

1.1.1 Wyższa Szkoła Informatyki Stosowanej i Zarządzania

1.1.3 pod auspicjami Polskiej Akademii Nauk

1.1.5 WYDZIAŁ INFORMATYKI

1.1.6 Kierunek INFORMATYKA

1.1.7 Studia I stopnia (dyplom inżyniera)

1.1.4

1.1.2



PRACA DYPLOMOWA

Norbert Herman

PROJEKT I IMPLEMENTACJA INFRASTRUKTURY CHMURY OBLICZENIOWEJ ZAPEWNIAJĄCEJ PRACĘ APLIKACJI W TRYBIE WYSOKIEJ DOSTĘPNOŚCI Z WYKORZYSTANIEM AMAZON WEB SERVICES

Promotor pracy:

dr inż. Jarosław Sikorski

WARSZAWA, rok akademicki 2017/2018

Autor: **Norbert Herman**

Tytuł: **PROJEKT I IMPLEMENTACJA INFRASTRUKTURY
CHMURY OBLICZENIOWEJ ZAPEWNIĄCEJ
PRACĘ APLIKACJI W TRYBIE WYSOKIEJ
DOSTĘPNOŚCI Z WYKORZYSTANIEM AMAZON
WEB SERVICES**

Wydział: **INFORMATYKI**

Kierunek: **INFORMATYKA**

Specjalność: **SIECI KOMPUTEROWE**

Studia: **I STOPNIA (DYPLOM INŻYNIERA)**

Dziekan Wydziału: **Dr inż. Jarosław Sikorski**

Promotor pracy: **Dr inż. Jarosław Sikorski**

Konsultant pracy: **Mgr inż. Paweł Pławiak**

Rok akademicki: **2017/2018**

OŚWIADCZENIE AUTORA PRACY DYPLOMOWEJ

Świadom(a) odpowiedzialności prawnej oświadczam, że niniejsza praca dyplomowa została napisana przeze mnie samodzielnie i nie zawiera treści uzyskanych w sposób niezgodny z obowiązującymi przepisami prawa, w szczególności z ustawą z dnia 4 lutego 1994 r. o prawie autorskim i prawach pokrewnych (Dz. U. z 1994 r. Nr 24, poz. 83 – tekst pierwotny i Dz. U. z 2000 r. Nr 80, poz. 904 – tekst jednolity, z późniejszymi zmianami).

Oświadczam, że wszystkie narzędzia informatyczne zastosowane do wykonania niniejszej pracy wykorzystałem(am) zgodnie z obowiązującymi przepisami prawa w zakresie ochrony własności intelektualnej i przemysłowej.

Oświadczam, że przedstawiona praca nie była wcześniej przedmiotem procedur związanych z uzyskaniem tytułu zawodowego w szkole wyższej.

Jednocześnie stwierdzam, że tekst pracy dyplomowej jest identyczny w obu składanych egzemplarzach i tożsamy z wersją elektroniczną przekazaną do UBI.

.....
Data

.....
Podpis autora pracy

Spis treści

Wstęp.....	9
Rozdział 1. Wprowadzenie do chmury obliczeniowej	11
1.1 Czym jest chmura obliczeniowa	11
1.2 Architektura chmury oraz jej cechy charakterystyczne.....	13
1.3 Modele wdrażania	15
1.4 Modele usług w chmurze.....	16
Rozdział 2. Chmura publiczna Amazon Web Services.....	20
2.1 Amazon Web Services.....	21
2.2 Oficjalne narzędzia wspomagające pracę w AWS.....	21
2.3 Narzędzia firm trzecich.....	24
Rozdział 3. Projekt oraz implementacja infrastruktury	25
3.1 Konta użytkowników w AWS oraz uprawnienia.....	26
3.2 Terraform – pierwsze kroki	28
3.3 Dane logowania oraz zmienne lokalne.....	29
3.4 Wirtualna chmura prywatna.....	30
3.4.1 Projekt sieci	32
3.4.2 Podsieci - publiczne i prywatne.....	33
3.4.3 Bramy domyślne oraz tablice routingu.....	34
3.5 Grupy Zabezpieczeń (z ang. Security Groups)	36
3.6 Serwer Bastion	37
Rozdział 4. Warstwa obliczeniowa	40
4.1 Baza danych	40
4.2 Sieciowy system plików	41
4.3 Równoważenie obciążenia.....	41
4.4 Ustawienia serwera aplikacyjnego	44
Rozdział 5. Dystrybucja aplikacji.....	48
5.1 System translacji nazw Internetowych – Route53	48
5.2 Certyfikat SSL.....	49
5.3 Globalna sieć dostarczania treści Amazon CloudFront.....	50

Zakończenie.....	52
Bibliografia.....	53
Spis ilustracji.....	54
Zawartość CD	55

Wstęp

Chmura obliczeniowa jest w ostatnich czasach jednym z najczęściej pojawiających się tematów w świecie informatyki. Pomimo wielu krytycznych głosów odnośnie zagrożeń jakie niesie za sobą wynoszenie znacznej części infrastruktury w ręce obcych korporacji ten model biznesowy ma wielu zwolenników, którzy na każdym kroku przekonują o jej wyższości w optymalizacji kosztów i wydajności. Co więcej, chmura pozwala na stosunkowo niewielki próg wejścia oraz możliwości znacznie łatwiejszego zarządzania i udostępniania produktów. Celem pracy jest zaprezentowanie w jaki sposób można uruchomić środowisko w chmurze Amazon Web Services z wykorzystaniem modelu infrastruktura jako usługa IaaS (z ang. Infrastructure as a Service), który pozwoli na wdrożenie aplikacji internetowej w trybie wysokiej dostępności.

Praca składa się z pięciu części, w rozdziale pierwszym opisano podstawowe informacje o chmurze obliczeniowej, zostało wyjaśnione pojęcie chmury obliczeniowej oraz wskazane jej cechy charakterystyczne. Zaprezentowane zostały różnice pomiędzy poszczególnymi modelami chmury wraz z przypadkami użycia, jak również wyjaśnione różnice pomiędzy modelami oferowanych usług w ramach infrastruktury jaką jest chmura. Rozdział drugi został w całości poświęcony środowisku Amazon Web Services. Zaprezentowano w nim popularne narzędzia wspomagające budowanie jak również zarządzanie środowiskiem w infrastrukturze Amazon Web Services.

Trzecia część pracy przedstawia sposoby, za pomocą których można zbudować oraz zarządzać infrastrukturą. Zaprezentowane zostały ogólnodostępne narzędzia, zarówno te przygotowane i udostępnione przez Amazon Web Services, jak również oprogramowanie stron trzecich. Rozdział ten opisuje scenariusz budowania od podstaw infrastruktury w modelu IaaS za pomocą narzędzia Terraform – uruchomiana została wirtualna sieć prywatna wraz z wszystkimi komponentami potrzebnymi do uruchomienia oraz zabezpieczenia wdrażanej aplikacji.

W rozdziale czwartym zaprezentowano sposoby wdrażania aplikacji webowej. W tym celu wykorzystana została popularna platforma blogowa WordPress. Aplikacja została zbudowana w oparciu o maszyny wirtualne Amazon EC2 pracujące po kontrolą systemu Amazon Linux, serwer Nginx oraz interpreter PHP. Serwery wykorzystują mechanizmy automatycznego skalowania pomiędzy którymi ruch jest stabilizowany za pomocą usługi równoważenia ruchu Amazon ELB. Do przechowywania i zarządzania danymi wykorzystane

zostały takie platformy środowiska Amazon Web Services jak sieciowy system plików Amazon EFS oraz baza danych Amazon Aurora.

Rozdział piąty jest prezentacją sposobu w jaki została zrealizowana dystrybucja aplikacji w Internecie. W tym celu wykorzystano wbudowane usługi chmury Amazon Web Services. Rozwiązywanie nazw odbędzie się przy pomocy systemu DNS (z ang. Domain Name System) Amazon Route53, a komunikacja z serwerami oraz dane statyczne będą udostępniane przy pomocy globalnego systemu dostarczania treści CDN (z ang. Content Delivery Network) Amazon CloudFront.

Rozdział 1. Wprowadzenie do chmury obliczeniowej

Technologie takie jak Internet, do którego dostęp jest możliwy niemalże z każdego miejsca na Ziemi, wirtualizacja, przetwarzanie sieciowe czy architektura zorientowana na usługi¹ przyczyniły się do powstania chmury obliczeniowej – obecnie jednej z najbardziej obiecujących technologii. Przetwarzanie w chmurze umożliwia zmianę tradycyjnego sposobu postrzegania i wykorzystywania zasobów obliczeniowych, dzięki czemu możemy całkowicie przedefiniować wzorce w sposobie zakupu oraz przydzielania zasobów związanych z infrastrukturą (procesor, pamięć operacyjna, pamięć masowa, przepustowość sieci), oprogramowaniem, warstwą aplikacji itp. Przetwarzanie w chmurze umożliwia dostarczanie tych zasobów w formie usługi na żądanie, zazwyczaj przez Internet, ponieważ dostęp do tychże zasobów może być zdalny, a same usługi mogą być geograficznie niezależne.

1.1 Czym jest chmura obliczeniowa

Nie da się jednoznacznie scharakteryzować czym tak naprawdę jest chmura obliczeniowa oraz jakie cechy charakterystyczne są do niej przypisane. W pracy [3] została podana taka definicja chmury obliczeniowej:

„Chmura jest dużą pulą ogólnodostępnych i łatwych w użyciu zasobów wirtualnych (takich jak sprzęt, platformy, usługi). Zasoby te mogą być dynamicznie rekonfigurowane w celu dostosowania do zmiennego obciążenia, co pozwala na optymalne wykorzystywanie zasobów. Ta pula jest zazwyczaj wykorzystywana w usługowym modelu płatności, w którym szczegóły oferty są gwarantowane w formie umowy o świadczenie usług SLA (z ang. Service Level Agreement).”²

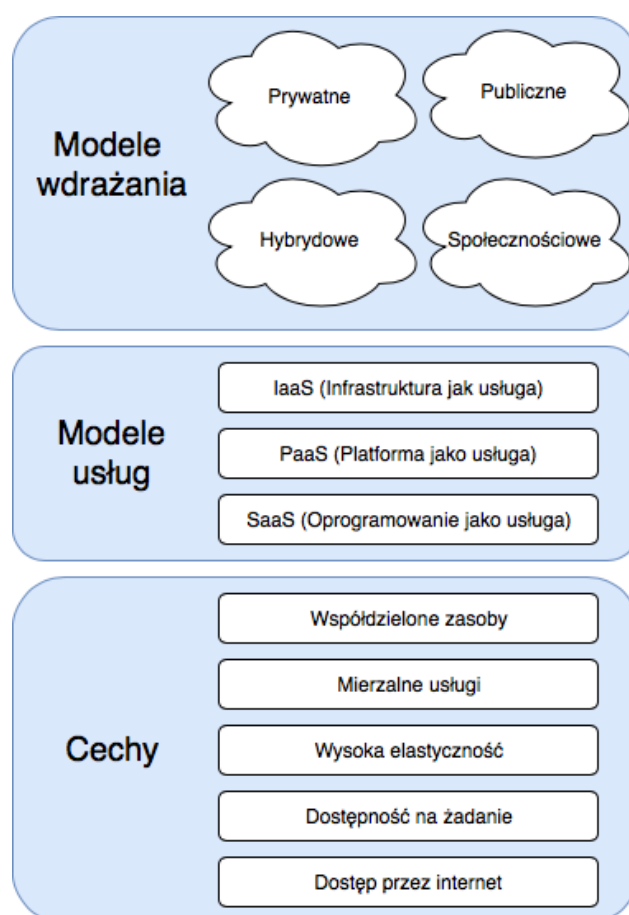
Natomiast Narodowy Instytut Standaryzacji i Technologii NIST (z ang. National Institute of Standards and Technology) [4] na potrzeby amerykańskich instytucji rządowych opracował następującą definicję przetwarzania w chmurze:

¹ Architektura zorientowana na usługi SOA (z ang. service-oriented architecture) – jest to koncepcja, w której główny nacisk stawia się na usługi spełniające wymagania użytkownika. [2]

² Tłumaczenie własne. Treść oryginalna: Clouds are a large pool of easily usable and accessible virtualized resources (such as hardware, development platforms and/or services). These resources can be dynamically reconfigured to adjust to a variable load (scale), allowing also for an optimum resource utilization. This pool of resources is typically exploited by a pay- per-use model in which guarantees are offered by the Infrastructure Provider by means of customized SLAs.

„Przetwarzanie w chmurze to model umożliwiający wszechobecny, wygodny dostęp na żądanie do współdzielonej puli konfigurowalnych zasobów (np. sieci, serwerów, pamięci masowych, aplikacji i usług), które mogą być szybko udostępnione przy minimalnym wysiłku włożonym w zarządzanie lub interakcję dostawcy usług. Ten model chmury składa się z pięciu podstawowych cech, trzech modeli usług oraz czterech modeli wdrażania.”³

Definicja stworzona przez NIST⁴ dodatkowo wprowadza podział na rodzaje chmur obliczeniowych oraz określa cechy jakie powinien posiadać system chmurowy. Podział ten został zaprezentowany na Rysunku 1.1 i zostanie opisany w kolejnych podrozdziałach.



Rys. 1.1 Modele i cechy przetwarzania w chmurze wg NIST⁵

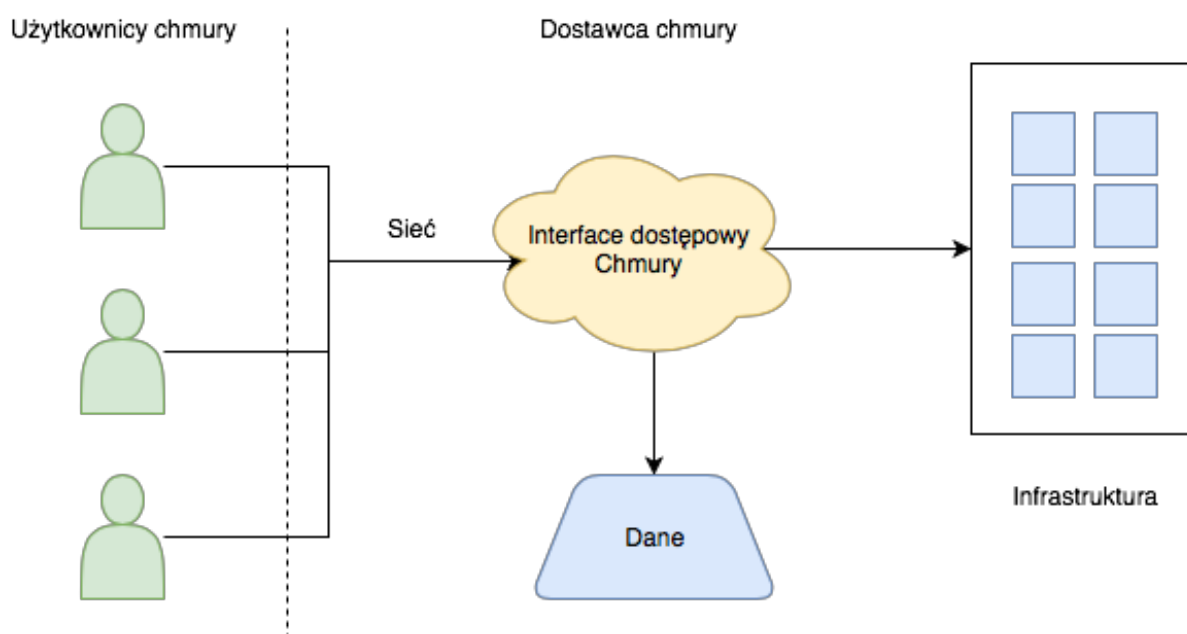
³ Tłumaczenie własne. Treść oryginalna: Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction. This cloud model is composed of five essential characteristics, three service models, and four deployment models.

