpenTofu, open-source’owy odpowiednik dla Terraform. Pozwala również na tworzenie infrastruktury za pomocą kodu zarówno w środowisku chmurowym jak i on-premise za pomocą tego samego języka deklaratywnego co Terraform, czyli Hashicorp Configuration Language (HCL).

## 4.3. Czym dokładnie jest Terraform i jak działa?

Terraform to narzędzie Infrastructure as Code stworzone przez firmę HashiCorp, umożliwiające definiowanie zarówno zasobów w chmurze, jak i na miejscu, za pomocą czytelnych dla człowieka plików konfiguracyjnych. Kluczową cechą Terraform jest możliwość zarządzania zarówno niskopoziomowymi elementami, takimi jak zasoby obliczeniowe, magazynowe czy sieciowe, jak i elementami wysokopoziomowymi, takimi jak wpisy DNS czy funkcje SaaS.

Jeśli chodzi o sposób działania Terraform to można go podzielić na trzy etapy.

1. Tworzenie kodu: W pierwszym etapie definiujemy zasoby, które mogą obejmować różnych dostawców i usługi chmurowe. Przykładowo, chcemy aby nasza konfiguracja wdrażała maszyny wirtualne w sieci Virtual Private Cloud (VPC) z grupami zabezpieczeń i load balancerem.
2. Planowanie (komenda terraform plan): Terraform tworzy plan wykonania, opisuje zadeklarowaną infrastrukturę do jej utworzenia, aktualizacji lub zniszczenia na podstawie istniejącej infrastruktury i konfiguracji (plik terraform state/ .tfstate).
3. Zastosowanie (komenda terraform apply): Po zatwierdzeniu i sprawdzeniu czy nie ma błędów w konfiguracji, Terraform wykonuje proponowane operacje w prawidłowej kolejności, szanując wszelkie zależności między tworzonymi zasobami. Przykładowo, w przypadku aktualizacji właściwości VPC i zmiany liczby maszyn wirtualnych, Terraform odtworzy VPC przed skalowaniem maszyn.

## 4.4. Historia Terraform

Historia Terraform sięga wprowadzenia przez Amazon Web Services (AWS) CloudFormation w 2011 roku. To wydarzenie skłoniło Mitchella Hashimoto do wyobrażenia sobie potrzeby otwartego, neutralnego wobec chmur rozwiązania, które mogłoby zapewnić spójne workflow na różnych platformach chmurowych. W jednym z postów na blogu   
na platformie Tumblr, idea dla Terraform została zaprezentowana i każdy został zaproszony do utworzenia takiego narzędzia.

Mimo tego, przez kilka lat nie pojawiło się żadne rozwiązanie. W obliczu narastających wyzwań związanych z infrastrukturą i braku odpowiedniego narzędzia, podjęto decyzję o stworzeniu Terraform. W lipcu 2014 roku wydano Terraform 0.1 jako otwartoźródłowe, neutralne wobec chmur narzędzie do zarządzania infrastrukturą jako kodem, początkowo obsługujące tylko AWS i DigitalOcean. Plan zakładał rozpoczęcie od tych dostawców   
i rozszerzenie zautomatyzowanego workflow do dowolnej infrastruktury poprzez dodanie nowych dostawców.

Okres bezpośrednio po wprowadzeniu Terraform charakteryzował się relatywnie niewielką liczbą pobrań, co nawet skłoniło do dyskusji na temat ewentualnego zamknięcia projektu. Niemniej jednak zespół wierzył w długoterminowy potencjał Terraform, zdając sobie sprawę, że sukces zależy od ekosystemu zbudowanego wokół niego, zwłaszcza od wsparcia różnych dostawców. Pod koniec 2016 roku Terraform zyskał rozpęd, z ponad 750 współtwórcami i wsparciem dla dostawców takich jak Microsoft Azure, Google Cloud   
i OpenStack. Rok 2017 był istotnym punktem zwrotnym dla Terraform, ze znacznym   
wzrostem liczby pobrań, podwajającym się co miesiąc. Ten okres został nazwany "rokiem Terraform". Kluczowe wydarzenia tego czasu to między innymi znaczące partnerstwo chmurowe z firmą Microsoft, sygnalizujące silne wsparcie dla Terraform ze strony jednego z głównych dostawców chmurowych. Wprowadzono program dostawców Terraform, ułatwiający tworzenie oficjalnych dostawców, a także uruchomiono Terraform Registry, umożliwiający społeczności publikowanie i dzielenie się wielokrotnego użytku modułami.

Od 2018 do 2020 roku nastąpił zwrot w stronę komercjalizacji Terraform, wprowadzenie Terraform Enterprise i skoncentrowanie się na przyciąganiu dużych organizacji jako płatnych klientów. W tym czasie pojawiła się Terraform 0.12, największe otwarte wydanie, jak dotąd. Obecnie Terraform osiągnął etap oficjalnego produktu 1.0, został pobrany ponad 100 milionów razy i ugruntował swoją pozycję jako standard do efektywnego zarządzania infrastrukturą chmurową. Społeczność odegrała kluczową rolę w sukcesie Terraform, dostarczając ponad tysiąc dostawców i 5,500 modułów.

Z ponad 1200 komercyjnymi klientami Terraform, w tym znaczącymi globalnymi organizacjami, Terraform Enterprise stało się standardem w automatyzacji infrastruktury. Wpływ produktu podkreślają świadectwa klientów, którzy podkreślają jego rolę w zwiększaniu produktywności, zmniejszaniu ryzyka i przyspieszaniu pracy zespołów aplikacyjnych. Terraform stał się integralną częścią transformacji cyfrowej dla organizacji światowej klasy, takich jak 3M, Allstate, Samsung, H&R Block, GitHub i wielu innych.[[7]](#footnote-7)

## 4.5. OpenTofu

OpenTofu, tak samo jak Terraform, to narzędzie do zarządzania infrastrukturą jako kodem (IaC), które umożliwia definiowanie zarówno zasobów w chmurze, jak i on-premises, za pomocą plików konfiguracyjnych, które można wersjonować, ponownie używać   
i udostępniać. Narzędzie to pozwala na spójny przepływ pracy do wdrażania i zarządzania infrastrukturą w całym jej cyklu życia. OpenTofu może zarządzać zarówno komponentami niskiego poziomu, takimi jak zasoby obliczeniowe, magazynowe i sieciowe, jak   
i komponentami wysokiego poziomu, takimi jak wpisy DNS i funkcje SaaS.

