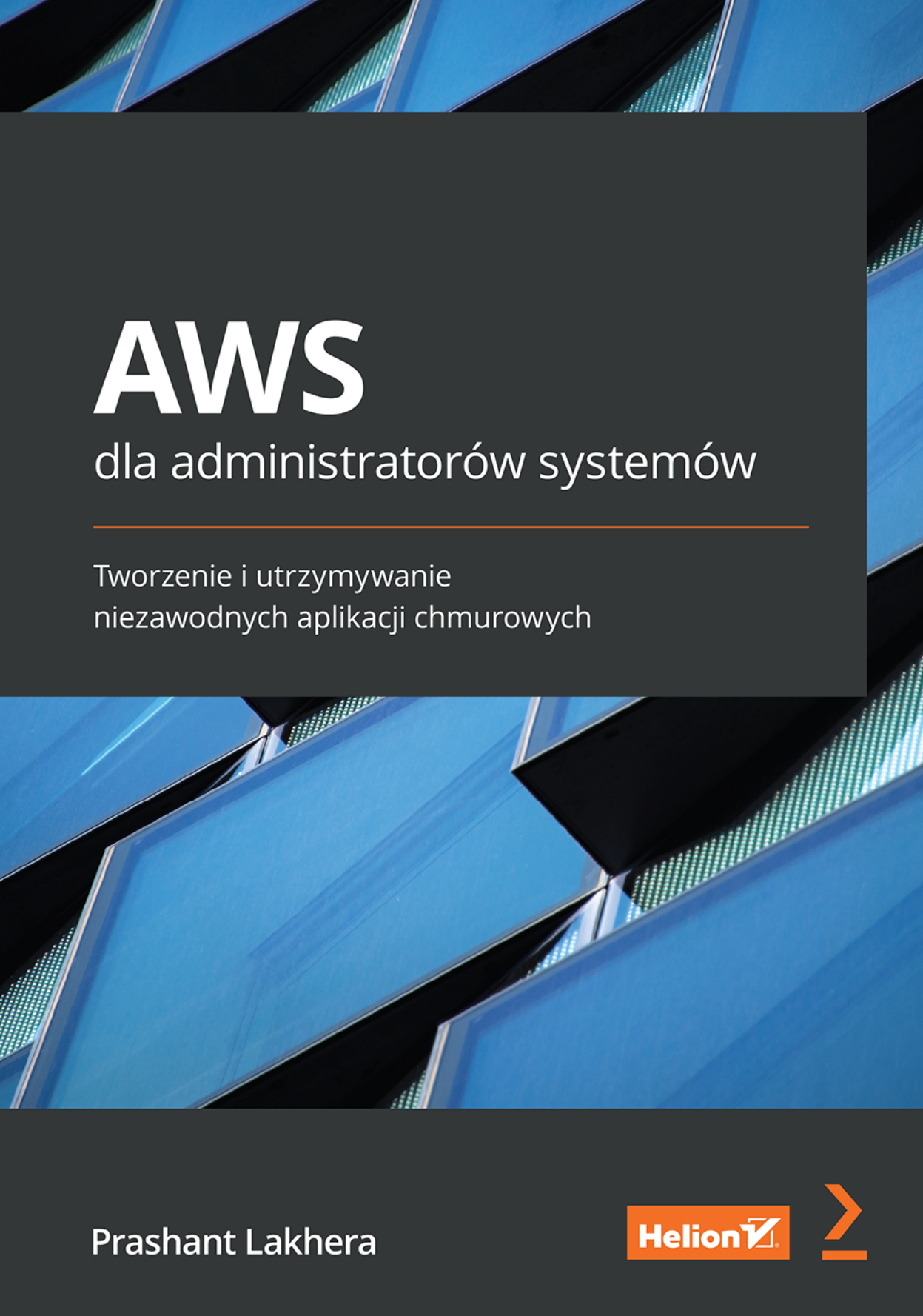




AWS

dla administratorów systemów

Tworzenie i utrzymywanie
niezawodnych aplikacji chmurowych

Prashant Lakhera

Helion 



Tytuł oryginału: AWS for System Administrators: Build, automate, and manage your infrastructure on the most popular cloud platform – AWS

Tłumaczenie: Krzysztof Bąbol
Korekta językowa: Expressis Verbis

ISBN: 978-83-283-9657-9

Copyright © Packt Publishing 2021. First published in the English language under the title 'AWS for System Administrators' – (9781800201538)

Polish edition copyright © 2023 by Helion S.A.