⁴ National Institute of Standards and Technology – Narodowy Instytut Standardów i Technologii

⁵ Opracowanie własne

1.2 Architektura chmury oraz jej cechy charakterystyczne

Architektura chmury składa się z dwóch podstawowych składników: dostawcy oraz użytkownika. Jako dostawcę chmury rozumiemy podmiot, który zajmuje się obsługą usług w chmurze, zaś użytkownik to jednostka, która tych usług żąda. Użytkownik uzyskuje dostęp poprzez takie urządzenia jak komputer osobisty, tablet czy też telefon komórkowy przy wykorzystaniu sieci lokalnej lub Internet. Dostawca chmury udostępnia interfejs⁶ dostępowy (np. strona WWW⁷ lub API⁸), przez który użytkownik na żądanie może uzyskać dostęp do zasobów takich jak infrastruktura, platforma czy oprogramowanie. Interfejs dostępowy jest częścią oprogramowania kontrolującego zarządzanie dostarczaniem zasobów.



Rys. 1.2 Architektura chmury⁹

Tak jak już wcześniej zostało wspomniane NIST¹⁰ w swojej definicji chmury obliczeniowej wymienia charakterystyczne cechy jakie powinna mieć chmura. Należą do nich:

⁶ Interfejs (ang. UI – User Interface) – przestrzeń zapewniająca interakcję użytkownika z systemem.

⁷ WWW (ang. World Wide Web) – Światowa Rozległa Sieć Internetowa

⁸ API (ang. application programming interface) – Interfejs programowania aplikacji, jest zestaw reguł służący komunikacji pomiędzy programami komputerowymi.

⁹ Opracowanie własne

¹⁰ National Institute of Standards and Technology – Narodowy Instytut Standardów i Technologii

- **Współdzielone zasoby** – na żądanie klientów oraz zgodnie z ich zapotrzebowaniem przydzielane są usługi ze wspólnej puli zasobów. Dodatkowo wielu klientom można przydzielić zasoby z tego samego serwera fizycznego (tak zwane multi tenancy¹¹), takie działanie przekłada się na optymalne zarządzanie i wykorzystywanie zasobów fizycznych. Klient nie ma zazwyczaj kontroli ani wiedzy o dokładnej lokalizacji dostarczanych zasobów, ale powinien być w stanie wybrać położenie na wyższym poziomie abstrakcji takim jak np. kraj.
- **Mierzalne usługi** – system zarządzający chmurą (tak zwany cloudOS) jest w stanie automatycznie optymalizować i kontrolować wykorzystanie zasobów mierząc wykorzystanie takich usług jak np. przepustowość sieci, obciążenie CPU¹² czy ilość danych. Zużycie zasobów jest monitorowane, kontrolowane i zgłaszane w możliwie przejrzysty sposób zarówno dla dostawcy jak i konsumenta.
- **Wysoka elastyczność** – jest to najbardziej doceniana cecha chmury, gdyż umożliwia użytkownikom dynamiczne przydzielanie i zwalnianie zasobów. Konsumentom nie muszą planować zapotrzebowania na usługi w przyszłości, gdyż chmura umożliwia skalowanie ilości wykorzystywanych zasobów zgodnie z aktualnym zapotrzebowaniem.
- **Dostępność na żądanie** – konsument może sam ustalić oraz zapewnić ilość potrzebnych zasobów bez konieczności kontaktu z dostawcą.
- **Dostęp przez Internet** – konsument ma możliwość zarządzania usługami globalnie wykorzystując sieć Internet oraz popularne urządzenia klienckie (jak np. laptop, tablet, telefon)

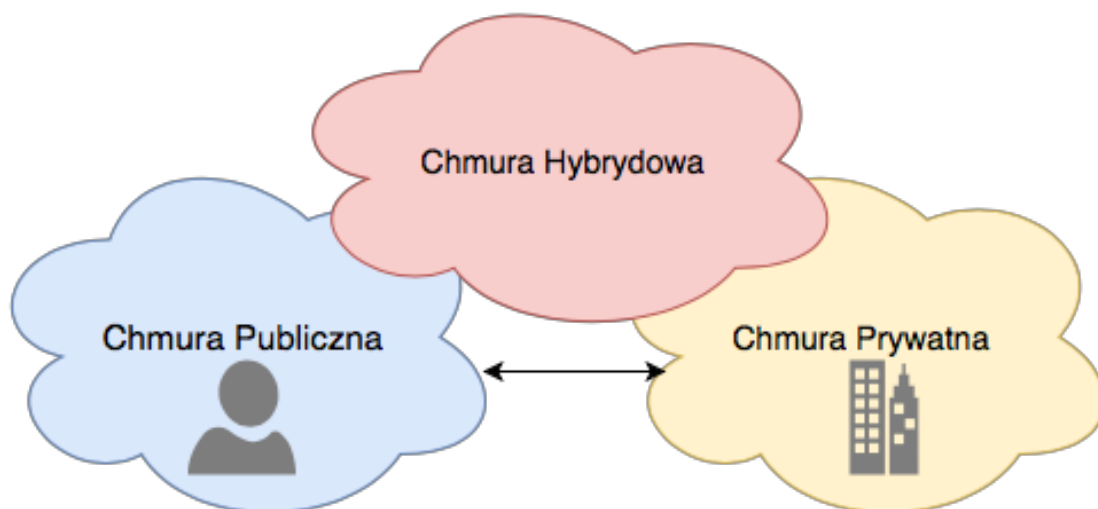
¹¹ Multi tenancy – korzystanie przez wielu użytkowników z tych samych zasobów [1]

¹² CPU (ang. Central processing unit) – procesor, czyli cyfrowe urządzenie odpowiedzialne za pobieranie, interpretację i wykonywanie rozkazów.

1.3 Modele wdrażania

Istnieją różne modele biznesowe, a co za tym idzie konsumenci mają inne potrzeby i wymagania. Dodatkowo czynniki takie jak zasoby robocze, wrażliwość danych czy po prostu typ prowadzonego biznesu wymagają indywidualnego podejścia, w związku z czym istnieją różne modele wdrażania chmury. Najpopularniejsze z nich to:

- **Chmura prywatna** – Rodzaj infrastruktury udostępniany i używany wyłącznie w ramach jednej organizacji obejmującej wielu konsumentów (np. jednostki organizacyjne). Zasoby fizyczne mogą być utrzymywane zarówno wewnątrz jak i na zewnątrz organizacji oraz mogą, ale nie muszą być własnością organizacji. Chmura prywatna jako model biznesowy umożliwia wykorzystywanie wewnętrznych procedur bezpieczeństwa, a tym samym może zapewnić znacznie wyższy poziom zabezpieczeń danych niż ma to miejsce w chmurze publicznej. Dodatkowym atutem jest fakt, iż sprzęt jest kontrolowany i wykorzystywany wyłącznie przez jedną organizację co bezpośrednio przekłada się na zwiększeniem kontroli nad wydajnością.
- **Chmura społecznościowa** - Infrastruktura budową przypominająca chmurę prywatną, jednak różniącą się tym, iż jest udostępniona do wyłącznego użytku przez kilka organizacji posiadających wspólne cechy. Może być zarządzana zarówno przez jedną jak i kilka organizacji w ramach społeczności. Przykładem takiego modelu może być chmura przygotowana i wykorzystywana przez instytucje rządowe.
- **Chmura publiczna** – Typ infrastruktury znajdujący się na terenie dostawcy chmury w którym usługi są udostępniane do powszechnego użytku ogółu społeczeństwa. W odróżnieniu od chmury prywatnej za bezpieczeństwo i prywatność odpowiedzialny jest dostawca usług, ponieważ użytkownik nie ma kontroli nad tym jak i gdzie dane są przechowywane. W związku z tym pomiędzy klientem a dostawcą istnieje możliwość podpisania umowy o poziomie usług SLA (z ang. Service Level Agreement) w którym określony jest poziom jakości usług QoS (z ang. Quality of Service). W przypadku, gdy usługodawca nie wywiąże się z umowy o SLA zazwyczaj płaci karę użytkownikowi. Najbardziej znani dostawcy chmury publicznej to Amazon AWS, Microsoft Azure oraz Google Cloud Platform.



Rys. 1.3 Modele wdrażania chmury¹³

- **Chmura hybrydowa** – Infrastruktura, która jest kompozycją co najmniej dwóch odrębnych infrastruktur chmurowych (prywatnej, społecznościowej lub publicznej) z zachowaniem ich unikalności jednak powiązanych ze sobą w celu umożliwienia bezpiecznego przenoszenia danych i aplikacji. Ten typ infrastruktury pozwala zachować zalety wynikające z zarządzania bezpieczeństwem w chmurze prywatnej jednocześnie zapewniając niemalże nieograniczone zasoby obliczeniowe, przez co często jest wykorzystywany w kontekście tak zwanego rozerwania chmury. Jest to sytuacja, w której określona aplikacja, działająca w zakresie chmury prywatnej potrzebuje dodatkowych zasobów (np. moc obliczeniowa czy miejsce na dane), których prywatna infrastruktura nie może dostarczyć. Następuje wtedy "rozerwanie" chmury (z ang. cloud bursting) w celu dostarczenia wymaganych zasobów z puli w chmurze publicznej.

1.4 Modele usług w chmurze

Wymienione wyżej modele wdrażania mogą zostać wykorzystane do świadczenia różnych usług w zależności od wymagań i poziomu wiedzy klienta. Rysunek 1.3 zestawia modele usług w chmurze, które zostały szczegółowo opisane poniżej. Obok trzech najpopularniejszych warstw została zaprezentowana również kolokacja, która co prawda nie jest modelem chmury, ale może być wykorzystywany przy wdrożeniach chmury prywatnej.

¹³ Opracowanie własne

- **Kolokacja** to najprościej mówiąc wynajęcie przestrzeni w serwerowni. Klient musi sam dostarczyć niezbędny hardware oraz oprogramowanie. Rola dostawcy w tym modelu ogranicza się do udostępnienia miejsca w serwerowni, prądu oraz dostępu do Internetu. W kolokacji płatności pobierane są zazwyczaj w postaci miesięcznego abonamentu a wysokość opłat zależna jest od ilości zadeklarowanego zużycia. Przykładem firmy udostępniającej miejsce w serwerowni jest firma Ataman¹⁴. Kolokacja jest idealnym rozwiązaniem dla organizacji, które chcą zbudować własną chmurę prywatną, ale nie mogą dostarczyć odpowiedniego zaplecza w postaci własnej serwerowni.
- **Infrastruktura jako usługa, IaaS** (z ang. Infrastructure as a Service) to usługa, w której klient jest odpowiedzialny za swoje systemy zaczynając od warstwy systemu operacyjnego. Do niedawna najpopularniejszą usługą typu IaaS był serwer dedykowany, czyli najprościej mówiąc oddzielny komputer, na którym użytkownik może zarządzać oprogramowaniem (w tym systemem operacyjnym), ale nie ma dostępu do podzespołów fizycznych. Aktualnie jednak dużo popularniejszym rozwiązaniem jest wynajmowanie przestrzeni oraz udostępnianie maszyn wirtualnych. W tym modelu użytkownik może decydować o parametrach maszyny wirtualnej (CPU¹⁵, RAM¹⁶, magazyn danych, sieć). Dostęp do maszyny najczęściej jest zapewniany przez protokoły SSH¹⁷, VNC¹⁸ oraz RDP¹⁹. Dostawca zobowiązuje się do zapewnienia pełnej infrastruktury, czyli np. prądu, klimatyzacji, sieci, ale również maszyn fizycznych oraz specjalnego oprogramowania do zarządzania maszynami wirtualnymi. Model płatności w tego typu usługach jest zazwyczaj bardzo prosty i opiera się na naliczaniu opłat za czas działania (obecnie najczęściej jest to naliczanie minutowe), a na wysokość opłat największy wpływ ma ilość zadeklarowanych zasobów oraz rodzaj oprogramowania.

¹⁴ <https://www.atman.pl/Kolokacja-ccms-pol-90.html> Dostęp: 25.05.2018r.

¹⁵ CPU (ang. Central processing unit) – procesor, czyli cyfrowe urządzenie odpowiedzialne za pobieranie, interpretację i wykonywanie rozkazów.

¹⁶ RAM (ang. Random-access memory) – rodzaj pamięci cyfrowej, umożliwiającą wielokrotny, łatwy i szybki zapis oraz odczyt.

¹⁷ Secure Shell (SSH) – szyfrowany protokół komunikacyjny działający w modelu klient-serwer.

¹⁸ Virtual Network Computing (VNC) – system zdalnego przekazywania obrazu.

¹⁹ Remote Desktop Protocol (RDP) – protokół zdalnej komunikacji ze terminalem graficznym działający w środowisku MS Windows

Najpopularniejsze usługi typu IaaS to AWS EC2²⁰, Azure Compute²¹, Google Compute Engine²².

- **Platforma jak usługa, PaaS** (z ang. Platform as a Service) jest to model, w którym użytkownik może bardzo szybko i łatwo skonfigurować oraz uruchomić swój produkt w chmurze. PaaS to usługa oferująca środowisko wykonawcze wraz z zestawem narzędzi wspierających wdrażanie. W odróżnieniu od IaaS klient nie musi mieć wiedzy o specyfikacji technicznej w której uruchamiana będzie aplikacja. Jest to model, który ma po prostu działać i jest idealnym rozwiązaniem dla deweloperów, gdyż pomaga skupić się na np. kodzie aplikacji i wgrywaniu go, ale nie wymaga zaawansowanej wiedzy z zakresu administracji systemami oraz bezpieczeństwa. System płatności w PaaS jest znacznie bardziej skomplikowany niż w IaaS, gdyż jest zależny od rodzaju aplikacji, wykorzystanych zasobów (np. mocy obliczeniowej czy ruchu sieciowego), co jest bardzo korzystne w przypadku aplikacji, które dopiero się rozwijają, gdyż pozwala na wygodne zarządzanie kosztami. Przykładowe serwisy to AWS Elastic Beanstalk²³, AWS Lambda²⁴, Google App Engine²⁵.
- **Oprogramowanie jako usługa, SaaS** (z ang. Software as a Service) to najpopularniejszy model usługi w chmurze publicznej, który nie wymaga od użytkownika wiedzy z zakresu budowania aplikacji. Pozwala na bardzo szybkie uruchomienie i korzystanie z aplikacji, bez konieczności instalacji na komputerze lokalnym. Zwalnia się w ten sposób klienta z kupna, instalacji oraz dbania o regularne aktualizowanie aplikacji. Model ten w dużej mierze chroni dostawcę przed nielegalnym wykorzystaniem, gdyż jedyny sposób użytkownika programu to połączenie się z usługą przez Internet. Kolejną ogromną zaletą SaaS są stosunkowo niewielkie wymagania sprzętowe oraz możliwość uruchamiania na wielu platformach bez konieczności przygotowywania oddzielnych wersji a różne systemy operacyjne, ponieważ aplikacje w modelu SaaS zazwyczaj działają w przeglądarce internetowej. Model płatności w tym wypadku jest bardzo prosty i ogranicza się do abonamentu za usługę, z tym, że zazwyczaj trzeba z góry określić potrzebny pakiet (np. w przypadku dysku

²⁰ <https://aws.amazon.com/ec2/> Dostęp: 25.05.2018r.

²¹ <https://azure.microsoft.com/pl-pl/product-categories/compute/> Dostęp: 25.05.2018r.