Narzędzie te powstało na skutek zmian w licencjonowaniu Terraform na licencję Business Source License (BUSL) przez HashiCorp, co spowodowało co do przyszłości projektu firmy Hashicorp. W odpowiedzi, społeczność zdecydowała się sforkować Terraform, tworząc OpenTofu, które będzie utrzymywane przez Linux Foundation pod licencją Mozilla Public License (MPL) dla zapewnienia otwartości i neutralności narzędzia. Celem OpenTofu jest powrót do pełnej licencji open source i zabezpieczenie przyszłości Terraform przed ewentualnymi nagłymi zmianami licencji.

Działanie OpenTofu jest dokładnie takie samo jak Terraform, czyli umożliwia zarządzanie dowolną infrastrukturą, śledzenie infrastruktury za pomocą pliku stanu, automatyzację zmian, standaryzację konfiguracji za pomocą modułów, oraz współpracę poprzez zapisywanie konfiguracji w systemie kontroli wersji. Narzędzie to stosuje podejście deklaratywne, co oznacza, że opisuje ono stan końcowy infrastruktury, eliminując potrzebę pisania kroków instrukcji. Również jest oparte na języku Hashicorp Configuration Language (HCL).[[8]](#footnote-8)

# 5. Budowa infrastruktury aplikacji chmurowej AWS przy użyciu Terraform

Obraz zawierający tekst, diagram, zrzut ekranu, Prostokąt

Opis wygenerowany automatycznie

**Rysunek 5. Schemat tworzonej infrastruktury w AWS**

(źródło: https://aws.amazon.com/architecture/icons/, praca własna, data dostępu: 08.06.2023)

Powyższy schemat infrastruktury przedstawia serwisy na platformie Amazon Web Services (AWS) stworzone za pomocą kodu Terraform. Jest to schemat infrastruktury potrzebnej do utworzenia i utrzymania aplikacji webowej. Na potrzeby projektu za pomocą Terraform stworzyłem w regionie US-EAST-1 (North Virginia) dwie Availability Zone – US-EAST-1A oraz US-EAST-1B, w których znajdują się NAT, Load Balancer równoważący natężenie ruchu w sieci, wirtualne serwery EC2 w autoskalującej się grupie, która zwiększa lub zmniejsza ich ilość w zależności od obecnych potrzeb. Do tego stworzona została również baza danych aplikacji MySQL wraz z jej backupem. Dla każdego z serwisów utworzone zostały także osobne podsieci, zarówno prywatne jak i publiczne.

## 5.1. Wykorzystane serwisy AWS

### AWS S3

Amazon Simple Storage Service (AWS S3)[[9]](#footnote-9) to usługa przechowywania obiektów w chmurze oferowana przez Amazon Web Services (AWS). Umożliwia przechowywanie i pobieranie dowolnej ilości danych w dowolnym czasie z dowolnego miejsca w internecie. Dzięki różnym klasom przechowywania danych i łatwym w obsłudze funkcjom zarządzania, można zoptymalizować koszty, organizować dane i skonfigurować kontrole dostępu do plików, aby spełnić konkretne wymagania biznesowe, organizacyjne i związane z zgodnością. AWS S3 może być wykorzystywane do:

* Back up’u oraz przywracania kluczowych danych,
* Archiwizacji danych z uwzględnieniem najniższych możliwych opłat,
* Budowy aplikacji chmurowych bądź wykorzystujących pliki zawarte w chmurze,
* Budowy Data Lake (Data Lake - scentralizowane repozytorium, które pozyskuje i przechowuje duże ilości danych w oryginalnej postaci. Ze względu na otwartą, skalowalną architekturę magazyn typu data lake może obsłużyć wszystkie typy danych z dowolnego źródła, od ustrukturyzowanych (tabel bazy danych, arkuszy programu Excel) po częściowo ustruk-turyzowane (pliki XML, strony internetowe) do takich bez struktury (obrazy, pliki dźwiękowe)[[10]](#footnote-10) wykorzystywanych między innymi w uczeniu maszynowym czy sztucznej inteligencji.

Obraz zawierający tekst, zrzut ekranu, oprogramowanie, Oprogramowanie multimedialne

Opis wygenerowany automatycznie

**Fragment kodu 1. Utworzony bucket w usłudze Amazon S3**

W aplikacji Amazon Simple Storage Service użyto do przechowywania aktualnego stanu pliku .tfstate. Dzięki temu każda osoba z zespołu, która pracowałaby nad tą samą infrastrukturą mogłaby działać bez obaw czy jej zmiany wpłyną na negatywnie na pracę pozostałych osób.

### Czym jest Terraform State i jak wpływa na pracę?

Obraz zawierający tekst, zrzut ekranu, oprogramowanie, Oprogramowanie multimedialne

Opis wygenerowany automatycznie

**Fragment kodu 2. Utworzony plik .tfstate w “mojdysk01” S3 bucket**

Jako, że Terraform jest narzędziem do zarządzania infrastrukturą, musi przechowywać informacje na temat jej aktualnego stanu. Domyślnie plik ten jest nazywany „terraform.tfstate” i jest przechowywane lokalnie, lecz dobrą praktyką jest przechowywanie go chmurze (np. AWS S3)/miejscu gdzie jest wersjonowany, szyfrowany i bezpiecznie współdzielony. Plik .tfstate jest używany przez Terraform do mapowania zasobów rzeczywistego świata na Twoją konfigurację, śledzenia metadanych i poprawy wydajności dla dużych infrastruktur.