All rights reserved. No part of this book may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying, recording or by any information storage retrieval system, without permission from the Publisher.

Wszelkie prawa zastrzeżone. Nieautoryzowane rozpowszechnianie całości lub fragmentu niniejszej publikacji w jakiegokolwiek postaci jest zabronione. Wykonywanie kopii metodą kserograficzną, fotograficzną, a także kopiowanie książki na nośniku filmowym, magnetycznym lub innym powoduje naruszenie praw autorskich niniejszej publikacji.

Wszystkie znaki występujące w tekście są zastrzeżonymi znakami firmowymi bądź towarowymi ich właścicieli.

Autor oraz wydawca dołożyli wszelkich starań, by zawarte w tej książce informacje były kompletne i rzetelne. Nie biorą jednak żadnej odpowiedzialności ani za ich wykorzystanie, ani za związane z tym ewentualne naruszenie praw patentowych lub autorskich. Autor oraz wydawca nie ponoszą również żadnej odpowiedzialności za ewentualne szkody wynikłe z wykorzystania informacji zawartych w książce.

Drogi Czytelniku!

Jeżeli chcesz ocenić tę książkę, zajrzyj pod adres

<https://helion.pl/user/opinie/awsadm>

Możesz tam wpisać swoje uwagi, spostrzeżenia, recenzję.

Helion S.A.

ul. Kościuszki 1c, 44-100 Gliwice

tel. 32 230 98 63

e-mail: helion@helion.pl

WWW: <https://helion.pl> (księgarnia internetowa, katalog książek)

Printed in Poland.

- Kup książkę
- Poleć książkę
- Oceń książkę

- Księgarnia internetowa
- Lubię to! » Nasza społeczność

Spis treści

O autorze	7
O recenzencie	8
Wstęp	9
Część I. Usługi i narzędzia platformy AWS	
Rozdział 1. Konfigurowanie środowiska AWS	15
Wymagania techniczne	16
Konfigurowanie środowiska	16
Instalowanie programu AWS CLI	20
Konfigurowanie uzupełniania wiersza poleceń	20
Konfigurowanie wiersza poleceń platformy AWS	21
Poznanie struktury poleceń programu AWS CLI	22
Prezentacja zestawu Boto3 dla języka Python	23
Instalowanie zestawu Boto3 dla języka Python	23
Weryfikowanie poprawności instalacji zestawu Boto3	23
Wprowadzenie do korzystania z usługi CloudFormation	24
Pisanie pierwszego szablonu CloudFormation	25
Tworzenie stosu CloudFormation przy użyciu konsoli AWS	27
Tworzenie stosu CloudFormation w programie AWS CLI	29
Wprowadzenie do korzystania z narzędzia Terraform	30
Instalowanie narzędzia Terraform	31
Tworzenie zasobów przy użyciu narzędzia Terraform	32
Instalowanie narzędzi w sposób zautomatyzowany	34
Podsumowanie	35

Rozdział 2. Ochrona konta AWS przy użyciu usługi IAM	36
Wymagania techniczne	37
Dodawanie użytkowników i tworzenie grup IAM	37
Użytkownicy IAM	37
Grupy IAM	38
Zapoznanie z zasadami IAM	40
Struktura zasady IAM	40
Identyfikator ARN	42
Ocena zasad IAM	42
Tworzenie zasad IAM w programie AWS CLI	43
Tworzenie ról IAM	45
Zalety stosowania ról IAM	45
Tworzenie roli IAM w programie Terraform	45
Usługa AWS Security Token Service (AWS STS)	48
Zalety usługi AWS STS	49
Przykłady użycia	49
Dostęp do usług na innym koncie z użyciem ról IAM	49
Praktyczny przykład uruchamiania danej instancji za pomocą usługi CloudFormation	56
Wymiana poświadczeń IAM przy użyciu zestawu Boto3	58
Wymagania wstępne	58
Tworzenie skryptu Boto3 do wymiany poświadczeń	59
Podsumowanie	61