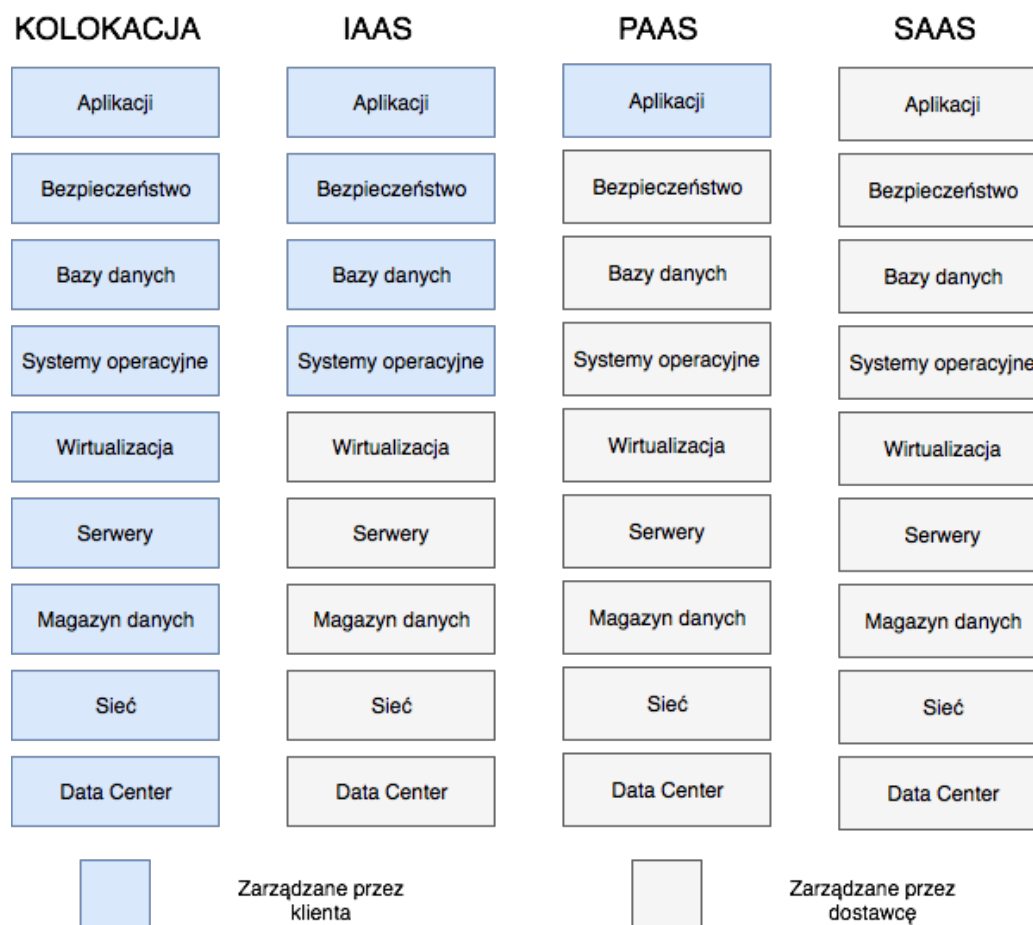
²² <https://cloud.google.com/compute/> Dostęp: 25.05.2018r.

²³ <https://aws.amazon.com/elasticbeanstalk/> Dostęp: 25.05.2018r.

²⁴ <https://aws.amazon.com/lambda/> Dostęp: 25.05.2018r.

²⁵ <https://cloud.google.com/appengine/> Dostęp: 25.05.2018r.

sieciowego z góry się płaci za przyznaną przestrzeń niezależnie od tego czy będzie wykorzystana). Na szczęście wiele aplikacji w trybie SaaS jest darmowa ewentualnie posiada darmowy plan, dzięki czemu zawsze można przetestować usługi. Do najbardziej popularnych usług w modelu SaaS zaliczają się takie serwisy jak: Google GMail²⁶, Dropbox²⁷ czy Microsoft Office365²⁸.



Rys. 1.4 Porównanie modeli chmury²⁹

²⁶ <https://www.google.com/gmail/> Dostęp: 25.05.2018r.

²⁷ <https://dropbox.com/> Dostęp: 25.05.2018r.

²⁸ <https://products.office.com/pl-PL/?ms.url=office365com> Dostęp: 25.05.2018r.

²⁹ Opracowanie własne na podstawie: <https://mycloudblog7.wordpress.com/2013/06/19/who-manages-cloud-iaas-paas-and-saas-services/>

Rozdział 2. Chmura publiczna Amazon Web Services

Na globalnym rynku dostawców technologii Informatycznych istnieje wielu dostawców chmury publicznej (Rys. 2.1), jednak niekwestionowanym liderem jest Amazon z pakietem usług w ramach Amazon Web Services, popularnie zwanym AWS. W związku z tym w pracy zostanie zaprezentowany właśnie ten dostawca jak również sposoby wdrażania oraz zarządzania modelem IaaS³⁰ w infrastrukturze AWS z wykorzystaniem ogólnodostępnych narzędzi ułatwiających pracę administratora systemów informatycznych.



Rys. 2.1 Światowi dostawcy IaaS [7]

³⁰ Infrastructure as a Service – Infrastruktura jako usługa

2.1 Amazon Web Services

W 2006 roku Amazon oficjalnie zaprezentował Amazon Web Services (AWS), czyli działalność związana z oferowaniem usług infrastruktury informatycznej dostępnych poprzez protokół HTTP³¹. AWS oferuje bardzo bogatą ofertę, dzięki czemu klient może dobrać odpowiedni pakiet usług i skonfigurować go pod siebie oraz swoje wymagania. Z usług AWS korzystają takie przedsiębiorstwa jak Netflix³², Spotify³³ czy Airbnb³⁴.

Podstawową usługą w ofercie AWS jest Amazon EC2³⁵, za pomocą której można tworzyć oraz uruchamiać maszyny wirtualne. Należy również zwrócić uwagę na nazewnictwo usług, w szczególności na to, iż wiele z nich posiada przedrostek Elastic czyli elastyczny. Nie jest to przypadkowe, gdyż usługi obliczeniowe oferowane przez AWS mają być jak najlepiej dostosowane do potrzeb użytkownika. Dodatkowo niezwykle ważnym aspektem jest fakt, że AWS posiada swoje centra danych już na każdym kontynencie, a dodatkowo w Stanach Zjednoczonych istnieje oddzielny region przeznaczony dla Rządu (GovCloud). Co więcej, każde takie centrum danych jest podzielone na strefy dostępności (AZ – z ang. Availability Zone). Taki podział gwarantuje, że przy odpowiednio zaplanowanym wdrożeniu aplikacji, tj. z wykorzystaniem podziału na regiony i strefy, aplikacja będzie zawsze dostępna dla użytkowników. [12]

2.2 Oficjalne narzędzia wspomagające pracę w AWS

Pracując z infrastrukturą komputerową, każdy administrator potrzebuje narzędzi, za pomocą których będzie mógł wdrażać, monitorować oraz zarządzać środowiskiem informatycznym. W przypadku administrowania chmurą obliczeniową jest dokładnie tak samo, co więcej, większość narzędzi jest ta sama (jak na przykład SSH³⁶, VNC³⁷, RDP³⁸). Poniżej zostały zaprezentowane popularne narzędzia wykorzystywane przy pracy ze środowiskiem AWS.

³¹ http – Hypertext Transfer Protocol

³² <https://netflix.com> Dostęp: 26.05.2018r.

³³ <https://spotify.com> Dostęp: 26.05.2018r.

³⁴ <https://airbnb.pl> Dostęp: 26.05.2018r.

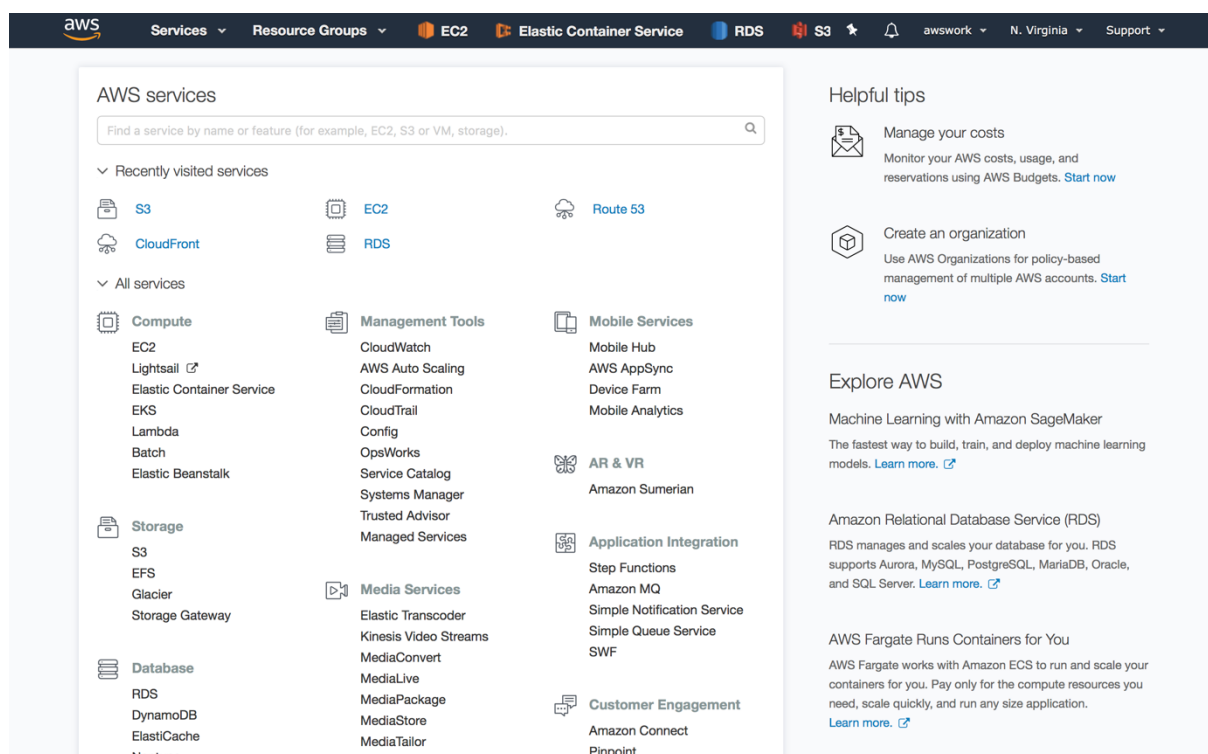
³⁵ Amazon EC2 – Amazon Elastic Compute Cloud

³⁶ Secure Shell (SSH) – szyfrowany protokół komunikacyjny działający w modelu klient-serwer.

³⁷ Virtual Network Computing (VNC) – system zdalnego przekazywania obrazu.

³⁸ Remote Desktop Protocol (RDP) – protokół zdalnej komunikacji ze terminalem graficznym działający w środowisku MS Windows

AWS Management Console – to interfejs graficzny umożliwiający dostęp do usług oferowanych przez AWS oraz zarządzanie nimi. Z poziomu tej konsoli możemy wykonać niemalże wszystkie czynności niezbędne do uruchomienia i zarządzania oferowanymi funkcjami. Jest to idealne narzędzie, dla osób które rozpoczynają swoją przygodę w tak rozbudowanym środowisku, gdyż nie wymaga pełnej znajomości usług.



Rys. 2.2 AWS Management Console

AWS Command Line Interface (CLI), interfejs z wierszem poleceń podobnie jak Management Console jest również oficjalnym narzędziem wydanym przez Amazon służącym do zarządzania usługami AWS. Jest to narzędzie wymagające przynajmniej podstawowej znajomości infrastruktury i metod zarządzania, przez co skierowany jest raczej do administratorów i deweloperów niż do zwykłych użytkowników. Dzięki AWS CLI można uruchamiać oraz konfigurować wszystkie dostępne usługi z poziomu wiersza poleceń, jak również można wykorzystać je przy automatyzacji zadań za pomocą skryptów.

AWS System Manager – jest to interfejs pozwalający na centralizację danych operacyjnych oraz automatyzację zadań w ramach zasobów AWS, takich jak na przykład EC2³⁹, S3⁴⁰

³⁹ Amazon EC2 (Elastic Compute Cloud) – usługa maszyn wirtualnych

⁴⁰ Amazon S3 (Simple Storage Service) – usługa przechowywania danych

czy RDS⁴¹. System Manager w znaczący sposób upraszcza pracę administratora systemów poprzez centralizację zarządzania zasobami aplikacjami, automatyzację zadań w ramach usług AWS oraz skraca czas wykrywania i rozwiązywania problemów.

AWS AutoScaling – narzędzie monitorujące wykorzystanie zasobów przez aplikacje oraz dostosowujące ich ilość w taki sposób, aby utrzymać stałą wydajność przy jednocześnie możliwie najniższym koszcie.

AWS CloudWatch – jest to usługa monitorująca zasoby oraz aplikacje w chmurze AWS. Za jej pomocą można gromadzić dane o wykorzystaniu zasobów, kolekcjonować logi aplikacji oraz automatycznie reagować na zmiany bądź niepożądane zdarzenia.

AWS CloudFormation – jest to usługa umożliwiająca tworzenie, wdrażanie i zarządzanie wszystkimi usługami w ramach infrastruktury AWS w postaci kodu (IaC)⁴².

Amazon OpsWorks - usługa do zarządzania konfiguracją, która zapewnia instancje z zainstalowanym i skonfigurowanym oprogramowaniem Chef⁴³ lub Puppet⁴⁴. Są to platformy służące do automatyzacji wdrażania konfiguracji na serwerach.

AWS Identity and Access Management (IAM) – usługa służąca do centralnego zarządzania uprawnieniami w AWS. Za jej pomocą można tworzyć użytkowników, przypisywać ich do grup oraz zarządzać poziomem uprawnień. Dodatkowo za pomocą IAM, można tworzyć role i przypisywać je do konkretnych usług, przykładowo tworząc rolę z uprawnieniami do zapisu logów w usłudze CloudWatch oraz przypisując ją do maszyny wirtualnej nie będzie potrzeby umieszczania na serwerze danych logowania, co znacznie zmniejsza ryzyko utraty danych logowania.

⁴¹ Amazon RDS (Relational Database Service) – usługa relacyjnych baz danych

⁴² IaC (z ang. Infrastructure as Code) – Infrastruktura jako kod

⁴³ <https://www.chef.io/chef/> Dostęp: 28.05.2018r.

⁴⁴ <https://puppet.com/> Dostęp 28.05.2018r.