Terraform używa pliku stanu do określenia, jakie zmiany należy wprowadzić w infrastrukturze. Przed każdą operacją Terraform wykonuje sprawdzenie, aby zaktualizować stan rzeczywistej infrastruktury. Głównym celem stanu Terraform jest przechowywanie powiązań między obiektami w zdalnym systemie a instancjami zasobów zadeklarowanymi w konfiguracji. Kiedy Terraform tworzy zdalny obiekt w odpowiedzi na zmianę konfiguracji, zapisuje tożsamość tego zdalnego obiektu wobec określonej instancji zasobu, a następnie potencjalnie aktualizuje lub usuwa ten obiekt w odpowiedzi na przyszłe zmiany konfiguracji.[[11]](#footnote-11)

### AWS DynamoDB

Amazon DynamoDB to w pełni zarządzana, bezserwerowa usługa bazodanowa oferowana przez Amazon Web Services (AWS). Jest to baza danych NoSQL o wysokiej dostępności, automatycznym skalowaniu i niskich opóźnieniach, zaprojektowana do obsługi szerokiego zakresu aplikacji, od prostych po bardziej zaawansowane, wymagające dużej ilości zapytań i szybkiej odpowiedzi. Stosuje model danych oparty na parze klucz-wartość. DynamoDB jest powszechnie stosowane w aplikacjach internetowych, aplikacjach mobilnych, grach, analizie danych i innych obszarach, gdzie wymagane są szybkie, skalowalne i niezawodne operacje bazodanowe.[[12]](#footnote-12)

Obraz zawierający zrzut ekranu, tekst, oprogramowanie, Oprogramowanie multimedialne

Opis wygenerowany automatycznie

**Fragment kodu 3. Tabela DynamoDB odpowiadająca za lock Terraform**

W tym projekcie, i w zasadzie w każdym innym związanym z Terraform, AWS DynamoDB przy współpracy z AWS S3 działa, aby zabezpieczyć plik .tfstate przed niepożądanym działaniem. To znaczy, że chroni plik przed równoczesnym działaniem dwóch lub więcej procesów/użytkowników uniemożliwiając równoczesne zmiany. Dzięki temu można bez problemu utrzymać infrastrukturę w tym samym stanie podczas pracy w zespole.

### Amazon Virtual Private Cloud

Obraz zawierający tekst, Czcionka, zrzut ekranu

Opis wygenerowany automatycznie

**Fragment kodu 4. Utworzone przez Terraform VPC o nazwie engineeringproject-vpc**

Amazon VPC pozwala użytkownikom na tworzenie izolowanych i niestandardowych sieci w chmurze AWS, co umożliwia im kontrolę nad środowiskiem sieciowym, w którym działają ich zasoby chmurowe. Oferuje możliwość zdefiniowania własnych adresów IP, utworzenia podsieci, zarządzania routingiem, kontrolowania dostępu, a także łączenia się z innymi sieciami, zarówno wewnątrz AWS, jak i na zewnątrz, przy użyciu technologii takich jak VPN (Virtual Private Network) czy Direct Connect.[[13]](#footnote-13)

W projekcie wykorzystano jedną Virtual Private Cloud, którą podzieliłem na sześć podsieci – dwie sieci publiczne z bramą NAT umożliwiającą „kontakt ze światem”, dwie sieci prywatne dla serwerów (po jednej dla każdej z Availability Zone) oraz po dwie sieci prywatne dla baz baz danych (AWS RDS). W tej VPC znalazły się również trzy grupy zabezpieczeń (security groups), jedna dla serwerów, jedna dla baz danych oraz jedna dla Load Balancer’a, który również zawiera się w tym samym VPC.

### AWS Load Balancer

Obraz zawierający tekst, Oprogramowanie multimedialne, oprogramowanie, zrzut ekranu

Opis wygenerowany automatycznie

**Fragment kodu 5. Utworzony przez Terraform load balancer**

AWS Load Balancer jako jedna z kluczowych usług służy do równoważenia obciążenia ruchu między różnymi zasobami lub instancjami w infrastrukturze chmurowej. Load balancer działa jako pośrednik pomiędzy użytkownikami a grupą zasobów, dbając o równomierny rozkład ruchu, zapewniając zrównoważone obciążenie infrastruktury i zwiększając dostępność oraz niezawodność aplikacji. Obsługuje on różne porty, pozwala monitorować przychodzący i wychodzący ruch, a współpracuje z przeróżnymi usługami, na przykład Auto Scalling Group.

W projekcie został skonfigurowany do równoważenia obciążenia, wspólpracy z EC2 Auto Scalling Group oraz udzielania dostępu dla poszczególnych Security Groups na portach 80, 443 i 3306.

### Amazon EC2 & Auto Scalling Group

Amazon Elastic Cloud Compute, czyli Amazon EC2, to usługa chmurowa umożliwiająca tworzenie wirtualnych serwerów. Wirtualne maszyny są wynajmowane przez użytkowników i uruchamiane w wybranych regionach i data center AWS. EC2 ma bardzo elastyczny charakter, instancje mogą być dostosowane do każdej z potrzeb – rozmiar, ilość, typ maszyny. Najbardziej rozpoznawalną cechą tej usługi jest płatność tylko za wykorzystywane zasoby.[[14]](#footnote-14)

Amazon EC2 można połączyć z Auto Scalling Group, czyli usługą umożliwiającą automatyczne zarządzanie i skalowanie grupy instancji EC2. Auto Scalling Group pozwala na dynamiczne dostosowywanie liczby działających instancji w zależności od obciążenia aplikacji lub innych czynników. Głównym celem ASG jest utrzymanie stabilnej i spójnej liczby instancji, a także reagowanie na zmiany w warunkach obciążenia.