Część II. Budowanie infrastruktury

Rozdział 3. Tworzenie centrum danych w chmurze przy użyciu usługi VPC	65
Wymagania techniczne	66
Konfigurowanie dwóch sieci VPC	66
Tworzenie pierwszej sieci VPC w konsoli AWS	67
Tworzenie drugiej sieci VPC przy użyciu usługi CloudFormation	85
Brama tranzytowa AWS	90
Tworzenie pierwszej bramy tranzytowej w konsoli AWS	92
Tworzenie drugiej bramy tranzytowej za pomocą narzędzia Terraform	97
Praktyczny przykład włączania dziennika przepływu VPC	99
Podsumowanie	110
Rozdział 4. Skalowalna moc obliczeniowa w chmurze z wykorzystaniem usługi EC2	111
Wymagania techniczne	112
Konfigurowanie instancji EC2	112
Tworzenie pierwszej instancji EC2 w konsoli AWS	112
Tworzenie instancji EC2 przy użyciu usługi AWS CloudFormation	118
Tworzenie w chmurze AWS alarmów dotyczących rozliczeń	120
Praktyczny przykład sprzątania nieużywanych obrazów AMI	128
Praktyczny przykład usuwania nieużywanych woluminów EBS	137
Praktyczny przykład codziennego wyłączania instancji	141
Podsumowanie	145

Część III. Nadawanie infrastrukturze skalowalności i elastyczności

Rozdział 5. Uodparnianie aplikacji na awarie dzięki usłudze Elastic Load Balancing	149
Wymagania techniczne	150
Różne moduły równoważenia obciążenia oferowane przez AWS	150
Konfigurowanie modułu równoważenia obciążenia aplikacji	151
Utworzenie modułu równoważenia obciążenia aplikacji	154
Automatyzowanie tworzenia modułu równoważenia obciążenia aplikacji przy użyciu narzędzia Terraform	163
Podsumowanie	168
Rozdział 6. Zwiększanie wydajności aplikacji dzięki usłudze AWS Auto Scaling	169
Wymagania techniczne	170
Konfigurowanie usługi Auto Scaling	170
Tworzenie szablonu uruchamiania	170
Tworzenie grupy AWS Auto Scaling	174
Weryfikowanie działania grupy automatycznego skalowania	180
Zasady usługi Auto Scaling	181
Skalowanie aplikacji w zależności od popytu	182
Testowanie grupy automatycznego skalowania	193
Tworzenie grupy automatycznego skalowania w narzędziu Terraform	194
Podsumowanie	195
Rozdział 7. Tworzenie relacyjnej bazy danych w chmurze przy użyciu usługi AWS Relational Database Service (RDS)	197
Wymagania techniczne	198
Różnorodność baz danych oferowanych w usłudze AWS RDS	198
Konfigurowanie usługi AWS RDS w trybie wysokiej dostępności	199
Konfigurowanie repliki do odczytu dla bazy danych MySQL	209
Automatyzowanie tworzenia bazy danych MySQL w usłudze AWS RDS przy użyciu narzędzia Terraform	210
Podsumowanie	214

Część IV. Warstwy monitorowania, wskaźników i kopii zapasowych

Rozdział 8. Monitorowanie usług AWS przy użyciu rozwiązań CloudWatch i SNS	217
Wymagania techniczne	218
Monitorowanie w usłudze CloudWatch	218
Monitorowanie w usłudze CloudWatch wskaźników niestandardowych	219
Pobieranie i instalowanie agenta CloudWatch	219
Tworzenie roli IAM używanej przez agenta CloudWatch	220
Uruchamianie agenta CloudWatch na serwerze	223
Usługa SNS	228
Usługa CloudWatch Events	231