2.3 Narzędzia firm trzecich

Terraform – podobnie do AWS CloudFormation jest narzędzie do tworzenia i zarządzania infrastrukturą w postaci kodu (IaC)⁴⁵. Program jest wydawany na zasadach Open Source przez firmę HashiCorp. Nie jest to oficjalne narzędzie do obsługi chmury AWS, pozwala użytkownikowi na jednoczesną orkiestrację zadań od wielu dostawców usług w chmurze (np. zaplanować wdrożenie środowiska chmury hybrydowej składającego się z lokalnej chmury prywatnej pracującej pod kontrolą systemu OpenStack⁴⁶ oraz publicznej chmury np. AWS). [11]

Ansible – oprogramowanie służące do zarządzania konfiguracją na serwerach. Jest to alternatywa dla takich narzędzi jak Chef, Puppet czy AWS System Manager, ale różni się tym, że nie potrzebuje zainstalowanego agenta na serwerze tylko używa ogólnodostępnych metod logowania na serwerze np. SSH⁴⁷. Istotną częścią Ansible są tak zwane playbooks, czyli pliki tekstowe ze scenariuszem wykonywania działań na serwerze. [8]

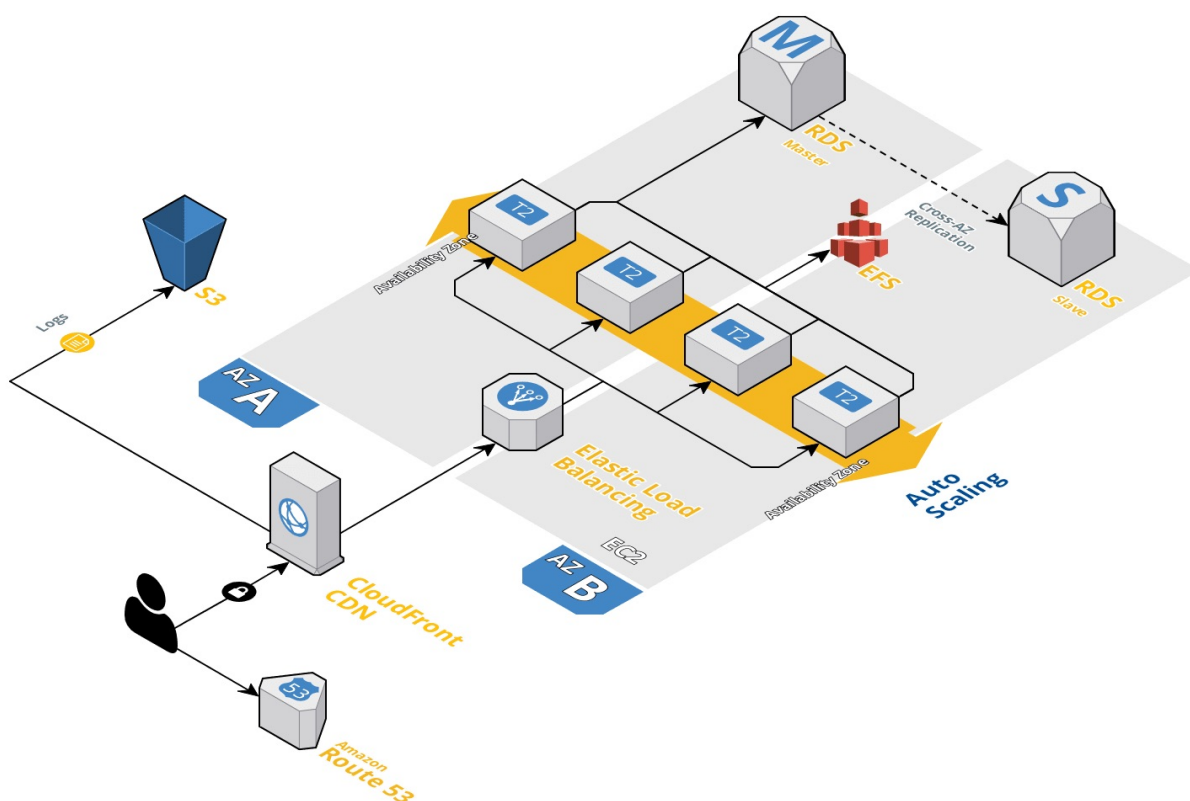
⁴⁵ IaC (z ang. Infrastructure as Code) – Infrastruktura jako kod

⁴⁶ OpenStack – oprogramowanie z dziedziny chmur obliczeniowych rozwijane przez Rackspace Cloud.
<https://www.openstack.org/>

⁴⁷ Secure Shell (SSH) – szyfrowany protokół komunikacyjny działający w modelu klient-serwer.

Rozdział 3. Projekt oraz implementacja infrastruktury

W poprzednim rozdziale zostały zaprezentowane narzędzia za pomocą których można budować i zarządzać infrastrukturą w chmurze publicznej AWS. W tej części pracy zostanie pokazane jak za pomocą narzędzia Terraform zbudować pełną infrastrukturę pod aplikację webową, w tym wypadku będzie to wdrożenie aplikacji WordPress w trybie wysokiej dostępności (HA – z ang. High Availability) z wykorzystaniem automatycznego skalowania, bazy danych Amazon Aurora MySQL oraz sieciowego systemu plików Amazon EFS. Dodatkowo zostanie skonfigurowany system DNS Amazon Route53 oraz system globalnego dostarczania treści Amazon CloudFront aby skrócić czasy odpowiedzi do użytkownika końcowego. Zastosowane także zostaną elementy zwiększające bezpieczeństwo aplikacji takie jak grupy zabezpieczeń, podsieci wraz z izolacją elementów infrastruktury oraz serwer służący do komunikacji wewnątrz sieci – tak zwany serwer bastion. Architektura aplikacji została zaprezentowana na Rysunku 3.1



Rys. 3.1 Architektura aplikacji

3.1 Konta użytkowników w AWS oraz uprawnienia

Pierwszy krok jaki należy uczynić to utworzyć dedykowanego użytkownika oraz nadać uprawnienia do usług, które będą wykorzystywane, po czym przypisać tylko programistyczny dostęp, czyli za pomocą identyfikatora dostępu (z ang. access key ID) oraz klucza dostępu (z ang. secret access key). Jest to para kluczy wykorzystywana do komunikacji z AWS poprzez AWS CLI⁴⁸ oraz API⁴⁹. Użytkownik taki nie ma dostępu do interfejsu menadżera poprzez przeglądarkę, a przez CLI lub API może wykonywać wszystkie czynności, do których wcześniej został przypisany dostęp. Można skorzystać z gotowych zestawów polityk lub utworzyć własne. W tym celu należy napisać własny dokument w formacie JSON⁵⁰ zawierający pola:

- “Effect” – określa czy polityka dopuszcza lub odrzuca dostęp
- “Action” – lista działań dopuszczonych lub odrzuconych
- “Resource” – lista zasobów, których dotyczy polityka
- “Condition” (Opcjonalnie) – okoliczności, w których polityka ma być zastosowana

Przykładowa polityka dla uprawnień administratora o pełnym dostępie do wszystkich zasobów wygląda następująco:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "*",
      "Resource": "*"
    }
  ]
}
```

Rysunki 3.2 oraz 3.3 przedstawiają zrzuty konsoli AWS z dodawania nowego użytkownika w konsoli AWS.

⁴⁸ CLI (Command Line Interface) – interfejs lini poleceń

⁴⁹ API (Application Programming Interface) – interfejs programistyczny aplikacji

⁵⁰ JSON - JavaScript Object Notation

Add user

1

2

3

4

Set user details

You can add multiple users at once with the same access type and permissions. [Learn more](#)

User name*

[+ Add another user](#)

Select AWS access type

Select how these users will access AWS. Access keys and autogenerated passwords are provided in the last step. [Learn more](#)

- Access type* ☒ **Programmatic access**
Enables an **access key ID** and **secret access key** for the AWS API, CLI, SDK, and other development tools.
- ☐ **AWS Management Console access**
Enables a **password** that allows users to sign-in to the AWS Management Console.

* Required


[Cancel](#)


[Next: Permissions](#)

Rys. 3.2 Dodawanie użytkownika IAM

Set permissions for teraform

 Add user to group











 Copy permissions from existing user

 Attach existing policies directly

Attach one or more existing policies directly to the users or create a new policy. [Learn more](#)

[Create policy](#)

[Refresh](#)

Filter: Policy type ▾		Q Search		Showing 348 results	
	Policy name ▾	Type	Attachments ▾	Description	
<input type="checkbox"/>	 AdministratorAccess	Job function	1	Provides full access to AWS services and resources.	
<input type="checkbox"/>	 AlexaForBusinessDe...	AWS managed	0	Provide device setup access to AlexaForBusiness s...	
<input type="checkbox"/>	 AlexaForBusinessFul...	AWS managed	0	Grants full access to AlexaForBusiness resources a...	
<input type="checkbox"/>	 AlexaForBusinessGa...	AWS managed	0	Provide gateway execution access to AlexaForBusin...	
<input type="checkbox"/>	 AlexaForBusinessRe...	AWS managed	0	Provide read only access to AlexaForBusiness servi...	
<input type="checkbox"/>	 AmazonAPIGateway...	AWS managed	0	Provides full access to create/edit/delete APIs in Am...	
<input type="checkbox"/>	 AmazonAPIGatewayAdmin...	AWS managed	0	Provides full access to invoke APIs in Amazon API G...	
<input type="checkbox"/>	 AmazonAPIGateway...	AWS managed	0	Allows API Gateway to push logs to user's account.	
<input type="checkbox"/>	 AmazonAppStreamF...	AWS managed	0	Provides full access to Amazon AppStream via the A...	
<input type="checkbox"/>	 AmazonAppStreamR...	AWS managed	0	Provides read only access to Amazon AppStream vi...	

Rys. 3.3 Przypisywanie uprawnień użytkownikowi IAM

3.2 Terraform – pierwsze kroki

Pracę z Terraform należy zacząć od instalacji aplikacji na lokalnym komputerze, wybierając odpowiednią wersję dla swojego środowiska⁵¹. Warto zaznaczyć, że jest to aplikacja wieloplatformowa⁵², także zarówno osoby korzystające z wszelkiego rodzaju systemów uniksopodobnych⁵³ (Linux, BSD, MacOS) jak również użytkownicy systemów z rodziny Microsoft Windows, mogą bezproblemowo korzystać z Terraform. Po instalacji i upewnieniu się, że aplikacja poprawnie dodała się do systemu, można zacząć pierwszy projekt.

Terraform korzysta z plików z rozszerzeniem .tf oraz składni HCL⁵⁴ (dopuszczalny jest również format JSON⁵⁵, aczkolwiek nie jest zalecany). Podczas uruchamiania aplikacji Terraform ładuje wszystkie pliki .tf w obrębie aktualnego katalogu, dlatego bardzo ważne jest, aby były one ze sobą powiązane oraz by nie trzymać różnych projektów jednym katalogu. Teoretycznie można utworzyć całą konfigurację w jednym pliku, jednak dobra praktyka jest taka, aby były to co najmniej trzy pliki:

- *main.tf* – konfiguracja infrastruktury
- *credentials.tf* – oddzielny plik z danymi do logowania (np. kluczami AWS)
- *terraform.tfvars* – plik z domyślnymi wartościami zmiennych
- *output.tf* – [opcjonalnie] plik ze zmiennymi które są generowane przez dostawcę w trakcie budowania infrastruktury, np. identyfikator maszyny wirtualnej

Po utworzeniu konfiguracji, ale jeszcze zanim zostanie ona wdrożona należy wydać dwie komendy:

- *terraform init* – na podstawie istniejącej konfiguracji doinstalowywane są wtyczki przypisane konkretnym usługom, takim jak np. wtyczka do obsługi chmury AWS.
- *terraform plan* – komenda ta sprawdza poprawność konfiguracji oraz informuje o tym jakie zmiany zostaną wykonane.

Akceptacja zmian odbywa się poprzez wpisanie komendy *terraform apply*. Niestety nie ma możliwości by wycofać ostatnią zmianę, jedyne co można zrobić w przypadku błędu to poprawić konfigurację ewentualnie usunąć całą infrastrukturę poleceniem *terraform destroy*.
[11]

⁵¹ <https://www.terraform.io/downloads.html> Dostęp: 01.06.2018r.

⁵² Wieloplatformowość – cecha pozwalająca na pracę na różnych platformach sprzętowych

⁵³ System uniksopodobny – system operacyjny budową zbliżony do systemu Unix

⁵⁴ HCL - HashiCorp Configuration Language [10]

⁵⁵ JSON - JavaScript Object Notation [9]

3.3 Dane logowania oraz zmienne lokalne

Procedurę budowania kodu infrastruktury należy zacząć od utworzenia plików z danymi logowania oraz ze zmiennymi lokalnymi. Na tym etapie pojawia się pierwsze pytanie – Skąd wziąć dane logowania? Otóż jest to para kluczy które były generowane w punkcie 3.1. Dobrą praktyką jest nie umieszczanie danych logowania bezpośrednio w projekcie, dlatego też należy zadeklarować zmienne i podawać klucze przy każdym uruchomieniu Terraform. Opcjonalnie można dodać je do pliku ze zmiennymi *terraform.tfvars*. W przypadku konta AWS plik *credentials.tf* powinien wyglądać następująco:

```
#### Credentials for AWS Account ####

variable "region" {
  default = "us-east-1"
}

variable "aws_access_key" {
  description = "Access Key for AWS account"
}

variable "aws_secret_key" {
  description = "Secret Key for AWS account"
}

provider "aws" {
  region      = "${var.region}"
  access_key  = "${var.aws_access_key}"
  secret_key  = "${var.aws_secret_key}"
}
```

Przyjrzyjmy się teraz strukturze pliku konfiguracyjnego Terraform, ponieważ w tym momencie jest zadeklarowany dostawcy usług (sekcja „provider”) z którego aplikacja ma korzystać, a co za tym idzie, po wydaniu polecenie *terraform init*, które wtyczki muszą zostać ściągnięte do folderu z konfiguracją.

W podanym kodzie również nie ma wprost wpisanych danych logowania, tylko zostały utworzone zmienne „aws_secret_key” oraz „aws_access_key”. Zmienne te zostaną wpisane bezpośrednio do pliku *terraform.tfvars*, czyli zbioru zmiennych, które są potrzebne

do wdrożenia konfiguracji. W przypadku projektowanej aplikacji wygląda dokument ten wygląda następująco:

```
#### Workshop project variables ####
# region = "us-east-1"
# aws_access_key =
# aws_secret_key =
# rds_username =
# rds_password =
# rds_database =
vpc_cidr = "10.100.0.0/16"
subnet_bastion = "10.100.0.0/27"
subnet_nat1a = "10.100.0.32/27"
subnet_nat1b = "10.100.0.64/27"
subnet_elb1a = "10.100.1.0/27"
subnet_elb1b = "10.100.1.32/27"
subnet_efs1a = "10.100.2.0/27"
subnet_efs1b = "10.100.2.32/27"
subnet_rds1a = "10.100.3.0/27"
subnet_rds1b = "10.100.3.32/27"
subnet_backend1a = "10.100.4.0/27"
subnet_backend1b = "10.100.4.32/27"
```

3.4 Wirtualna chmura prywatna

Podstawowym komponentem, bez którego nie uda się stworzyć infrastruktury w chmurze AWS jest wirtualna chmura prywatna VPC (z ang. Virtual Private Cloud). Jest to odizolowana sekcja chmury AWS, w której można uruchamiać zasoby AWS w zdefiniowanej przez siebie sieci wirtualnej. AWS pozwala na pełną konfigurację nad środowiskiem sieci wirtualnej, w tym na wybór zakresu adresów IPv4 oraz IPv6, tworzenie bram dostępowych, podsieci wirtualnych, konfigurację tablic routingu oraz konfigurację zabezpieczeń.

Na tym etapie należy zadeklarować adres sieci lokalnej z puli adresów prywatnych. W przykładzie posłużymy się adresacją 10.100.0.0/16, która została zadeklarowana w zmiennej „*vpc_cidr*”. Użyta została maska 16-bitowa ze względu na to, że w dalszym etapie sieć będzie dzielona na podsieci przeznaczone na konkretne składowe aplikacji oraz na potencjalną rozbudowę. Dodatkowo dla całego VPC zostanie uruchomiony protokół IPv6, jednak w tym

przypadku jego CIDR⁵⁶ jest generowany automatycznie. Zawartość pliku *VPC.tf* wygląda następująco:

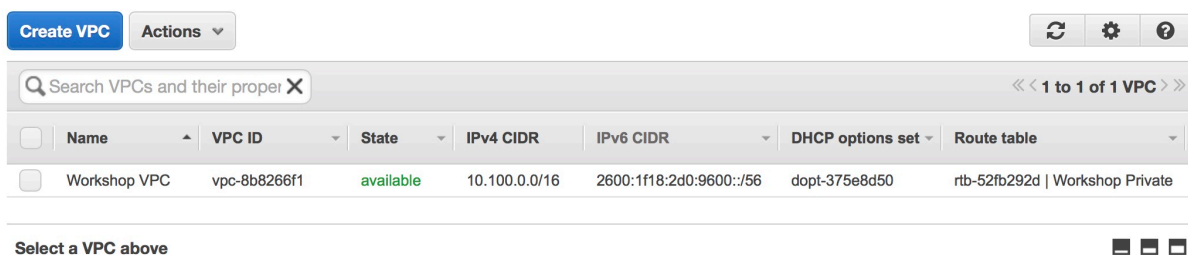
```
##### Main VPC #####
variable "vpc_cidr" {
  description = "Main CIDR for Infrastructure"
}

resource "aws_vpc" "workshop" {
  cidr_block                = "${var.vpc_cidr}"
  assign_generated_ipv6_cidr_block = "true"
  enable_dns_hostnames      = "true"
  enable_dns_support        = "true"

  tags {
    Name = "Workshop VPC"
  }
}
```

W poprzednich kodach źródłowych były zadeklarowane tylko zmienne (z ang. variables) oraz sekcja „provider”. Tym razem mamy już konkretne odwołanie do zasobu (z ang. resource), który ma zostać utworzony.

Wykonanie w tym momencie komendy *terraform apply* zakończyłoby się sukcesem co widać na poniższych zrzutach ekranu (Rys.3.4 i 3.5):



Rys. 3.4 Zrzut z AWS Web Management po zbudowaniu VPC

⁵⁶ CIDR (ang. Classless Inter-Domain Routing) – bezklasowa metoda przydzielania adresów IP

```

Plan: 1 to add, 0 to change, 0 to destroy.