Obraz zawierający tekst, oprogramowanie, Oprogramowanie multimedialne, zrzut ekranu

Opis wygenerowany automatycznie

**Fragment kodu 6. Utworzone przez kod Terraform instancje EC2 w Auto Scalling Group**

Terraform w danym projekcie tworzy domyślnie 3 maszyny wirtualne, które są skalowane w zależności od obciążenia procesora. Jeżeli obciążenie procesora co najmniej jednego serwera przekracza 70%, zostanie utworzona kolejna, taka sama instancja, aby zrównoważyć obciążenie. Zostanie również wysłane powiadomienie o utworzeniu kolejnej wirtualnej maszyny. Zadziała to również w drugą stronę, jeżeli co najmniej jedna z instancji będzie miała poniżej 5%, to zostanie usunięta, a administrator zostanie o tym powiadomiony.

### Amazon RDS

Amazon Relational Database Service[[15]](#footnote-15) to w pełni zarządzana usługa baz danych w chmurze. Amazon RDS obsługuje różne rodzaje baz danych, takie jak MySQL, PostgreSQL, MariaDB, Oracle Database i Microsoft SQL Server, więc każdy może dostosować odpowiednią bazę danych do swoich potrzeb. Zapewnia możliwość elastycznego skalowania zasobów bazy danych, zarówno wertykalnie (poprzez zwiększanie mocy obliczeniowej) jak i horyzontalnie (poprzez dodawanie czy usuwanie replik danych). Umożliwia również automatyczne tworzenie kopii zapasowych bazy danych i pozwala na ich łatwe przywracanie w przypadku awarii lub utraty danych.

Obraz zawierający tekst, oprogramowanie, Oprogramowanie multimedialne, zrzut ekranu

Opis wygenerowany automatycznie

**Fragment kodu 7. Baza danych “database01” stworzona przy użyciu Terraform**

Stworzona infrastruktura wykorzystuje Amazon RDS jako serwer bazodanowy dla potencjalnej aplikacji chmurowej. Jest skonfigurowana tak, aby back up tej samej bazy tworzył się w innej Availability Zone.

## 5.2. Kod Terraform

### Schemat kodu

Obraz zawierający tekst, zrzut ekranu, oprogramowanie, Oprogramowanie multimedialne

Opis wygenerowany automatycznie

**Fragment kodu 8. Pliki zawarte w repozytorium Github**

Kod Terraform został podzielony zgodnie z najlepszymi praktykami[[16]](#footnote-16), w którym jedna, tworzona infrastruktura znalazła się w jednym repozytorium Github. Repozytorium podzielono na dwa foldery. W folderze „modules” zostały opisane moduły zasobów AWS (np. bazodanowy RDS), dla których każdy ma swój własny folder. Natomiast w folderze „main” znalazło się wywołanie kodu tworzącego infrastrukturę oraz określenie gdzie znajdywać się będzie plik .tfstate i z jakiego dostawcy chmurowego będziemy korzystać.

### Objaśnienie kodu w folderze „main”

Obraz zawierający tekst, zrzut ekranu, design

Opis wygenerowany automatycznie

**Fragment kodu 9. Pliki lokalne zawarte w fodlerze „main”**

Schemat kodu w folderze „main” na komputerze lokalnym.

Obraz zawierający tekst, zrzut ekranu, Czcionka, oprogramowanie

Opis wygenerowany automatycznie

**Fragment kodu 10. Zawartość pliku terraform.tfvars zawierającego zadeklarowane zmienne**

Plik terraform.tfvars to plik zawierający zadeklarowane zmienne dla całego repozytorium. Nie jest publikowany na Github (poprzez plik .gitignore), ponieważ zawiera wrażliwe dane takie jak dostępy do bazy danych i adresacje sieci, ale bierze czynny udział w konfiguracji infrastruktury. Do tego zawiera informację o regionie i nazwie projektu.

Obraz zawierający tekst, zrzut ekranu, oprogramowanie, Czcionka

Opis wygenerowany automatycznie

**Fragment kodu 11. Kod zawarty w pliku .gitignore mówiący czego nieprzenosić do repozytorium zdalnego**

Plik .gitignore zawiera wykluczane pliki i foldery, które mają trafić do repozytorium Github. Są to między innymi pliki .tfstate, zawierające klucz do połączenia się z konsolą AWS oraz pliki związane z np. logami Terraform.

Obraz zawierający tekst, zrzut ekranu, oprogramowanie, Czcionka

Opis wygenerowany automatycznie

**Fragment kodu 12. Kod zawarty w pliku providers.tf**

Plik „providers.tf” zawiera informację z jakiego dostawcy chmurowego oraz z jakiej wersji będzie korzystać Terraform. Zawiera także informację o regionie, który bierze dane z zadeklarowanej zmiennej (z pliku terraform.tfvars).

Obraz zawierający tekst, zrzut ekranu, Czcionka, oprogramowanie

Opis wygenerowany automatycznie

**Fragment kodu 13. Kod zawarty w pliku backend.tf**

W pliku „backend.tf” zadeklarowane zostało gdzie zapisywać się będzie plik stanu Terraform nazwany „projektinzynierski.tfstate”. Lokalizacją będzie bucket S3 o nazwie „mojdysk01” w regionie us-east-1, a lock state zostanie stworzony w tabeli DynamoDB o nazwie „dynamotable01”.

Obraz zawierający tekst, zrzut ekranu, oprogramowanie

Opis wygenerowany automatycznie

**Fragment kodu 14. Kod pliku main.tf wywołujący utworzone moduły**

Obraz zawierający tekst, zrzut ekranu, Czcionka

Opis wygenerowany automatycznie

**Fragment kodu 15. Dalsza część kodu z pliku main/main.tf**

Plik „main.tf” zawiera wywołanie wszystkich modułów AWS zapisanych w folderze „modules”.