Automatyzowanie powiadomień o alarmach przy użyciu wiadomości e-mail i kanału Slack	234
Konfigurowanie platformy Slack	234
Konfigurowanie usługi CloudWatch	237
Tworzenie funkcji Lambda	242
Testowanie integracji	245
Podsumowanie	248
Rozdział 9. Centralizowanie dzienników w celu ich analizowania	249
Wymagania techniczne	250
Dlaczego musimy zarządzać dziennikami?	250
Konfigurowanie agenta CloudWatch	251
Konfigurowanie usługi AWS Elasticsearch i narzędzia Kibana	258
Podsumowanie	264
Rozdział 10. Rozwiązanie centralizujące tworzenie kopii zapasowych w chmurze	265
Wymagania techniczne	266
Różne sposoby tworzenia kopii zapasowych oferowane przez AWS	266
Dlaczego tworzymy zapasowe kopie danych?	266
Konfigurowanie usługi AWS DLM	268
Tworzenie kopii zapasowej danych w usłudze S3 przy użyciu programu AWS CLI	272
Przenoszenie danych w usłudze S3 do klasy pamięci Glacier za pomocą zasad cyklu działania	274
Automatyzowanie przenoszenia danych z S3 do usługi Glacier w narzędziu Terraform	278
Podsumowanie	281
Rozdział 11. Odzyskiwanie awaryjne w chmurze AWS	282
Wymagania techniczne	283
Rozwiązania w zakresie odzyskiwania awaryjnego oferowane przez AWS	283
Kopie zapasowe i przywracanie danych	283
Płomyk dyżurny	284
Rezerwa ciepła w chmurze AWS	285
Rezerwa gorąca (w wielu lokacjach)	286
Konfigurowanie witryny internetowej tak, by w razie awarii przełączała się na wiadro S3	297
Podsumowanie	304
Rozdział 12. Porady i wskazówki dotyczące korzystania z platformy AWS	306
Wymagania techniczne	307
Typowe pułapki — ograniczenia usługi VPC	307
Jakie podsieci wybrać podczas budowania sieci VPC?	308
Dedykowana instancja czy dedykowany host — co wybrać?	309
Potencjał granicy uprawnień IAM	310
Niestandardowe wskaźniki CloudWatch	311
Tagowanie, ciągłe tagowanie — dlaczego to ważne?	312
Ochrona instancji EC2 i woluminów EBS przed zamknięciem	313
Jak zmniejszyć rachunki za korzystanie z chmury AWS?	314
Wybór nazwy wiadra AWS i jak sprawić, by była losowa	315
Automatyzowanie tworzenia obrazów AMI	315
Tworzenie obrazu AMI w konsoli AWS	316
Tworzenie obrazu AMI w programie AWS CLI	317
Automatyzowanie tworzenia obrazu AMI przy użyciu narzędzia Packer	317
Podsumowanie	319

Skalowalna moc obliczeniowa w chmurze z wykorzystaniem usługi EC2

W poprzednim rozdziale pokazano, jak skonfigurować sieć w usłudze VPC, która działa tak jak własne centrum danych w chmurze. Komponent sieciowy jest już gotowy. Następnym krokiem będzie utworzenie maszyn wirtualnych, w terminologii AWS zwanych także instancjami, w których można umieścić kod.

Usługa AWS EC2, która zapewnia skalowalną moc obliczeniową w chmurze, jest jednym z najbardziej znanych komponentów platformy AWS. Dzięki niej nie trzeba wcześniej inwestować w sprzęt i można szybciej opracować, a także wdrożyć aplikację w chmurze. Pozwala na skalowanie stosownie do wymagań w górę i w dół w celu obsłużenia zmiennej liczby żądań, także jej chwilowego wzrostu.

Rozdział ten rozpocznie się opisem konfigurowania instancji EC2 w konsoli AWS. Ręczne konfigurowanie jest pracochłonne, dowiecie się więc, jak zautomatyzować ten proces za pomocą usługi CloudFormation. Przyjrzymy się też jednemu z najistotniejszych zadań — zarządzaniu rozliczeniami na platformie AWS — i ustawimy stosowny alarm. Zagadnienia związane z oszczędzaniem na rachunkach za korzystanie z platformy przeanalizujemy na trzech praktycznych przykładach: wyłączania nieobciążonych instancji, czyszczenia niepotrzebnych obrazów **Amazon Machine Image (AMI)** i usuwania nieużywanych woluminów EBS.

W tym rozdziale:

- konfigurowanie instancji EC2,
- tworzenie alarmów dotyczących rozliczeń w chmurze AWS,
- praktyczny przykład sprzątania nieużywanych obrazów AMI,
- praktyczny przykład usuwania nieużywanych woluminów EBS,
- praktyczny przykład codziennego wyłączania instancji.