Do you want to perform these actions?
  Terraform will perform the actions described above.
  Only 'yes' will be accepted to approve.

Enter a value: yes

aws_vpc.workshop: Creating...
  assign_generated_ipv6_cidr_block: "" => "true"
  cidr_block: "" => "10.100.0.0/16"
  default_network_acl_id: "" => "<computed>"
  default_route_table_id: "" => "<computed>"
  default_security_group_id: "" => "<computed>"
  dhcp_options_id: "" => "<computed>"
  enable_classiclink: "" => "<computed>"
  enable_classiclink_dns_support: "" => "<computed>"
  enable_dns_hostnames: "" => "true"
  enable_dns_support: "" => "true"
  instance_tenancy: "" => "<computed>"
  ipv6_association_id: "" => "<computed>"
  ipv6_cidr_block: "" => "<computed>"
  main_route_table_id: "" => "<computed>"
  tags.%: "" => "1"
  tags.Name: "" => "Workshop VPC"
aws_vpc.workshop: Still creating... (10s elapsed)
aws_vpc.workshop: Creation complete after 18s (ID: vpc-07fb1f7d)

Apply complete! Resources: 1 added, 0 changed, 0 destroyed.

Outputs:

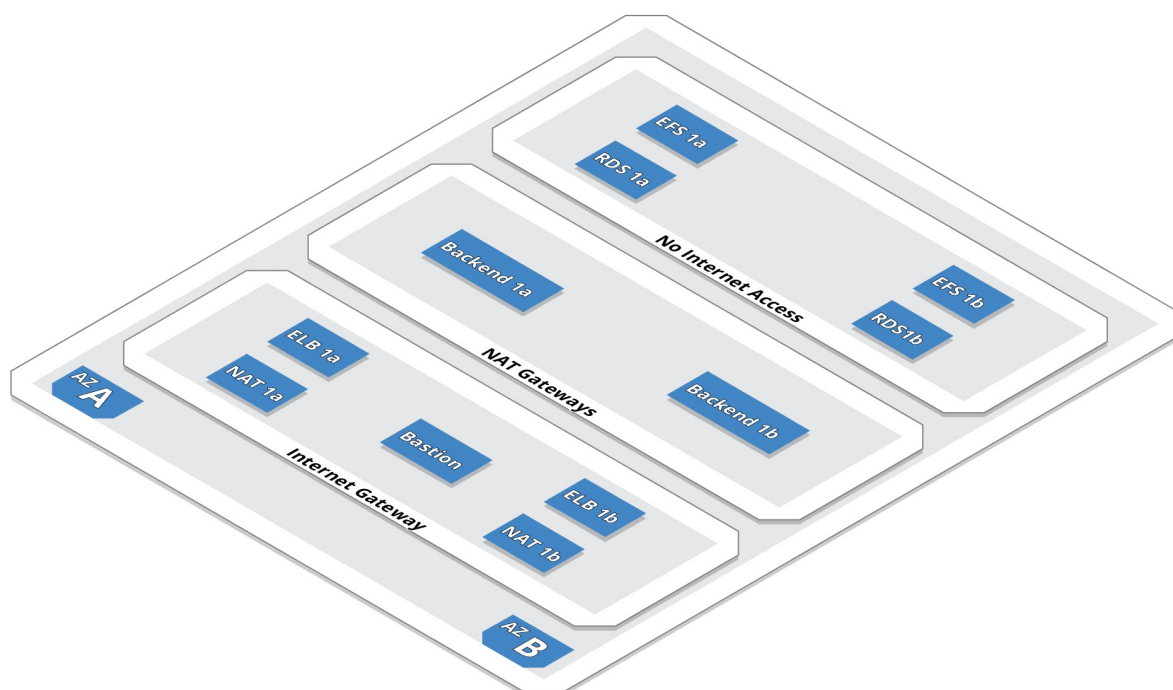
VPC ID = vpc-07fb1f7d

```

Rys. 3.5 Przykładowy wynik komendy terraform apply

3.4.1 Projekt sieci

Aplikacja ma działać w trybie Wysokiej Dostępności HA (z ang. High Availability) co oznacza, że przerwy w dostarczaniu aplikacji muszą być możliwie najniższe, a najlepiej jakby każda awaria była transparentna dla użytkownika końcowego. W takim przypadku należy założyć, że awarii, może ulec centrum danych (ang. Data Center), w którym jest utrzymywana aplikacja. Dlatego też wszystkie komponenty będą uruchamiane co najmniej dwa razy w dwóch różnych strefach dostępności. Jako strefy dostępności należy rozumieć co najmniej dwa centra danych świadczących te same usługi i działające w obrębie jednego regionu, ale geograficznie niezależne od siebie. W przypadku budowanego projektu zostanie wykorzystany region **us-east-1** znajdujący się na terenie Stanów Zjednoczonych oraz stref dostępności **a** i **b**. Strefy dostępności nie wymagają wdrażania przez użytkownika, jedyne co można zrobić to zadeklarować z których stref chciałoby się skorzystać.



Rys. 3.6 Struktura podsieci wraz z bramami dostępowymi⁵⁷

3.4.2 Podsieci - publiczne i prywatne

Dla każdej składowej aplikacji zostanie utworzona osobna podsieć, czyli sieci będące częścią innej większej sieci np. sieci o adresach 10.11.12.0/27 i 10.11.12.32/27 są podsieciami sieci 10.11.12.0/26. Zostaną one zdublowane tak aby występowały zarówno w strefie, a jak i b. Plan ten został zobrazowany na Rys. 3.6. Wszystkie podsieci będą budowane razem z komponentami, które zostaną do nich przypisane, np. w pliku konfiguracyjnym z bazami danych znajdą się odpowiednie sekcje budujące podsieci, umieszczając je w odpowiednich strefach oraz przypisując do odpowiedniej tablicy routingu. Przykładowy plik konfiguracyjny dla podsieci w strefie dostępności a:

```
variable "subnet_EXAMPLE" {
  description = "EXAMPLE Subnet in 1st Availability Zone"
}

resource "aws_subnet" "workshop-example" {
```

⁵⁷ Opracowanie własne

```

vpc_id                = "VPC_ID"
cidr_block             = "192.168.1.0/24"
map_public_ip_on_launch = "true"
availability_zone      = "us-east-1a"

tags {
  Name = "EXAMPLE SUBNET"
}
}

```

3.4.3 Bramy domyślne oraz tablice routingu

Kolejnym krokiem w budowaniu podstaw infrastruktury jest utworzenie bram sieciowych, czyli urządzeń odpowiedzialnych za kierowanie ruchem wychodzącym i przychodzącym pomiędzy urządzeniami wewnątrz sieci a Internetem lub innymi sieciami prywatnymi, gdy np. jest skonfigurowane połączenie VPN. W AWS są trzy typy bram sieciowych:

- **IGW (Internet Gateway)** – jest to podstawowy komponent wirtualnej chmury prywatnej AWS odpowiedzialny za komunikację pomiędzy instancjami w VPC i Internetem. Brama Internetowa ma za zadanie udostępniać dostęp do Internetu oraz odpowiada za translację adresów (NAT) dla usług, którym przypisano publiczny adres IP. IGW obsługuje zarówno adresację IPv4 jak IPv6.⁵⁸
- **EIGW (Egress-Only Internet Gateway)** – jest to komponent dostępowy do Internetu przeznaczony dla protokołu IPv6. Ten typ bramy w odróżnieniu od IGW nie dopuszcza ruchu z Internetu do instancji, a jedynie zezwala na komunikację wychodzącą.⁵⁹
- **NAT (Network Address Translations)** – umożliwia komunikację z Internetem instancjom znajdującym się w prywatnych podsieciach, tj. takich które nie mają przypisanych publicznych adresów IP. Urządzenie NAT przekształca źródłowy adres IPv4 na swój własny adres dla połączeń wychodzących, a dla odpowiedzi lub połączeń przychodzących przekazuje ruch na odpowiedni adres prywatny. W AWS urządzenia

⁵⁸ https://docs.aws.amazon.com/AmazonVPC/latest/UserGuide/VPC_Internet_Gateway.html Dostęp: 26.05.2018r.

⁵⁹ <https://docs.aws.amazon.com/AmazonVPC/latest/UserGuide/egress-only-internet-gateway.html> Dostęp: 26.05.2018r.

NAT można uruchamiać na dwa sposoby jako NAT Gateway lub Instancje NAT. Pierwszy sposób jest bardziej zalecany, gdyż zapewnia większą dostępność oraz nie ma ograniczeń w ruchu sieciowym, jednak Instancja NAT pozwala na pełną kontrolę nad ruchem przychodzącym i wychodzącym.⁶⁰

Aby utworzyć IGW oraz EIGW należy stworzyć plik *Gateways.tf* o następującej treści:

```
#### Internet Gateway ####
resource "aws_internet_gateway" "workshop" {
  vpc_id = "${aws_vpc.workshop.id}"

  tags {
    Name = "Workshop IGW"
  }
}

#### Egress Only Internet Gateway ####
resource "aws_egress_only_internet_gateway" "workshop" {
  vpc_id = "${aws_vpc.workshop.id}"
}
```

Ponieważ, brama domyślna jest już utworzona należy skonfigurować infrastrukturę w taki sposób, aby ruch Internetowy był kierowany na nią. Służą do tego tablice routingu, czyli zestawy reguł, tak zwanych tras, które określają, dokąd ma być kierowany ruch sieciowy. W tym przypadku utworzymy dwie tablice – publiczną i prywatną, z czego ta druga będzie domyślną tablicą dla wszystkich nowopowstałych podsieci i nie będzie ona zezwalała na ruch wychodzący i przychodzący poza adresację VPC. Konfiguracja znajduje się w pliku *Route_Tables.tf*.

Ze względu na to, że projekt (Rys. 3.4) zakłada izolację serwerów produkcyjnych od świata za pomocą usługi NAT⁶¹, należy utworzyć również urządzenia NAT. W tym wypadku wystarczające są oferowane przez Amazon niezarządzalne NAT Gateways. Aby uruchomić NAT należy najpierw utworzyć podsieci w puli publicznej, w których urządzenia te się znajdą oraz dla każdego urządzenia zarezerwować stały publiczny adres IPv4.

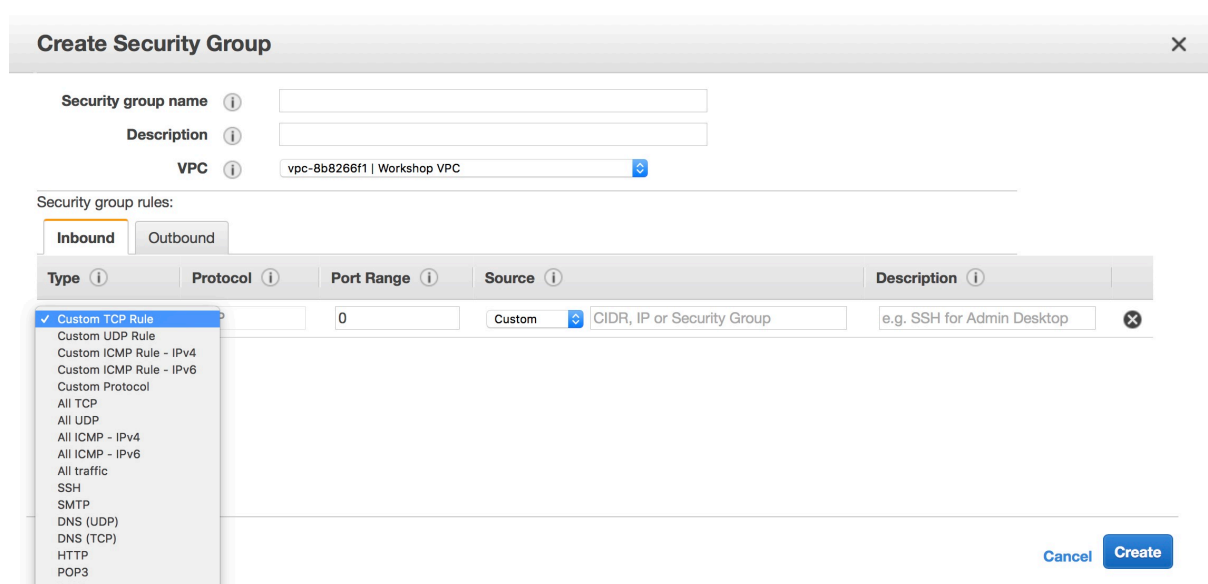
⁶⁰ https://docs.aws.amazon.com/AmazonVPC/latest/UserGuide/VPC_NAT_Instance.html Dostęp: 26.05.2018r.

⁶¹ NAT – Network Address Translations

Należy również utworzyć kolejne tablice routingu osobne dla każdego urządzenia NAT. W tych tablicach zostanie też wykorzystana zbudowana wcześniej brama EIGW⁶² dla połączeń protokoły IPv6. Konfiguracja urządzeń oraz tablic routingu znajduje się w plikach *NAT_Gateways.tf* oraz *NAT_RouteTables.tf*.

3.5 Grupy Zabezpieczeń (z ang. Security Groups)

Jeden z najważniejszych komponentów podczas budowania infrastruktury w chmurze AWS. Jest to wirtualna zaporą ogniową (z ang. Firewall) kontrolująca ruch przychodzący i wychodzący. Istotne jest, że grupy zabezpieczeń działają na poziomie instancji a nie poziomie podsieci, dlatego każda instancja musi być przypisana do przynajmniej jednej grupy zabezpieczeń.⁶³ W środowisku AWS Grupy Zabezpieczeń przechowywane są w formie tablicy, w której należy określić protokół (TCP⁶⁴ lub UDP⁶⁵), port lub zakres portów oraz źródło, w przypadku połączeń przychodzących lub cel, w przypadku połączeń wychodzących. Ruch odrzucony przez Grupę Zabezpieczeń w żaden sposób nie obciąża instancji, dlatego przy dobrze skonfigurowanych grupach nie ma potrzeby stosowania firewalli na instancjach. Rys. 3.7 przedstawia zrzut ekranu z dodawania reguły przychodzącej.



Rys. 3.7 Dodawanie wpisu do Grupy Zabezpieczeń w paneli AWS

⁶² EIGW (ang. Egress-Only Internet Gateway)

⁶³ https://docs.aws.amazon.com/AmazonVPC/latest/UserGuide/VPC_SecurityGroups.html Dostęp: 26.05.2018r

⁶⁴ TCP (ang. Transmission Control Protocol) – Protokół sterowania transmisją

⁶⁵ UDP (ang. User Datagram Protocol) – Protokół pakietów użytkownika

3.6 Serwer Bastion

Zgodnie z założeniami wdrażana aplikacja ma być automatycznie skalowalna i uruchamiana oraz konfigurowana za pomocą narzędzi do tego przeznaczonych. Jednakże, zdarza się, że zaistnieje potrzeba by osobiście zdiagnozować aplikację lub serwer, na którym jest uruchomiona aplikacja. Ponieważ w zaprezentowanej architekturze infrastruktury nie uwzględniono zestawiania połączenia VPN⁶⁶ ze środowiskiem lokalnym to warto skorzystać z tak zwanego serwera Bastion. Jest to nic innego jak maszyna wirtualna znajdująca się wewnątrz publicznej podsieci oraz mającej dostęp do prywatnej części infrastruktury. Połączenie z takim serwerem odbywa się najczęściej za pomocą protokołów SSH⁶⁷ lub RDP⁶⁸ i dopiero z niego można łączyć się dalej ze składowymi aplikacjami w celu diagnozy i ewentualnej konfiguracji. Ważne jest, aby odpowiednio zabezpieczyć taki serwer, tj. utworzyć odpowiednie grupy zabezpieczeń, ale też np. nie pozostawiać takiej instancji cały czas uruchomionej, gdyż z racji tego, że jest ona publiczna jest wystawiona na próby włamania przez cyberprzestępców.

Tworzenie takiej instancji należy zacząć od utworzenia pary kluczy RSA⁶⁹ (prywatny oraz publiczny) do zabezpieczenia połączeń SSH⁷⁰, gdyż w gotowych obrazach maszyn wirtualnych w AWS nie ma możliwości logowania z użyciem hasła. W przypadku systemów uniksopodobnych⁷¹ można to zrobić wykonując w wierszu poleceń komendę:

```
ssh-keygen -f workshop-bastion -t rsa
```

Dodatkowo do uruchomienia maszyny wirtualnej należy uwzględnić zależności. Do takich wymagań należy określenie z jakiego obrazu ma być ona uruchomiona – w tym przypadku został użyty oficjalny obraz dystrybucji Amazon Linux⁷² w wersji 2. Co więcej na tym etapie określany jest typ instancji w zależności od wymaganych zasobów oraz możliwości finansowych. Obowiązkowe jest również by określić VPC, utworzyć podsieć w jakiej ma się znajdować instancja oraz przypisać ją do odpowiedniej tablicy routingu. Kolejnym krokiem jest utworzenie grupy zabezpieczeń, w której należy zezwolić na ruch przychodzący z dowolnego miejsca na porcie 22 (SSH). Na tym etapie należy też skopiować wygenerowany

⁶⁶ VPN (ang. Virtual Private Network) – Wirtualna sieć prywatna

⁶⁷ SSH (ang. Secure Shell) – szyfrowany protokół komunikacyjny działający w modelu klient-serwer.

⁶⁸ RDP (ang. Remote Desktop Protocol) – protokół zdalnej komunikacji z terminalem graficznym działający w środowisku MS Windows

⁶⁹ RSA – asymetryczny algorytm kryptograficzny. Jest to obecnie najpopularniejszy algorytm kryptograficzny wykorzystujący klucz publiczny.

⁷⁰ Secure Shell (SSH) – szyfrowany protokół komunikacyjny działający w modelu klient-serwer.

⁷¹ System uniksopodobny – system operacyjny budową zbliżony do systemu Unix

⁷² <https://aws.amazon.com/amazon-linux-ami/> Dostęp: 26.05.2018r.

wcześniej klucz publiczny do zasobów AWS. Konfiguracja serwera Bastion znajduje się w pliku *Bastion.tf*.

Podstawowym zadaniem tego serwera jest rozwiązywanie problemów związanych z instancjami produkcyjnymi, systemem plików czy bazą danych, dlatego też podczas uruchamiania instancji system automatycznie zainstaluje przydatne pakiety. Są to między innymi sterowniki do systemu EFS, klient MySQL, klient telnet jak również edytor VIM. Za instalację i wstępną konfigurację odpowiada pole *user_data* które w tym wypadku wskazuje na plik *bastion-user_data.yml*, który z racji tego, iż została wybrana autorska dystrybucja Linuksa udostępniana przez Amazon, nie jest typowym skryptem w języku Bash⁷³, tylko plikiem YAML który na etapie uruchamiania jest przetwarzany na komendy zrozumiałe dla systemu Amazon Linux. Zawartość pliku *bastion-user_data.yml* wygląda następująco:

```
#cloud-config
repo_update: true
repo_upgrade: all
preserve_hostname: false
hostname: workshop-bastion
manage_etc_hosts: true

packages:
- amazon-efs-utils
- mc
- telnet
- mysql

runcmd:
- mkdir -p /mnt/efs
- echo "${EFS_ID}:/mnt/efs efs tls,_netdev" >> /etc/fstab
- mount -a -t efs defaults
- amazon-linux-extras install vim
```

Opcjonalnie można byłoby użyć standardowych poleceń powłoki Bash. Zawartość pliku *bastion-user_data.yml* wyglądałaby wówczas następująco:

⁷³ Bash – powłoka systemowa Unix.

```
#!/bin/bash
yum update -y &&
yum upgrade -y &&
yum install amazon-efs-utils mc telnet mysql -y &&
hostname workshop-bastion &&
echo "workshop-bastion" > /etc/hostname &&
mkdir -p /mnt/efs &&
echo "${EFS_ID}:/mnt/efs efs tls,_netdev" >> /etc/fstab &&
mount -a -t efs defaults &&
amazon-linux-extras install vim
```

Na powyższych przykładach widać, że konfiguracje nieznacznie się różnią. Jednak, użycie *‘cloud-config’* dodaje wpisy do usługi odpowiedzialnej za konfigurację systemu – cloud-init oraz przy okazji zabezpiecza przed zmianami takimi jak np. utrata nazwy podczas aktualizacji (w infrastrukturze AWS domyślne nazwy systemów to ich adresy IP).

Rozdział 4. Warstwa obliczeniowa

4.1 Baza danych

Amazon RDS (z ang. Relational Database Service) to usługa oferująca łatwą w konfiguracji, utrzymaniu oraz skalowaniu relacyjną bazę danych w chmurze. Podstawową zaletą tego rozwiązania jest odciążenie administratora od takich zadań jak obsługa sprzętu, konfiguracji serwerów czy tworzenia kopii bezpieczeństwa, dzięki czemu można skupić się na utrzymaniu aplikacji oraz zapewnić jej wysoką dostępność.⁷⁴

W ramach RDS dostępne są wszystkie popularne silniki bazodanowe takie jak Oracle⁷⁵, Microsoft SQL Server⁷⁶, MySQL⁷⁷ czy PostgreSQL⁷⁸. Dodatkowo Amazon oferuje swoją własną bazę danych Aurora⁷⁹, która występuje w dwóch wersjach kompatybilności – z MySQL oraz z PostgreSQL. Ze względu na fakt, że to właśnie Aurora jest najlepiej zoptymalizowaną bazą danych pod usługi w AWS zostanie ona wykorzystana jako silnik bazodanowy dla implementowanej aplikacji. W tym wypadku wybranym rozwiązaniem będzie nie pojedyncza instancja bazodanowa a klaster, ze względu na wymaganą wysoką dostępność. W związku z tym należy utworzyć podsieci w przynajmniej dwóch strefach dostępności puli prywatnej, tj. bez dostępu do sieci Internet oraz dodanie ich do tak zwanej grupy podsieci, czyli kolekcji podsieci w których będą znajdowały się instancje z bazami danych. W przypadku dodania większej ilości podsieci niż planowanych instancji baz danych, nie mamy bezpośredniego wpływu na wybór podsieci – o tym decydują automaty AWSowe. Kolejnym krokiem jest stworzenie grupy zabezpieczeń zezwalającej na dostęp do bazy danych. W związku tym, że w projekcie jest wykorzystywana baza danych kompatybilna z MySQL to komunikacja odbywać się będzie na standardowym porcie 3306 po protokole TCP. Dostęp zostaje przyznany tylko dla instancji serwujących aplikację produkcyjną oraz serwera Bastion. Ze względów bezpieczeństwa w konfiguracji nie zostały podane dane dostępowe do bazy danych, a jedynie zadeklarowane zmienne *var.rds_username* i *var.rds_password*, którym można przypisać wartości w pliku *terraform.tfvars* lub aplikacja Terraform zapyta o nie podczas budowania infrastruktury. Konfiguracja bazy danych znajduje się w pliku *RDS.tf*.

⁷⁴ <https://aws.amazon.com/rds/> Dostęp: 02.06.2018r.

⁷⁵ <https://aws.amazon.com/rds/oracle/> Dostęp: 02.06.2018r.

⁷⁶ <https://aws.amazon.com/rds/sqlserver/> Dostęp: 02.06.2018r.

⁷⁷ <https://aws.amazon.com/rds/mysql/> Dostęp: 02.06.2018r.

⁷⁸ <https://aws.amazon.com/rds/postgresql/> Dostęp: 02.06.2018r.

⁷⁹ <https://aws.amazon.com/rds/aurora/> Dostęp: 02.06.2018r.

4.2 Sieciowy system plików

Amazon Elastic File System (EFS) to oparta o protokół NFS⁸⁰ usługa zapewniająca prostą, skalowalną i elastyczną pamięć masową, którą można wykorzystać z zasobami Amazon EC2. EFS pozwala w bardzo szybko i łatwo tworzyć sieciowe systemy plików. Podstawową zaletą tej usługi jest jej elastyczność, czyli automatyczne zmniejszenie i powiększanie w zależności od zapotrzebowania na miejsce. Dodatkowo wraz ze zwiększaniem powierzchni wzrasta maksymalna przepustowość sieci oraz dopuszczalna ilość operacji wejścia/wyjścia (IOPS)⁸¹. Ponieważ jest to sieciowa pamięć masowa, także pozwalana na używanie jej w tym samym czasie na wielu instancjach jednocześnie.

Tworzenie zasobu EFS składa się z dwóch części – utworzenia bramy dostępowej, tak zwanego punktu dostępowego oraz utworzenia systemu plików. W przypadku tworzenia za pomocą AWS Management Console te operacje są połączone, jednak w Terraform należy wykonać te operacje osobno. Punkty montowania muszą być przypisane do konkretnych podsieci oraz grup zabezpieczeń. Podsieci w których zostaną uruchomione punkty montowania, będą podsieciami prywatnymi bez dostępu do Internetu, co jest spowodowane faktem, iż pełna kontrola nad system EFS jest po stronie AWS. Użytkownik może tylko podmontować zasoby w zarządzanych przez siebie instancjach.

Komunikacja z usługą Amazon EFS odbywa się na standardowym porcie NFS, tj. 2049. Dlatego też w grupie zabezpieczeń zostaje zdefiniowany tylko ten port do połączeń przychodzących. W przypadku wdrażanej aplikacji jedynymi instancjami, które będą potrzebowały dostępu do EFS będą serwery EC2 z zainstalowaną aplikacją oraz serwer Bastion, w razie potrzeby wprowadzania ręcznych zmian w plikach. W pliku *EFS.tf* znajduje się pełna konfiguracja systemu EFS.

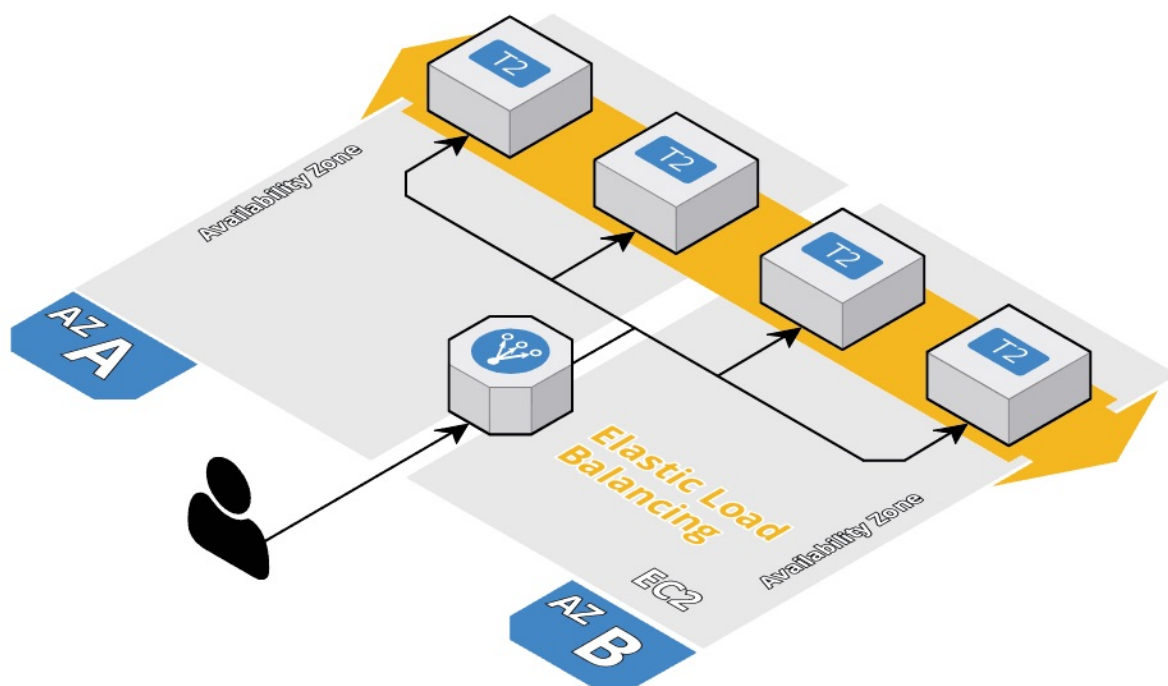
4.3 Równoważenie obciążenia

Amazon Elastic Load Balancing (ELB) jest usługą oferującą równoważenie ruchu poprzez rozsyłanie go pomiędzy wieloma obiektami docelowymi, mogą to być instancje EC2, kontenery ECS lub adresy IP. ELB nie wymaga do działania konfiguracji w trybie wysokiej dostępności, to znaczy, że może być skonfigurowana do obsługi jednej strefy dostępności

⁸⁰ NFS (ang. Network File System) – Sieciowy System Plików [14]

⁸¹ IOPS (ang. input/output operations per second) - jednostka wydajności dysków, określająca ilość operacji wejścia wyjścia jakie urządzenie może wykonać w ciągu sekundy. [13]

i w jej obrębie równoważyć ruch pomiędzy obiektami. W ramach AWS istnieją trzy typy balanserów ruchu⁸²:



Rys. 4.1 Schemat działania równoważnika ruchu⁸³

- **Równoważnik aplikacyjny ALB** (z ang. Application Load Balancer) – moduł równoważący obciążenie aplikacji, działający w warstwie 7 modelu OSI⁸⁴. Jest to najlepszy wybór do równoważenia ruchu HTTP/HTTPS
- **Równoważnik sieciowy NLB** (z ang. Network Load Balancer) – moduł działający w warstwie 4 modeli OSI, najlepszy do równoważenia ruchu po protokole TCP. Ten typ balansera jest w stanie obsłużyć miliony żądań na sekundę, a także zapewnia małe opóźnienia (z ang. Low Latency)
- **Klasyczny równoważnik CLB** (z ang. Classic Load Balancer) – jest to moduł starego typu przystosowany do obsługi EC2-Classic. Balanser pracuje zarówno w warstwie 7 jak i 4 modelu OSI. Nie jest już rozwijany.

⁸² <https://aws.amazon.com/elasticloadbalancing/> Dostęp: 26.05.2018r

⁸³ Opracowanie własne

⁸⁴ OSI (ang. SO Open Systems Interconnection Reference Model) – standard opisujący strukturę komunikacji w sieci.

Ze względu na rodzaj wdrażanego systemu, tj. aplikacja internetowa, w pracy zostanie wykorzystany Application Load Balancer, który będzie nasłuchiwał na standardowych portach HTTP i HTTPS, czyli 80 i 443 oraz będzie łączył się do instancji serwerowych na porcie 80. W tym celu należy utworzyć grupę docelową (Target Group) oraz zadeklarować dwa porty nasłuchujące (ang. Listners). Grupa docelowa jest zasobem, do którego w późniejszym etapie będą automatycznie dodawane instancje serwujące aplikację, a jej konfiguracja ogranicza się do podania protokołu i portu, na którym będzie łączyć się do maszyn wirtualnych. Dodatkowo został zadeklarowany czas wyrejestrowywania instancji w przypadku braku odpowiedzi oraz sesja 'stickyness' ustawiająca „ciasteczko” po stronie klienta w taki sposób, aby przez 60sekund łączył się do tego samego serwera. W przypadku konfiguracji portów nasłuchujących należy zadeklarować w zasadzie tylko port i protokół oraz wybrać domyślną akcję, w tym przypadku jest to przekierowanie całego ruchu na konkretną grupę docelową. Dodatkowo podczas tworzenia portu nasłuchującego dla protokołu HTTPS należy ustawić domyślny certyfikat, którym będzie szyfrowany ruch oraz wybrać politykę bezpieczeństwa.⁸⁵

Balansery pełnią rolę punktu dostępowego dla użytkowników, dlatego też należy utworzyć podsieci w puli publicznej przeznaczone tylko dla nich. W AWS nie ma możliwości przypisania stałych adresów IP⁸⁶ do Load Balncerów aplikacyjnych, jednak po utworzeniu zostanie wygenerowany adres DNS⁸⁷ za pomocą którego będzie możliwa komunikacja z równoważnikiem ruchu. Ponieważ dostęp do równoważników jest publiczny należy go odpowiednio zabezpieczyć. Na tym etapie będzie to utworzenie odpowiedniej grupy zabezpieczeń, dopuszczającej ruch jedynie na portach które są obsługiwane, w tym przypadku porty 80 i 443 dla protokołów IPv4 oraz IPv6 dla wszystkich połączeń przychodzących oraz port 80 protokołu IPv4 dla połączeń wychodzących, ale tylko dla podsieci backendowych. Konfiguracja Load Balancerów znajduje się w pliku *ELB.tf*.

⁸⁵ <https://docs.aws.amazon.com/elasticloadbalancing/> Dostęp: 26.05.2018r

⁸⁶ IP (ang. Internet Protocol) – podstawowy protokół komunikacyjny w sieciowej warstwie modelu OSI

⁸⁷ DNS (ang. Domain Name System) – system nazw domenowych, usługa tłumacząca adresy zrozumiałe dla użytkowników na adresy internetowe (IP)

4.4 Ustawienia serwera aplikacyjnego

WordPress jest to darmowa aplikacja typu CMS⁸⁸ bazująca na języku PHP⁸⁹ oraz bazie danych MySQL. Aplikacja jest głównie wykorzystywana do budowania blogów⁹⁰ lub małych portali informacyjnych i jest wykorzystywana przez ponad 50% wszystkich stron tego typu⁹¹. Ponieważ jest to aplikacja internetowa, do uruchomienia wymagany jest serwer HTTP oraz interpreter PHP. W projekcie zostanie użyty Nginx⁹² w wersji 1.12 oraz PHP w wersji 7.2. Ze względu na to, iż aplikacja będzie działała bez przerw, katalog ze plikami WordPress zostanie wgrany na system plików EFS, który został utworzony w podrozdziale 4.2.

Podobnie jak w przypadku serwera Bastion, dla instancji Backendowych należy wygenerować parę kluczy RSA, do nawiązywania połączeń SSH, można to zrobić komendą:

```
ssh-keygen -f workshop-backend -t rsa
```

Należy pamiętać, że otrzymany plik *workshop-backend.pub* musi znaleźć się w katalogu z plikami konfiguracyjnymi tworzonej infrastruktury. Plik z kluczem prywatnym należy przechowywać w bezpiecznym miejscu tak aby nie miały do niego inne osoby.

W projekcie proces budowy instancji Backendu został podzielony na cztery osobne pliki, wzajemnie się uzupełniające, są to:

- *Backend.tf* – w pliku tym znajduje się konfiguracja podstawowych zależności takich jak grupy zabezpieczeń oraz podsieci. W przypadku grup zabezpieczeń, zezwolony został ruch przychodzący na porcie 22 (SSH) dla całej podsieci Bastion oraz na porcie 80 (HTTP) dla podsieci z równoważnikami ruchu. Z racji, że instancje potrzebują dostępu do takich zasobów jak NAT, EFS, RDS ruch wychodzący nie posiada ograniczeń. Utworzone zostały także dwie odrębne podsieci w dwóch różnych strefach dostępności. Podsieci te zostały skonfigurowane w taki sposób, aby korzystały ruch Internetowy był kierowany na urządzenia NAT.
- *Backend_LC.tf* – jest to konfiguracja uruchamiania. Jej zakres obejmuje wybór domyślnego obrazu instancji, w tym wypadku podobnie jak przy maszynie Bastion

⁸⁸ CMS (ang. Content Management System) – system zarządzania treścią, jest to oprogramowanie ułatwiające tworzenie i zarządzanie stronami WWW.

⁸⁹ PHP (ang. Personal Home Page) – język programowania zaprojektowany do budowania stron internetowych oraz aplikacji webowych.

⁹⁰ Blog (ang. Web log) – typ strony internetowej zawierający uporządkowane chronologicznie wpisy.

⁹¹ Źródło: <https://trends.builtwith.com/cms> Dostęp: 28.05.2018r.

⁹² <https://www.nginx.com> Dostęp: 28.05.2018r.

wybór padł na Amazon Linux 2. Na tym etapie też jest wgrywany uprzednio wygenerowany klucz publiczny, za pomocą którego w wyjątkowych przypadkach będzie nawiązywane połączenie z maszynami produkcyjnymi. Ostatnim elementem w zakresie tego pliku jest utworzenie szablonu uruchamiania (Launch Configuration), który zostanie wykorzystany przez grupę automatycznego skalowania do uruchamiania instancji EC2. Podczas tworzenia konfiguracji uruchamiania należy podać takie informacje o instancjach jak identyfikator obrazu maszyny Amazon Linux (AMI), typ instancji, para kluczy, grupy zabezpieczeń.

- *Backend_AS.tf* – plik z informacjami odnośnie grupy automatycznego skalowania (Autoscaling Group). Grupa ta odpowiada za utrzymanie odpowiedniej ilości instancji potrzebnych do obsługi obciążenia aplikacji, jak również dopilnowania, aby zawsze działała minimalna ilość instancji. By utworzyć taką grupę należy określić minimalną ilość pracujących instancji, maksymalną ilość oraz wartość oczekiwaną. Wymagane są też takie wartości jak podsieci i strefy dostępności w których maszyny wirtualne będą umieszczone oraz grupa docelowa do której będą automatycznie dodawane. Instancje są uruchamiane automatycznie a ich konfiguracja jest pobierana z utworzonego wcześniej szablonu uruchamiania, który również musi być określony w konfiguracji grupy skalowania.
- *Backend-user_data.yml* – analogicznie do serwera Bastion, jest to konfiguracja startowa instancji. Zawarte są w niej takie informacje jak wymagane pakiety oraz punkty montowania zasobów EFS. Dodatkowo pobrana zostaje konfiguracja serwera Nginx, która została wcześniej przygotowana i umieszczona na serwerze Amazon S3⁹³. W konfiguracji zawarta jest również weryfikacja czy aplikacja WordPress została zainstalowana, a w przypadku, gdy system nie odnajdzie katalogu z aplikacją zostaje ona pobrana z Internetu oraz rozpakowana do katalogu `/var/www/html/wordpress`. Sprawdzenie odbywa się poprzez wykonanie następującego polecenia warunkowego:

```
if
    cd /var/www/html/wordpress;
then
    exit;
```

⁹³ Amazon S3 (Amazon Simple Storage Service) – usługa przechowywania danych a AWS.

```

else
    curl -o wordpress.tar.gz -fSL
    "https://wordpress.org/wordpress-latest.tar.gz";
    tar -xzf wordpress.tar.gz -C /var/www/html/;
    rm -f wordpress.tar.gz;
    chown -R apache:apache /var/www/html/wordpress;
fi

```

Uruchomienie serwerów z taką konfiguracją pozwala na uruchomienie aplikacji oraz rozpoczęcie konfiguracji aplikacji WordPress. W tym celu należy skorzystać z informacji, które zostały zwrócone w terminalu po zbudowaniu infrastruktury, a wcześniej zadeklarowane w pliku *output.tf* (Rys. 4.2)

```

xavar@Norbet-MBP ..a/terraform 2
subnet_id: "" => "subnet-19b46c37"
tags.%: "" => "1"
tags.Name: "" => "Workshop Bastion"
tenancy: "" => "<computed>"
user_data: "" => "b19aa3b093f9da66fe5ab3a4b5049cb43edf88c7"
volume_tags.%: "" => "<computed>"
vpc_security_group_ids.#: "" => "<computed>"
aws_autoscaling_group.backend: Still creating... (40s elapsed)
aws_instance.bastion: Still creating... (10s elapsed)
aws_autoscaling_group.backend: Still creating... (50s elapsed)
aws_instance.bastion: Creation complete after 19s (ID: i-08a9bc9a0eb6a813f)
aws_autoscaling_group.backend: Still creating... (1m0s elapsed)
aws_autoscaling_group.backend: Still creating... (1m10s elapsed)
aws_autoscaling_group.backend: Creation complete after 1m12s (ID: Backend)

Apply complete! Resources: 2 added, 2 changed, 1 destroyed.

Outputs:

Bastion Address = ec2-184-72-173-35.compute-1.amazonaws.com
Database Address = workshop-rds-cluster.cluster-cuegzcrzvdci.us-east-1.rds.amazonaws.com
Load Balancer Address = Workshop-ALB-670213767.us-east-1.elb.amazonaws.com
Route53 Name Servers = [
    ns-1276.awsdns-31.org,
    ns-1828.awsdns-36.co.uk,
    ns-20.awsdns-02.com,
    ns-577.awsdns-08.net
]
VPC ID = vpc-8b8266f1