### Pokazanie i wyjaśnienie kodu z folderu „modules”

Obraz zawierający tekst, zrzut ekranu

Opis wygenerowany automatycznie

**Fragment kodu 16. Schemat kodu znajdujące się w folderze „modules”**

W folderze „modules” każdy ze wspomnianych modułów zawiera prawie zawsze 3 te same foldery – „variables.tf” zawierający zmienne zadeklarowane w terraform.tfvars, „output.tf” zwracający zdefiniowane wartości utworzonych zasobów i ułatwiający przez to ich identyfikację oraz „main.tf” zawierający główny kod. Wyjątkiem jest folder „asg” (czyli Auto Scalling Group) zawierający prosty skrypt, a także folder „key” zawierający klucz publiczny do połączenia z instancjami AWS.

### Moduł Virtual Private Cloud (VPC)

Obraz zawierający tekst, zrzut ekranu, oprogramowanie, System operacyjny

Opis wygenerowany automatycznie

**Fragment kodu 17. Kod Terraform tworzący Virtual Private Cloud**

Obraz zawierający tekst, zrzut ekranu, oprogramowanie

Opis wygenerowany automatycznie

**Fragment kodu 18. Dalsza część kodu VPC/main.tf**

Obraz zawierający tekst, zrzut ekranu, oprogramowanie, Oprogramowanie multimedialne

Opis wygenerowany automatycznie

**Fragment kodu 19. Dalsza część kodu VPC/main.tf**

W zasobie VPC na początku tworzony jest Virtual Private Cloud w sieci 10.0.0.0/16, a następnie brama sieciowa umożliwiająca komunikację. W późniejszym etapie tworzone są podsieci, w każdym z dostępnych Availability Zone (sprawdzenie dostępności odbywa się wcześniej). Tworzone są dwie sieci publiczne o adresach 10.0.1.0/24 i 10.0.2.0/24. Następnie tworzone są 4 sieci prywatne o adresach 10.0.3.0/24, 10.0.4.0/24, 10.0.5.0/24 oraz 10.0.6.0/24. Każdemu z tworzonych zasobów nadany jest odpowiadający mu tag.

Obraz zawierający tekst, zrzut ekranu, Czcionka

Opis wygenerowany automatycznie

**Fragment kodu 20. Zmienne wykorzystywane dla modułu VPC**

Obraz zawierający tekst, zrzut ekranu, oprogramowanie

Opis wygenerowany automatycznie

**Fragment kodu 21. Dane wyjściowe modułu VPC**

Powyżej pliki „variables.tf” i „output.tf” zawierające deklaracje zmiennych oraz dane wyjściowe dla tworzonego zasobu.

### Moduł Security Group

Obraz zawierający tekst, zrzut ekranu, oprogramowanie, wyświetlacz

Opis wygenerowany automatycznie

**Fragment kodu 22. Kod Terraform tworzący grupy zabezpieczeń dla chmury AWS**

Obraz zawierający tekst, zrzut ekranu, oprogramowanie, Oprogramowanie multimedialne

Opis wygenerowany automatycznie

**Fragment kodu 23. Dalsza część kodu Terraform tworzącego grupy zabezpieczeń dla chmury AWS**

Obraz zawierający tekst, zrzut ekranu, oprogramowanie, Oprogramowanie multimedialne

Opis wygenerowany automatycznie

**Fragment kodu 24. Dalsza część kodu Terraform tworzącego grupy zabezpieczeń dla chmury AWS**

W module security-group utworzono trzy security groupy – “lb\_security\_group”, która jest grupą zabezpieczeń dla Load Balancera, „client\_sg” dla serwerów EC2 oraz „database\_sg” dla baz danych Amazon RDS. Dla poszczególnych grup zabezpieczeń udostępniono ruch między nimi poprzez otwarcie takich portów jak 80 (http), 443 (https) oraz 3306 (mysql).

Obraz zawierający tekst, oprogramowanie, zrzut ekranu

Opis wygenerowany automatycznie

**Fragment kodu 25. Utworzone Security Groups w chmurze AWS (3 od dołu)**

### Moduł RDS (Amazon RDS)

Obraz zawierający tekst, zrzut ekranu, oprogramowanie, Oprogramowanie multimedialne

Opis wygenerowany automatycznie

**Fragment kodu 26. Kod tworzący bazę danych MySQL w chmurze AWS przy pomocy Terraform**

W pliku „main.tf” modułu RDS opisano bazę danych MySQL. Tworzona jest ona na silniku bazodanowym MySQL w wersji 5.7. Rozmiar przydzielonej pamięci to 20GB, a dane dostępowe są podane we wspominanym wcześniej pliku terraform.tfvars. Multi\_az „true” oznacza, że baza zostanie utworzona wraz w różnych Availability Zone (czyli główna baza w jednym AZ, backup w drugim). Storage\_type „gp2” określa, że zostanie użyty dysk SSD, a skip\_final\_snapshot o wartości true oznacza, że przy usuwaniu bazy nie będzie robiona migawka. Backup\_retention\_period z wartością 0 (dni) oznacza, że backupy nie będą w ogóle przechowywane (Z powodu ewentualnie generowanych kosztów ustawiłem wartość na 0).

Obraz zawierający tekst, zrzut ekranu, Czcionka

Opis wygenerowany automatycznie

**Fragment kodu 27. Zadeklarowane zmienne dla kodu tworzącego instancje RDS**

W pliku „variables.tf” modułu RDS odwołałem się do wcześniej zadeklarowanych zmiennych oraz utworzyłem dwie nowe – „db\_name” oraz „db\_sub\_name”.