Wymagania techniczne

Aby jak najwięcej wynieść z lektury tego rozdziału, należy mieć podstawową wiedzę o usługach AWS. Trzeba znać pojęcia takie jak hipernadzorca (ang. *hypervisor*), maszyna wirtualna, elastyczny magazyn blokowy (ang. *elastic block storage*) czy **Amazon Machine Image (AMI)**, obraz maszyny w chmurze Amazon). Poza tym konieczna jest podstawowa znajomość narzędzi CloudFormation i Terraform, które zostały już omówione w rozdziale 1., „Konfigurowanie środowiska AWS”.

Pliki z przykładowym kodem znajdziesz w archiwum dostępnym pod adresem <https://ftp.helion.pl/przyklady/awsadm.zip>, w folderze *r04*.

Konfigurowanie instancji EC2

Elastic Compute Cloud (EC2, elastyczna chmura obliczeniowa) jest to maszyna wirtualna działająca w chmurze. Aby móc z niej korzystać, nie trzeba jednak wydawać kilku tysięcy złotych — opłaty uiszcza się na bieżąco (ang. *pay-as-you-go*). W tym modelu płaci się tylko za czas, w którym zasób jest używany.

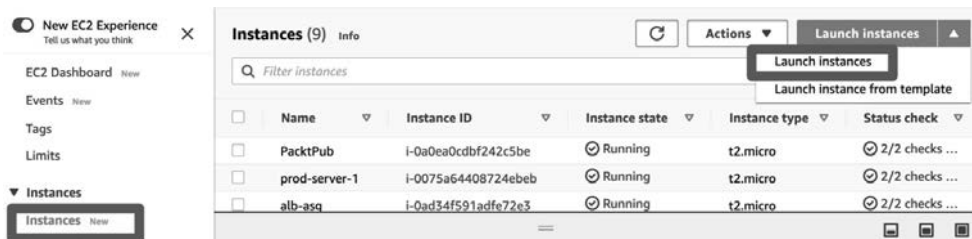
Kolejną zaletą korzystania z chmury jest to, że można w zależności od wymagań łatwo przejść na korzystanie z zasobów wyższej lub niższej klasy. Jeśli na przykład na początku do działania aplikacji wystarczy instancja o jednym procesorze i 1 GB pamięci, to później, kiedy użycie wzrośnie, można przejść (nawet bez żadnych przestojów) na instancję dwuprocesorową z 2 GB pamięci. W tradycyjnym środowisku pozostalibyśmy na zawsze z komputerem o jednym procesorze i 1 GB pamięci, ale w chmurze przejście na większą instancję (2 procesory i 2 GB) jest bardzo proste.

Tworzenie pierwszej instancji EC2 w konsoli AWS

Pierwszym krokiem do skonfigurowania aplikacji jest uruchomienie instancji EC2, w której ta aplikacja zostanie umieszczona, a początkującym najłatwiej zrobić to w konsoli AWS.

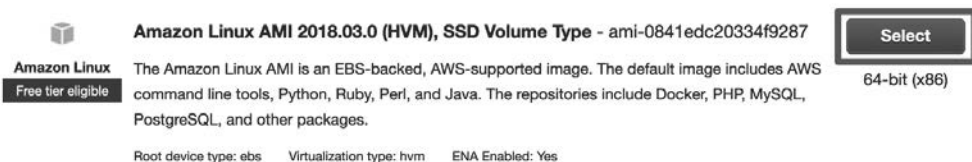
Aby uruchomić instancję w konsoli AWS, musisz wykonać serię kroków:

1. Wejdź do konsoli EC2 na stronę <https://us-west-2.console.aws.amazon.com/ec2/v2/>.
2. Kliknij pozycję *Instances*, a potem przycisk *Launch instances* (uruchom instancję; rysunek 4.1). W górnym prawym rogu następnej strony kliknij przycisk *Opt out to the old experience* (wycofaj się do starego środowiska).