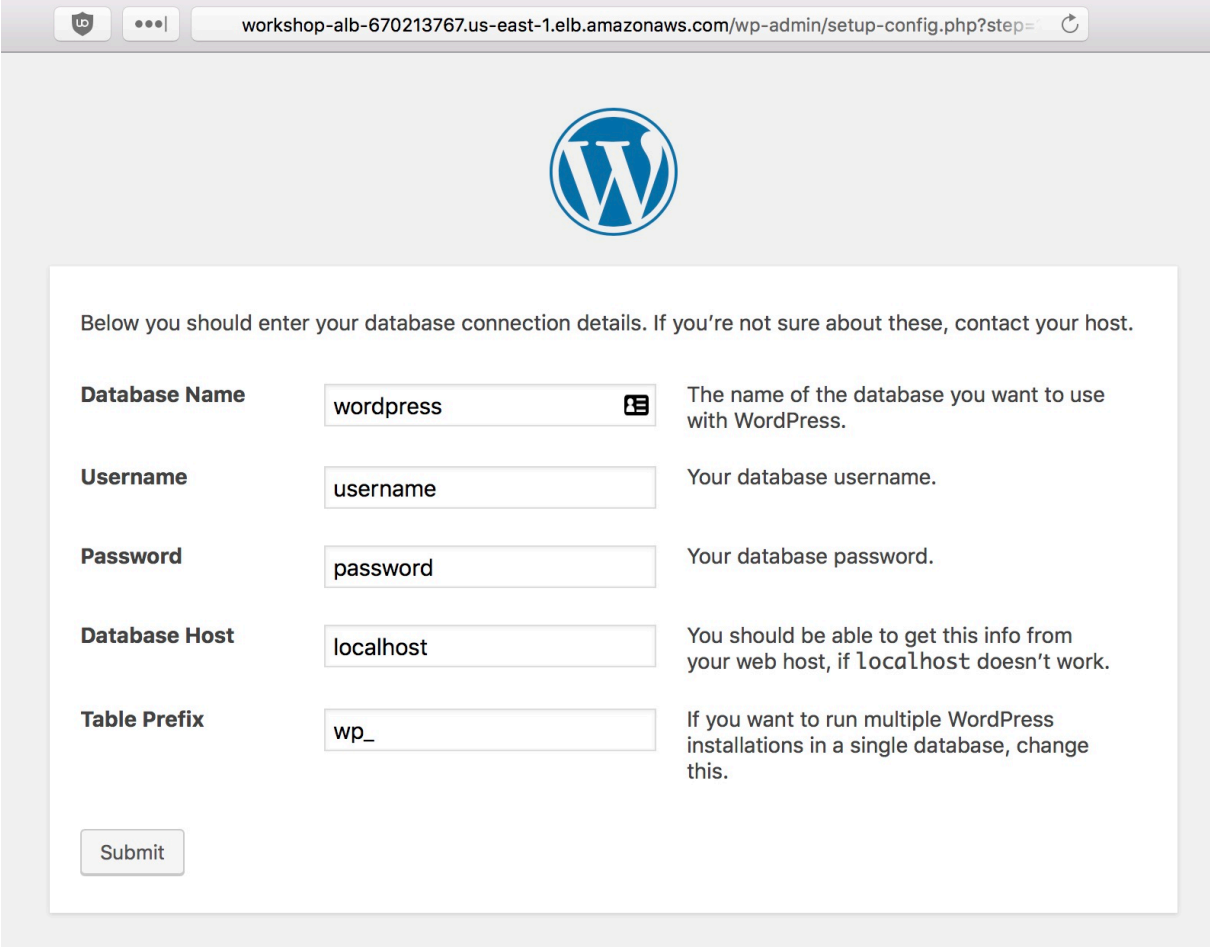
```

Rys. 4.2 Informacje końcowe aplikacji

By rozpocząć konfigurację aplikacji WordPress należy użyć adresu opisanego jako *Load Balancer Address* i użyć go w przeglądarce internetowej (np. Google Chrome⁹⁴), co spowoduje uruchomienie konfiguratora WordPress. Na trzeciej stronie konfiguracji aplikacja poprosi o dane logowania do bazy danych (Rys. 4.3). W polach *Database Name*, *Username* oraz *Password* należy podać wartości określone w rozdziale 4.1 zaś w polu *Database*

⁹⁴ <https://www.google.com/chrome/> Dostęp: 04.06.2018r.

Host należy umieścić wygenerowany przez Terraform adres opisany jako *Database Address*. Tak przygotowana aplikacja jest gotowa do publicznej dystrybucji.



The screenshot shows a web browser window with the URL `workshop-alb-670213767.us-east-1.elb.amazonaws.com/wp-admin/setup-config.php?step=`. The page features the WordPress logo at the top center. Below the logo, a message states: "Below you should enter your database connection details. If you're not sure about these, contact your host." The form contains five input fields, each with a label, a value, and a description:

Field Label	Value	Description
Database Name	wordpress	The name of the database you want to use with WordPress.
Username	username	Your database username.
Password	password	Your database password.
Database Host	localhost	You should be able to get this info from your web host, if localhost doesn't work.
Table Prefix	wp_	If you want to run multiple WordPress installations in a single database, change this.

At the bottom left of the form is a "Submit" button.

Rys. 4.3 Konfiguracja aplikacji WordPress

Rozdział 5. Dystrybucja aplikacji

Ostatnim etapem wdrażania aplikacji jest udostępnienie jej do użytku publicznego. Do realizacji celu zostały wykorzystane takie komponenty infrastruktury AWS jak Route53 oraz CloudFront. Dostęp publiczny do aplikacji odbędzie się za pomocą publicznej domeny *workshop.edu.pl*.

5.1 System translacji nazw Internetowych – Route53

Amazon Route53 jest wysoce dostępna i skalowalna usługa rozwiązywania nazw internetowych DNS (z ang. Domain Name System) oferowana w ramach usług AWS. Amazon Route53 bardzo dobrze łączy się z innymi usługami AWS, takimi jak Elastic Load Balancing czy CloudFront, poprzez udostępnienie funkcji aliasu dla generowanych przez AWS adresów DNS. Funkcja ta różni się od typowego CNAME⁹⁵, że nie zachodzi tu mapowanie aliasu na inny adres (np. *domena1.pl* na *domena2.pl*) tylko jest on mapowany bezpośrednio na adres IP, a co za tym idzie klient wpisując nazwę adresu tylko raz wysyła żądanie do serwera DNS o przetłumaczenie na IP. Dodatkowo Route53 posiada funkcje Traffic Flow (przepływ ruchu) takie jak np. Load Balancing geograficzny, czyli usługa, która wykrywa lokalizację klienta i kieruje go do najbliższego serwera czy awaryjne przełączanie (z ang. failover), które zawczasu wykrywa niedziałające serwery oraz wyklucza je.⁹⁶

Uruchomienie usługi DNS polega na utworzeniu nowej strefy dla posiadanej domeny, bez żadnych dodatkowych konfiguracji. Dodatkowo zostały utworzone rekordy A (dla IPv4) i AAAA (dla IPv6) będące aliasami adresu DNS usługi CloudFront. Konfiguracja znajduje się w pliku *Route53.tf*.

W wyniku wygenerowania nowej strefy automatycznie zostały utworzone rekordy NS (z ang. Name Server), które wskazują na adresy, w których domena jest przechowywana. Adresy te należy dodać w panelu konfiguracyjnym usługodawcy, u którego domena została kupiona np. *home.pl* *godaddy.com* itp.

⁹⁵ CNAME - typ rekordu DNS, który mapuje alias na rzeczywistą nazwę domeny

⁹⁶ <https://aws.amazon.com/route53/> Dostęp: 26.05.2018r.

Back to Hosted Zones Create Record Set Import Zone File Delete Record Set Test Record Set				
<input type="text" value="Record Set Name"/> <input type="button" value="X"/> <input type="button" value="Any Type"/> <input type="checkbox"/> Aliases Only <input type="checkbox"/> Weighted Only				
<< < Displaying 1 to 8 out of 8 Record Sets > >>				
<input type="checkbox"/>	Name	Type	Value	Evaluate Target Health
<input type="checkbox"/>	workshop.edu.pl.	A	ALIAS d1pm3n0j2hizam.cloudfront.net. (z2fdtndataq	Yes
<input type="checkbox"/>	www.workshop.edu.pl.	A	ALIAS d1pm3n0j2hizam.cloudfront.net. (z2fdtndataq	Yes
<input type="checkbox"/>	workshop.edu.pl.	AAAA	ALIAS d1pm3n0j2hizam.cloudfront.net. (z2fdtndataq	Yes
<input type="checkbox"/>	www.workshop.edu.pl.	AAAA	ALIAS d1pm3n0j2hizam.cloudfront.net. (z2fdtndataq	Yes
<input type="checkbox"/>	_44af9fad89e2cec72f4e70fe5cbab7ec.workshop.edu.pl.	CNAME	_1f94b906bc893fdf00dc6696e9a2c6a0.acm-validati	-
<input type="checkbox"/>	_4adb557a339ea9e917302b8e312ded3c.www.workshop.edu.pl.	CNAME	_5394f9a7ab6924f2f67661a5e232c28d.acm-validati	-
<input type="checkbox"/>	workshop.edu.pl.	NS	ns-20.awsdns-02.com. ns-577.awsdns-08.net. ns-1276.awsdns-31.org. ns-1828.awsdns-36.co.uk.	-
<input type="checkbox"/>	workshop.edu.pl.	SOA	ns-1276.awsdns-31.org. awsdns-hostmaster.amazoni	-

Rys. 5.1 Zrzut ekranu wygenerowanej strefy DNS

5.2 Certyfikat SSL

Ponieważ łącznie z aplikacją jest domyślnie szyfrowane (ruch z portu 80 jest przekierowywany na port 443) do wdrożenia potrzebny jest również certyfikat SSL, który pomimo braku informacji o konfiguracji został już wcześniej wykorzystany do zabezpieczenia komunikacji z równoważnikiem ruchu na porcie 443. Warto zaznaczyć, że Amazon nie pobiera opłat z wydawanie certyfikatów, ale ogranicza ich wykorzystanie tylko usług oferowanych w ramach ich infrastruktury.

Utworzenie Certyfikatu ogranicza się do podania nazw domen, które mają być chronione oraz potwierdzenia posiadania domen. Projekt zakłada uruchomienie aplikacji pod adresem *workshop.edu.pl* dlatego objęte certyfikacją będą tylko domena główna oraz subdomena *www.workshop.edu.pl*. Weryfikacji własności może, przebiegać na dwa sposoby:

- Weryfikacja mailowa – zostaje wysłany e-mail z potwierdzeniem na adres *admin@domena*. W wiadomości znajduje się link, za pomocą którego uprawniona osoba potwierdza tożsamość.
- Weryfikacja DNS – zostaje wygenerowany jeden lub więcej (w zależności od ilości domen i subdomen objętych certyfikacją) rekord CNAME, który należy dodać do konfiguracji DNS domeny.

Ponieważ, konfiguracja DNS dla wybranej domeny została ustawiona w ramach AWS to korzystając z drugiego sposobu weryfikacji można to w łatwy sposób zautomatyzować, dlatego też został użyty ten sposób. Całość konfiguracji certyfikatu oraz dodawania rekordów DNS znajduje się w pliku *Certyficate.tf*. Na Rysunku 5.1 zaprezentowany został zrzut ekranu po przeprowadzonej walidacji.

The screenshot displays the AWS Certificate Manager console for a certificate named 'Workshop Cert' issued to 'workshop.edu.pl' and 'www.workshop.edu.pl'. The status is 'Issued'. Below the status, a table shows the validation status for both domains as 'Success'. The 'Details' section provides comprehensive information about the certificate, including its type, domain name, and various identifiers. The 'Tags' section at the bottom shows an 'Edit' button and a tag named 'Workshop Cert'.

Domain	Validation status
workshop.edu.pl	Success
www.workshop.edu.pl	Success

Property	Value
Type	Amazon Issued
In use?	No
Domain name	workshop.edu.pl
Number of additional names	1
Additional names	www.workshop.edu.pl
Identifier	d2c661d5-73c4-42d9-935e-4daaeacfb680
Serial number	08:c1:db:9a:99:28:c7:ec:ca:16:84:06:b6:15:3c:cb
Requested at	2018-06-15T11:14:33UTC
Issued at	2018-06-15T12:21:05UTC
Not before	2018-06-15T00:00:00UTC
Not after	2019-07-15T12:00:00UTC
Public key info	RSA 2048-bit
Signature algorithm	SHA256WITHRSA
ARN	arn:aws:acm:us-east-1:019164397105:certificate/d2c661d5-73c4-42d9-935e-4daaeacfb680
Validation state	None

Rys. 5.2 Wynik poprawnej konfiguracji Certyfikatu

5.3 Globalna sieć dostarczania treści Amazon CloudFront

Amazon CloudFront jest to rozproszona, wysoko dostępna, globalna sieć dostarczania treści CDN (z ang. Content Delivery Network) zapewniająca bezpieczne przesyłanie danych. Cechą charakterystyczną usługi są niskie opóźnienia oraz dużych prędkościach przesyłania. Głównym zadaniem każdej usługi CDN, w tym CloudFront, jest buforowanie treści dzięki czemu serwery produkcyjne są odciążone od serwowania statycznych treści. Ponadto CloudFront wspiera usługę Lambda@Edge, za pomocą której można uruchamiać kod blisko użytkownika. CloudFront jest usługą w pełni zintegrowaną z AWS, co oznacza, że lokalizacje fizyczne, pomimo że nie są umieszczone bezpośrednio w centrach danych to są bezpośrednio połączone z globalną infrastrukturą AWS.⁹⁷

⁹⁷ <https://aws.amazon.com/cloudfront/> Dostęp: 26.05.2018r.

Podstawowe uruchomienie usługi CloudFront wymaga skonfigurowania źródła danych oraz domyślnego zachowania bufora (cache). W przypadku wdrażanej aplikacji, źródłem danych będzie load balancer aplikacyjny, do którego usługa będzie łączyć się po porcie HTTPS. W przypadku buforowania, należy uwzględnić, fakt, iż w aplikacji WordPress dużo pracy odbywa się po stronie serwera, także domyślnym zachowaniem będzie wyłącznie buforowanie. Jednak, aby nie stracić najważniejszej funkcji systemu, zostały utworzone dodatkowe ustawienia dla lokalizacji /wp-content/ oraz /wp-include/ w których WordPress domyślnie przechowuje dane statyczne. Domyślnym zachowaniem dystrybucji jest przekierowywanie adresu z protokołu HTTP na HTTPS oraz wspieranie kompresji danych. Zabiegi te pomagają użytkownikowi końcowemu zwiększyć prędkość pobieranych danych oraz zapewniają bezpieczeństwo przekazywanych informacji. Dystrybucja została również skonfigurowana w taki sposób, by nikt nie mógł się pod nią podszyć i wspiera tylko adresy domenowe *workshop.edu.pl* oraz wariant *www.workszop.edu.pl*. Dodatkowo wszystkie zapytania i odpowiedzi do dystrybucji są logowane. W tym celu został utworzony zasób Amazon S3⁹⁸ w którym system CloudFront będzie zapisywał wszystkie logi. Pełna konfiguracja dystrybucji znajduje się w pliku *Cloudfront.tf*.

⁹⁸ Amazon S3 (Amazon Simple Storage Service) – usługa przechowywania danych a AWS.

Zakończenie

Założeniem niniejszej pracy było opracowanie i wdrożenie pełnej infrastruktury zapewniającej pracę aplikacji webowej w trybie wysokiej dokładności w oparciu o chmurę Amazon Web Services. Stworzone środowisko spełnia wszystkie postawione wymagania.

W pierwszych dwóch rozdziałach pracy zostały zaprezentowane podstawowe pojęcia związane z przetwarzaniem w chmurze oraz przedstawione narzędzia wspomagające pracę administratora w środowisku Amazon Web Services. Kolejne części to dokumentacja kodu konfiguracji potrzebnej do uruchomienia pełnej infrastruktury za pomocą narzędzia Terraform. Narzędzia tego typu w znacznym stopniu ułatwiają pracę administratorów zarządzających infrastrukturami w chmurze. Po pierwsze przyspieszają pracę – w przypadku chmur publicznych takich jak Amazon Web Services, podstawowe zarządzanie przez przeglądarkę internetową jest czasochłonne i w wielu przypadkach nieintuicyjne. Przechowywanie konfiguracji infrastruktury w postaci kodu zapewnia przejrzystość ustawień, gdyż nie trzeba przełączać się pomiędzy kartami konfiguracyjnymi. Dodatkowo narzędzia te pozwalają na pracę w tak zwanym trybie offline⁹⁹ – kod można przygotować wcześniej i uruchomić np. po konsultacji z inną osobą techniczną lub po uzyskaniu dostępu do Internetu. Kolejnym istotną cechą jest fakt, że dla administratorów, którzy pracują dla różnych klientów raz napisany kod infrastruktury pozwala zaoszczędzić czas i wysiłek włożony w powtarzalność niektórych zadań – wystarczy zmodyfikować kod z wcześniej przygotowanego projektu.

⁹⁹ Offline – bez dostępu do Internetu

Bibliografia

- [1] D. Jilk, „Keys to the PaaS Game: Multi-Tenancy” 27 Listopad 2011. [Online]. Dostęp: <http://cloudcomputing.sys-con.com/node/2073281>
- [2] D. K. Barry, „Service-Oriented Architecture (SOA) Definition” [Online]. Dostęp: https://www.service-architecture.com/articles/web-services/service-oriented_architecture_soa_definition.html
- [3] L. M. Vaquero, L. Roderio-Merino, J. Caceres i M. Lindner, „A break in the clouds: Towards a cloud definition” Styczeń 2009. [Online]. Dostęp: <http://ccr.sigcomm.org/online/files/p50-v39n11-vaqueroA.pdf>
- [4] P. Mell i T. Grance, „The NIST Definition of Cloud Computing” Wrzesień 2011. [Online]. Dostęp: <https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-145.pdf>
- [5] P. Srisuresh i M. Holdrege, „RFC 2663 IP Network Address Translator (NAT) Terminology and Considerations” Sierpień 1999. [Online]. Dostęp: <https://tools.ietf.org/html/rfc2663>
- [6] Y. Rekhter, B. Moskowitz, D. Karrenberg, G. J. de Groot i E. Lear, „RFC 1918 Address Allocation for Private Internets” Luty 1996. [Online]. Dostęp: <https://tools.ietf.org/html/rfc1918>
- [7] „Microsoft Azure vs AWS: Cloud Comparison Guide [2018 Update],” Marzec 2018. [Online]. Dostęp: <https://www.redpixie.com/blog/microsoft-azure-aws-guide>
- [8] „Oficjalna dokumentacja Ansible” 2018. [Online]. Dostęp: <http://docs.ansible.com/ansible/latest/index.html>
- [9] „Oficjalna dokumentacja JSON,” [Online]. Dostęp: <https://www.json.org/json-pl.html>.
- [10] „Oficjalna dokumentacja oraz kod źródłowy HCL” [Online]. Dostęp: <https://github.com/hashicorp/hcl>
- [11] „Oficjalna dokumentacja Terraform” 2018. [Online]. Dostęp: <https://www.terraform.io/docs/index.html>
- [12] „Oficjalna strona Amazon Web Services” 2018. [Online]. Dostęp: <https://aws.amazon.com/>

Spis ilustracji

Rys. 1.1 Modele i cechy przetwarzania w chmurze	12
Rys. 1.2 Architektura chmury	13
Rys. 1.3 Modele wdrażania chmury	16
Rys. 1.4 Porównanie modeli chmury	19
Rys. 2.1 Światowi dostawcy IaaS [7]	20
Rys. 2.2 AWS Managment Console	22
Rys. 3.1 Architektura aplikacji	25
Rys. 3.2 Dodawanie użytkownika IAM	27
Rys. 3.3 Przypisywanie uprawnień użytkownikowi IAM	27
Rys. 3.4 Zrzut z AWS Web Management po zbudowaniu VPC	31
Rys. 3.5 Przykładowy wynik komendy terraform apply	32
Rys. 3.6 Struktura podsieci wraz z bramami dostępowymi	33
Rys. 3.7 Dodawanie wpisu do Grupy Zabezpieczeń w paneli AWS	36
Rys. 4.1 Informacje końcowe aplikacji	46
Rys. 4.2 Konfiguracja aplikacji WordPress	47
Rys. 5.1 Zrzut ekranu wygenerowanej strefy DNS	49
Rys. 5.2 Wynik poprawnej konfiguracji Certyfikatu	50

Zawartość CD

1. Praca dyplomowa w formacie DOCX
2. Praca dyplomowa w formacie PDF
3. Pliki źródłowe wdrażanej Infrastruktury
 - 3.1. Backend_AS.tf
 - 3.2. Backend_LC.tf
 - 3.3. backend-user_data.yml
 - 3.4. Backend.tf
 - 3.5. bastion-user_data.yml
 - 3.6. Bastion.tf
 - 3.7. Certificate.tf
 - 3.8. Cloudfront.tf
 - 3.9. credentials.tf
 - 3.10. EFS.tf
 - 3.11. ELB.tf
 - 3.12. Gateways.tf
 - 3.13. NAT_Gateways.tf
 - 3.14. NAT_RouteTables.tf
 - 3.15. output.tf
 - 3.16. RDS.tf
 - 3.17. Route53.tf
 - 3.18. RouteTables.tf
 - 3.19. terraform.tfvars
 - 3.20. VPC.tf
 - 3.21. workshop-backend
 - 3.22. workshop-backend.pub
 - 3.23. workshop-bastion
 - 3.24. workshop-bastion.pub

Tytuł pracy:

Projekt i implementacja infrastruktury chmury obliczeniowej zapewniającej pracę aplikacji w trybie wysokiej dostępności z wykorzystaniem Amazon Web Services

Streszczenie:

W pracy przedstawiono implementację infrastruktury w chmurze obliczeniowej Amazon Web Services pod aplikację internetową WordPress w trybie wysokiej dostępności. Infrastruktura została przedstawiona jako kod współpracujący z aplikacją Terraform. Zaprezentowane zostały również podstawowe zagadnienia związane z chmurą obliczeniową oraz narzędzia pracy administratora.

Tytuł w języku angielskim:

Design and implementation of cloud computing infrastructure for high availability web application in Amazon Web Services

Streszczenie w języku angielskim:

This thesis presents the implementation of the infrastructure in the Amazon Web Services cloud for the WordPress web application in the high availability mode. The infrastructure has been presented as a code cooperating with the Terraform application. Also this work presented the basic issues related to cloud computing and administrator's work tools.