### Moduł NAT



**Fragment kodu 28. Kod tworzący bramy NAT oraz EIP dla podsieci publicznych**

Obraz zawierający tekst, zrzut ekranu, oprogramowanie, System operacyjny

Opis wygenerowany automatycznie

**Fragment kodu 29. Kod tworzący tablice routingu oraz dodaje podsieci prywatne do niej**

Obraz zawierający tekst, zrzut ekranu, Czcionka, oprogramowanie

Opis wygenerowany automatycznie

**Fragment kodu 30. Dalsza część kodu tworzącego tablice routingu oraz dodającego podsieci prywatne**

W pliku „main.tf” w folderze NAT znajduję się konfiguracja adresacji i powiązań między sieciami. Na początku nadawane jest EIP (Elastic IP, czyli dynamiczny adres przypisawany dla zasobów w chmurze) dla podsieci publicznych. Następnie do stworzonej tablicy routingu dodawane są pozostałe adresy prywatnych podsieci, aby umożliwić komunikację między nimi.

Obraz zawierający tekst, zrzut ekranu, Czcionka

Opis wygenerowany automatycznie

**Fragment kodu 31. Kod zawierający zmienne dla modułu NAT**

Obraz zawierający tekst, Oprogramowanie multimedialne, oprogramowanie, zrzut ekranu

Opis wygenerowany automatycznie

**Fragment kodu 32. Utworzone NAT przy pomocy kodu w chmurze AWS**

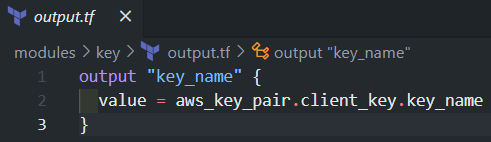
### Moduł KEY

Obraz zawierający tekst, zrzut ekranu, wyświetlacz, oprogramowanie

Opis wygenerowany automatycznie

**Fragment kodu 33. Kod tworzący klucz prywatny oraz powiązujący z nim klucz publiczny**

Moduł ten pozwala na tworzenie kluczy prywatnych do EC2. Zasób aws\_key\_pair tworzy klucz publiczny powiązany z kluczem wygenerowanym przez zasób „tls\_private\_key”, który do szyfru używa 4096 bitowego algorytmu RSA.



**Fragment kodu 34. Dane wyjściowe dla modułu KEY**

Obraz zawierający tekst, zrzut ekranu

Opis wygenerowany automatycznie

**Fragment kodu 35. Utworzona para kluczy w AWS EC2**

### Moduł EC2 Auto Scalling Group (ASG)

Obraz zawierający tekst, zrzut ekranu, oprogramowanie

Opis wygenerowany automatycznie

**Fragment kodu 36. Kod tworzący Auto Scalling Group dla EC2 oraz ustalający jej parametry**

Obraz zawierający tekst, zrzut ekranu, oprogramowanie, System operacyjny

Opis wygenerowany automatycznie

**Fragment kodu 37. Dalsza część kodu tworzącego Auto Scalling Group dla EC2, utworzenie powiadomień poprzez AWS CloudWatch dla obciążenia maszyn**

Obraz zawierający tekst, zrzut ekranu, Czcionka, oprogramowanie

Opis wygenerowany automatycznie

**Fragment kodu 38. Utworzenie powiadomień poprzez AWS CloudWatch dla obciążenia maszyn**

W module Auto Scalling Group opisano wirtualne serwery, które skalują się wedle obciążenia procesora. Do powiadomień o nadmiernym lub zbyt małym zużyciu używana jest usługa AWS CloudWatch. Serwery skalują się wedle zużycia procesora – jeśli wykorzystanie CPU jednej z maszyn jest na poziomie większym bądź równym 70% startuje kolejny, taki sam serwer. Analogicznie dzieje się to samo dla skalowania w dół, tylko usunięcie jednej z maszyn następuję przy 5% zużyciu procesora. Zmiany następują zawsze o 1 wirtualny serwer (w góre lub w dół), a przerwa przed ewentualną następną zmianą wynosi co najmniej 300 sekund.

Obraz zawierający tekst, zrzut ekranu, oprogramowanie

Opis wygenerowany automatycznie

**Fragment kodu 39. Zmienne dla tworzonych zasobów ASG/EC2**

W pliku ze zmiennymi zdefiniowano system, na którym postawione będą serwery ("ami-053b0d53c279acc90", czyli Ubuntu), jej rodzaj (czyli t2.micro), minimalną (2) i maksymalną (6) ilość utrzymywanych maszyn oraz ich docelową/początkową ilość (3). Zmienna „asg\_health\_check\_type” o wartości „ELB” oznacza, że Auto Scaling Group użyje informacji o stanie zdrowia dostarczanych przez Load Balancer do podejmowania decyzji o tym, czy dodać nową instancję do grupy czy też usunąć istniejącą instancję.

### Moduł App Load Balancer (ALB)

Obraz zawierający tekst, zrzut ekranu, oprogramowanie, Oprogramowanie multimedialne

Opis wygenerowany automatycznie

**Fragment kodu 40. Kod tworzący Application Load Balancer**

Obraz zawierający tekst, zrzut ekranu, oprogramowanie, Czcionka

Opis wygenerowany automatycznie

**Fragment kodu 41. Dalsza część kodu tworzącego Application Load Balancer**

Kod z pliku main.tf w folderze “alb” tworzy Load Balancer typu aplikacyjnego (Application Load Balancer), czyli taki który przekierowuje ruch w warstwie aplikacji sieci. Kieruje on ruchem HTTP czy HTTPS. Działa on w pierwszej i drugiej podsieci publicznej nadzorując oraz przekierowując ruch na porcie 80. Wybór takiego typu Load Balancera spowodowany był tym, że infrastruktura ma być gotowa pod tworzenie ewentualnej aplikacji chmurowej.