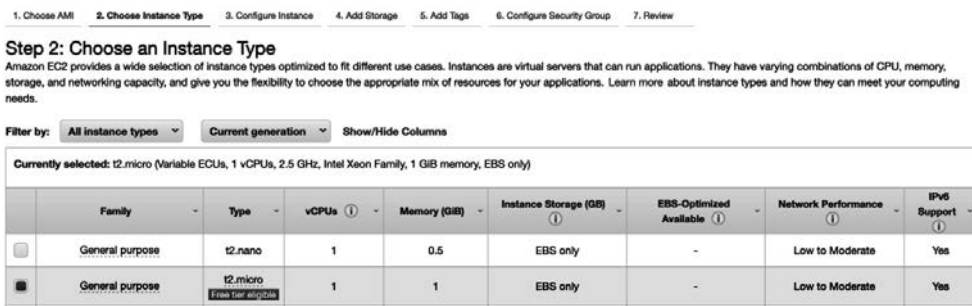
Rysunek 4.1. Uruchamianie pierwszej instancji EC2 w konsoli AWS

3. AMI to obraz ISO systemu operacyjnego i aplikacji. Na potrzeby tej demonstracji na ekranie *Choose an Amazon Machine Image (AMI)* (wybierz obraz AMI) skorzystamy z obrazu AMI dystrybucji Amazon Linux (rysunek 4.2), ale można wybrać dowolny obraz odpowiadający wymaganiom (na przykład CentOS, Red Hat czy Windows).



Rysunek 4.2. Wybieranie obrazu AMI z systemem Amazon Linux

4. Następnym krokiem jest wybór typu instancji. AWS oferuje duży wybór typów, zaspokajający różne wymagania i potrzeby klientów. W tym przykładzie skorzystamy z typu *t2.micro*, w którym do dyspozycji jest jeden wirtualny procesor (ang. *virtual CPU*, *vCPU*) i 1 GiB pamięci (rysunek 4.3).



Rysunek 4.3. Wybór typu instancji

5. W następnej sekcji, *Configure Instance Details* (skonfiguruj szczegóły instancji), wybierzemy tylko kilka parametrów, a pozostałe ustawienia zachowamy jako domyślne (rysunek 4.4). Oto parametry, które musisz podać: z listy rozwijanej *Network* (sieć) wybierz sieć VPC (prod-vpc), utworzoną w rozdziale 3., „Tworzenie centrum danych w chmurze przy użyciu usługi VPC”, z listy rozwijanej *Subnet* — podsieć w strefie dostępności us-west-2a, a w polu tekstowym *User data* (dane użytkownika), pod nagłówkiem *Advanced Details* (zaawansowane szczegóły), wklej skrypt. Znajdziesz go w archiwum dostępnym pod adresem <https://ftp.helion.pl/przyklady/awssadm.zip>, w folderze *r04/html/install_apache.sh*. Zostanie zainstalowany serwer Apache. Skrypt uruchomi go, sklonuje oryginalne repozytorium z serwisu GitHub i skopiuje właściwy folder do głównego katalogu dokumentów serwera Apache. Teraz kliknij przycisk *Next: Add Storage* (dalej: dodaj przestrzeń dyskową).

Step 3: Configure Instance Details

Configure the instance to suit your requirements. You can launch multiple instances from the same AMI, request Spot instances to take advantage of the lower pricing, assign an access management role to the instance, and more.

Number of instances 1 Launch into Auto Scaling Group

Purchasing option ☐ Request Spot instances

Network Create new VPC

Subnet Create new subnet
244 IP Addresses available

Auto-assign Public IP

Placement group ☐ Add instance to placement group

Capacity Reservation

Domain join directory Create new directory

IAM role Create new IAM role

Shutdown behavior

Stop - Hibernate behavior ☐ Enable hibernation as an additional stop behavior

Enable termination protection ☐ Protect against accidental termination

Monitoring ☐ Enable CloudWatch detailed monitoring
Additional charges apply.

Tenancy
Additional charges will apply for dedicated tenancy.

Elastic inference ☐ Add an Elastic Inference accelerator
Additional charges apply.

Credit specification ☐ Unlimited
Additional charges may apply.

File systems Create new file system

▼ Network interfaces

Device	Network Interface	Subnet	Primary IP	Secondary IP addresses	IPv6 IPs
eth0	New network interface	subnet-0ba0a071	Auto-assign	Add IP	Add IP

Add Device

▼ Advanced Details

Metadata accessible

Metadata version

Metadata token response hop limit

User data ☒ As text ☐ As file ☐ Input is already base64 encoded

```
#/bin/bash
yum -y install httpd git
service httpd start
echo "This is coming from default apache page" >> /var/www/html/index.html
cd
git clone https://github.com/PacktPublishing/Mastering-AWS-System-Administration.git
cd Mastering-AWS-System-Administration/Chapter4-Scaleable-compute-capacity-in-the-cloud-via-EC2/html/
cp -avr work /var/www/html/
```

Cancel Previous Review and Launch Next: Add Storage

Rysunek 4.4. Konfigurowanie szczegółów instancji

6. W sekcji *Add Storage* (dodaj przestrzeń dyskową) pozostaw domyślne ustawienia — zachowaj rozmiar woluminu głównego (ang. *root volume*) wynoszący 8 GB (rysunek 4.5). Wolumin główny to miejsce, w którym będzie instalowany system operacyjny z obrazu AMI. Kliknij przycisk *Next: Add Tags* (dalej: dodaj tagi).

Step 4: Add Storage

Your instance will be launched with the following storage device settings. You can attach additional EBS volumes and instance store volumes to your instance, or edit the settings of the root volume. You can also attach additional EBS volumes after launching an instance, but not instance store volumes. Learn more about storage options in Amazon EC2.

Volume Type	Device	Snapshot	Size (GiB)	Volume Type	IOPS	Throughput (MB/s)	Delete on Termination	Encryption
Root	/dev/xvda	snap-039044fa8c365b7e1	8	General Purpose SSD (gp2)	100 / 3000	N/A	☒	Not Encrypted

[Add New Volume](#)

Free tier eligible customers can get up to 30 GiB of EBS General Purpose (SSD) or Magnetic storage. Learn more about free usage tier eligibility and usage restrictions.

[Cancel](#) [Previous](#) [Review and Launch](#) [Next: Add Tags](#)

Rysunek 4.5. Dodawanie przestrzeni dyskowej

7. W sekcji *Add Tags* (dodaj tagi) dodaje się do instancji tagi (rysunek 4.6). Są to metadane (pary klucz-wartość) przydatne do śledzenia zasobów. W polu *Key* (klucz) możesz na przykład wpisać *Name* (nazwa), a w polu *Value* (wartość) — *prod-server*. Kliknij przycisk *Next: Configure Security Group* (dalej: skonfiguruj grupy zabezpieczeń).

Step 5: Add Tags

A tag consists of a case-sensitive key-value pair. For example, you could define a tag with key = Name and value = Webserver. A copy of a tag can be applied to volumes, instances or both. Tags will be applied to all instances and volumes. Learn more about tagging your Amazon EC2 resources.

Key	Value	Instances	Volumes
Name	prod-server	☒	☒

[Add another tag](#) (Up to 50 tags maximum)

[Cancel](#) [Previous](#) [Review and Launch](#) [Next: Configure Security Group](#)

Rysunek 4.6. Dodawanie tagów

8. W sekcji *Configure Security Group* (rysunek 4.7) zachowaj regułę domyślną, która zezwala na ruch *SSH* (port 22), i kliknij przycisk *Add Rule*, by dodać regułę dla ruchu *HTTP* (port 80). Z listy rozwijanej *Type* (typ) wybierz *HTTP*, z listy *Protocol* (protokół) — *TCP*, w polu *Port Range* (zakres portów) wpisz **80**, jako typ źródła (*Source*) wybierz *Custom* (niestandardowe) i podaj adres IP **0.0.0.0/0** (pozwoli to na ruch z dowolnego miejsca). Takie postępowanie jest niezgodne z zasadami bezpieczeństwa. Należy zawsze zezwalać tylko na ruch z konkretnej podsieci lub określonych adresów IP, ale w tym ćwiczeniu pozwolimy na ruch z dowolnego miejsca. Grupa zabezpieczeń działa jak wirtualna zapora sieciowa i zezwala na ruch na podstawie zawartych w niej reguł. Kliknij przycisk *Review and Launch* (przejrzyj i uruchom).