Obraz zawierający tekst, zrzut ekranu, Czcionka, oprogramowanie

Opis wygenerowany automatycznie

**Fragment kodu 42. Dane wyjściowe dla modułu Application Load Balancer**

Obraz zawierający tekst, Czcionka, zrzut ekranu

Opis wygenerowany automatycznie

**Fragment kodu 43. Zadeklarowane zmienne dla modułu Application Load Balancer**

Powyższy kod określa dane wyjściowe z modułu „alb” oraz zadeklarowane zmienne.

# 6. Porównanie kosztowe serwerów on-premise z serwerami chmurowymi AWS

Zakładając, że cykl życia naszej infrastruktury trwałby 3 lata można dokonać porównania kosztowego powyższej infrastruktury w chmurze AWS z infrastrukturą on-premise. Pod uwagę biorę, że serwery w chmurze działają ze średnim obciążeniem na 4 jednocześnie uruchomionych maszynach wirtualnych.

Tabela . Porównanie cen kupna fizycznego serwera z VM w chmurze AWS o podobnej mocy obliczeniowej w 2023 roku, okres 1 roku oraz 3 lat

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Rodzaj serwera** | **Licencja RHEL Server Standard  (1 rok)** | **Licencja Vmware vSphere Standard  (1 rok)** | **Koszt użytku  (przez 12 miesięcy)** | **Koszty użytku (przez 36 miesięcy)** |
| 4x AWS t3.large EC2 +  Amazon EBS Storage 100 GB/per 1 EC2 instance | n/d | n/d | 1 094,40 USD | 3 283,20 USD |
| Fujitsu PRIMERGY RX2540 M5, Intel® Xeon® Gold 800, 64 GB SDRAM + 2x Fujitsu S26361-F5733-L480 480GB | 799 USD | 1394 USD | 13 273,81 USD | 5 884,23 USD |

(źródło: Kalkulator cen AWS oraz senetic.pl, https://calculator.aws/#/ oraz https://www.senetic.pl, data dostępu: 07.12.2023 )

W tabeli przedstawiono porównanie cen serwerów chmurowych z serwerem fizycznym Fujitsu PRIMERGY RX2540 M5, który posłuży jako wirtualizator dla postawienia 4 maszyn o podobnej specyfikacji co instancje chmurowe. Do ceny serwera fizycznego należy dodać licencję, które umożliwią pełne wykorzystanie jego możliwości. Cena w pierwszych 12 miesiącach obejmuję zakup potrzebnego sprzętu, a następnie po 36 miesiącach jest podzielona na ilość lat, w których jest wykorzystywany sprzęt (w przypadku cen AWS ilość ta jest mnożona trzykrotnie). Kosztami utrzymania infrastruktury on-premise mogą być również rzeczy niepoliczalne takie jak ewentualne awarie sprzętu, konserwacja/utrzymanie czy losowe zdarzenia np. pożar, zalanie serwerowni. W cloud ta odpowiedzialność spada na dostawcę chmurowego. Warto byłoby również rozważyć zakup backupu dla takiego urządzenia. W chmurze możemy wykonywać snapshoty, które będą znacznie tańsze niż zakup drugiego takie samego serwera lub skorzystać z Amazon S3 czy AWS Backup.

# 7. Zakończenie

## 7.1. Podsumowanie

Wynikiem tej pracy jest infrastruktura chmurowa stworzona przy pomocy Terraform w chmurze Amazon Web Services (AWS). Serwery chmurowe, w tym przypadku, pozwalają na większą elastyczność dla osób zarządzających danym środowiskiem. Również skalowalność takiego rozwiązania pozwoli na lepszy „user experience” dla osób korzystających z ewentualnie później powstałej strony WWW. Na podstawie porównania kosztowego można stwierdzić, że jest to również korzystniejsze rozwiązanie dla biznesu. Jednakże, każdy przypadek trzeba rozpatrywać w indywidualnie, dlatego nie można stwierdzić, że infrastruktura chmurowa jest najlepszym rozwiązaniem w każdym przypadku. Czasami koszty mogą nie grać głównej roli, ponieważ czynnikiem, który zdecyduje o innej decyzji biznesowej może być np. brak odpowiednio wykwalifikowanej kadry pracowniczej, która nie rozumie działania chmury bądź wewnętrzne zasady panujące w organizacji, które wymuszają korzystanie tylko i wyłącznie z serwerów lokalnych podyktowane ochroną prywatności danych. Jeszcze w innym wypadku może się okazać, że infrastruktura chmurowa na dłuższą metę nie jest opłacalna dla danego typu przedsiębiorstwa i trzeba będzie powrócić do poprzedniego rozwiązania. Podsumowując, cel pracy został osiągniety.

# Literatura

[1] Marek Serafin, *Wirtualizacja w praktyce*, Helion, Gliwice, 2012, str. 13, 59-62

[2] Arthur Mateos, Jothy Rosenberg, *Chmura obliczeniowa rozwiązania dla biznesu*,   
Helion, Gliwice, 2011

[3] Strona główna Azure, https://azure.microsoft.com/pl-pl/resources/cloud-computing-dictionary/what-is-devops z dnia 06.12.2023

[4] Borko Fuhrt, Armando Escalante, *Handbook of Cloud Computing*, Springer, Nowy Jork, 2010

[5] Jeroen Mulder, *Multi-Cloud Architecture and Governance. Leverage Azure, AWS, GCP, and VMware vSphere to build effective multi-cloud solutions,* Packt Publishing,   
Birmingham, 2020

[6] Kief Morris, *Infrastructure as Code. Dynamic Systems for the Cloud Age,* O’Reilly,   
Sebastopol. 2021

[7] Strona główna Hashicorp, https://www.hashicorp.com/resources/the-story-of-hashicorp-terraform-with-mitchell-hashimoto z dnia 05.12.2023

[8] Strona główna OpenTofu, https://opentofu.org/manifesto/ z dnia 05.12.2023

[9] Strona główna AWS, https://aws.amazon.com/s3/ z dnia 02.12.2023

[10] Strona główna Azure, https://azure.microsoft.com/pl-pl/resources/cloud-computing-dictionary/what-is-a-data-lake z dnia 02.12.2023

[12] Strona dokumentacji Terraform, https://developer.hashicorp.com/terraform/language/state z dnia 02.12.2023

[13] Strona główna AWS, https://aws.amazon.com/dynamodb/ z dnia 02.12.202

[14] Strona dokumentacji AWS, https://docs.aws.amazon.com/vpc/latest/userguide/what-is-amazon-vpc.html z dnia 02.12.2023

[15] Strona główna AWS, https://aws.amazon.com/ec2/ z dnia 02.12.2023

[16] Strona dokumentacji AWS, https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/Welcome.html z dnia 02.12.2023

[17] Strona odnośnie dobrych praktyk kodu Terraform, https://www.terraform-best-practices.com/v/pl/key-concepts z dnia 06.12.2023

Wrocław, dnia …………………….

**OŚWIADCZENIE**

Ja, niżej podpisany/a ......................................................................................................., świadomy/a odpowiedzialności prawnej, student/ka Uniwersytetu WSB Merito we Wrocławiu oświadczam, że przedkładaną pracę dyplomową napisałem/napisałam samodzielnie, korzystając jedynie z pomocy merytorycznej promotora.

Oznacza to, że przy pisaniu pracy, poza niezbędnymi konsultacjami, nie korzystałem/am z pomocy innych osób, a w szczególności nie zlecałem/am opracowania pracy lub jej części innym osobom, ani nie odpisywałem/am tej pracy lub jej części od innych osób. Wszystkie wykorzystane przeze mnie źródła, w tym fragmenty prac i ustalenia innych osób, zostały odpowiednio zaznaczone, z zachowaniem praw autorskich oraz wydawniczych. Praca nie zawiera treści uzyskanych w sposób niezgodny z obowiązującymi przepisami.

Oświadczam ponadto, że niniejsza praca nie została przedstawiona do oceny w innej procedurze egzaminacyjnej ani nie była publikowana.

Oświadczam również, że egzemplarz pracy dyplomowej zatwierdzonej przez Promotora jest całkowicie zgodny z egzemplarzem pracy dyplomowej zamieszczonej w wersji elektronicznej w uczelnianym systemie elektronicznym.

Jednocześnie przyjmuję do wiadomości, że gdyby niniejsze oświadczenie okazało się niezgodne z prawdą stanowi to podstawę stwierdzenia nieważności postępowania w sprawie nadania tytułu zawodowego i unieważnienia dyplomu.

...............…….………..............................................

(czytelny podpis studenta/studentki oraz nr albumu)

1. Marek Serafin, *Wirtualizacja w praktyce*, Helion, Gliwice, 2012, str. 13, 59-62 [↑](#footnote-ref-1)
2. Arthur Mateos, Jothy Rosenberg, *Chmura obliczeniowa rozwiązania dla biznesu*, Helion, Gliwice, 2011, str. 26 [↑](#footnote-ref-2)
3. Oficjalna strona Microsoft, https://azure.microsoft.com/pl-pl/resources/cloud-computing-dictionary/what-is-devops, data dostępu: 06.12.2023 [↑](#footnote-ref-3)
4. Borko Fuhrt, Armando Escalante, Handbook of Cloud Computing, Springer, Nowy Jork, 2010, str. 7-22, 84-91 [↑](#footnote-ref-4)
5. Jeroen Mulder, *Multi-Cloud Architecture and Governance. Leverage Azure, AWS, GCP, and VMware vSphere to build effective multi-cloud solutions,* Packt Publishing, Birmingham, 2020, str. 4-8, 70-73 [↑](#footnote-ref-5)
6. Kief Morris, *Infrastructure as Code. Dynamic Systems for the Cloud Age*, O’Reilly, Sebastopol, 2021, str. 4 [↑](#footnote-ref-6)
7. Oficjalna strona Hashicorp, https://www.hashicorp.com/resources/the-story-of-hashicorp-terraform-with-mitchell-hashimoto, data dostępu: 05.12.2023 [↑](#footnote-ref-7)
8. Oficjalna strona OpenTofu, https://opentofu.org/manifesto/, data dostępu 05.12.2023 [↑](#footnote-ref-8)
9. Oficjalna strona Amazon Web Services, https://aws.amazon.com/s3/, data dostępu: 02.12.2023 [↑](#footnote-ref-9)
10. Oficjalna strona Microsoft Azure, https://azure.microsoft.com/pl-pl/resources/cloud-computing-dictionary/what-is-a-data-lake, data dostępu 02.12.2023 [↑](#footnote-ref-10)
11. Oficjalna strona dokumentacji Terraform, https://developer.hashicorp.com/terraform/language/state, data dostępu: 02.12.2023 [↑](#footnote-ref-11)
12. Oficjalna strona Amazon Web Services, https://aws.amazon.com/dynamodb/, data dostępu: 02.12.2023 [↑](#footnote-ref-12)
13. Oficjalna strona Amazon Web Services, https://docs.aws.amazon.com/vpc/latest/userguide/what-is-amazon-vpc.html, data dostępu: 02.12.2023 [↑](#footnote-ref-13)
14. Oficjalna strona AWS, https://aws.amazon.com/ec2/, data dostępu: 02.12.2023 [↑](#footnote-ref-14)
15. Oficjalna strona dokumentacji AWS, https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/Welcome.html, data dostępu: 02.12.2023 [↑](#footnote-ref-15)
16. Blog członka AWS Heroes Program (AWS Community Hero), https://www.terraform-best-practices.com/v/pl/key-concepts, data dostępu 06.12.2023 [↑](#footnote-ref-16)