1. Choose AMI 2. Choose Instance Type 3. Configure Instance 4. Add Storage 5. Add Tags 6. Configure Security Group 7. Review

Step 6: Configure Security Group

A security group is a set of firewall rules that control the traffic for your instance. On this page, you can add rules to allow specific traffic to reach your instance. For example, if you want to set up a web server and allow Internet traffic to reach your instance, add rules that allow unrestricted access to the HTTP and HTTPS ports. You can create a new security group or select from an existing one below. Learn more about Amazon EC2 security groups.

Assign a security group: ☒ Create a new security group☐ Select an existing security groupSecurity group name: Description:

Type	Protocol	Port Range	Source	Description
SSH	TCP	22	Custom 0.0.0.0/0	e.g. SSH for Admin Desktop
HTTP	TCP	80	Custom 0.0.0.0/0, ::/0	e.g. SSH for Admin Desktop

Add Rule

Cancel Previous Review and Launch

Rysunek 4.7. Konfigurowanie grupy zabezpieczeń

9. Przejrzyj wszystkie ustawienia. Jeśli konfiguracja jest poprawna, kliknij przycisk *Launch* (uruchom; rysunek 4.8).

Step 7: Review Instance Launch

Please review your instance launch details. You can go back to edit changes for each section. Click **Launch** to assign a key pair to your instance and complete the launch process.



Improve your instances' security. Your security group, vpc-prod-sg, is open to the world.

Your instances may be accessible from any IP address. We recommend that you update your security group rules to allow access from known IP addresses only.

You can also open additional ports in your security group to facilitate access to the application or service you're running, e.g., HTTP (80) for web servers. Edit security groups

▼ AMI Details

Edit AMI



Amazon Linux AMI 2018.03.0 (HVM), SSD Volume Type - ami-067f5c3d5a99edc90



The Amazon Linux AMI is an EBS-backed, AWS-supported image. The default image includes AWS command line tools, Python, Ruby, Perl, and Java. The repositories include Docker, PHP, MySQL, PostgreSQL, and other packages.

Root Device Type: ebs Virtualization type: hvm

▼ Instance Type

Edit instance type

Instance Type	ECUs	vCPUs	Memory (GiB)	Instance Storage (GB)	EBS-Optimized Available	Network Performance
t2.micro	Variable	1	1	EBS only	-	Low to Moderate

▼ Security Groups

Edit security groups

Security group name vpc-prod-sg

Cancel Previous Launch

Rysunek 4.8. Przegląd konfiguracji uruchomieniowej

10. Aby móc się zalogować do nowo utworzonej instancji, będziesz potrzebować pary kluczy. W ostatnim kroku przed jej uruchomieniem (rysunek 4.9) możesz utworzyć taką parę. Nadaj jej sensowną nazwę, na przykład vpc-prod, a potem kliknij przycisk *Download Key Pair* (pobierz parę kluczy). Po jej pobraniu kliknij przycisk *Launch Instances* (uruchom instancje).
11. Zobaczysz ekran taki jak na rysunku 4.10. Kliknij instancję (tu na przykład i-0ea882bccd75ad1de; na Twoim koncie AWS identyfikator będzie inny).

Select an existing key pair or create a new key pair



A key pair consists of a **public key** that AWS stores, and a **private key file** that you store. Together, they allow you to connect to your instance securely. For Windows AMIs, the private key file is required to obtain the password used to log into your instance. For Linux AMIs, the private key file allows you to securely SSH into your instance.

Note: The selected key pair will be added to the set of keys authorized for this instance. Learn more about removing existing key pairs from a public AMI.

Create a new key pair

Key pair name

vpc-prod

Download Key Pair



You have to download the **private key file** (*.pem file) before you can continue. **Store it in a secure and accessible location.** You will not be able to download the file again after it's created.

Cancel

Launch Instances

Rysunek 4.9. Tworzenie pary kluczy

Launch Status



Your instances are now launching

The following instance launches have been initiated: i-0ea882bccd75ad1de [View launch log](#)

Get notified of estimated charges

Create billing alerts to get an email notification when estimated charges on your AWS bill exceed an amount you define (for example, if you exceed the free usage tier).

Rysunek 4.10. Stan uruchamiania

12. Aby zalogować się do tej instancji, będziesz potrzebować klucza prywatnego (utworzonego we wcześniejszym kroku) i publicznego adresu IP instancji (uzyskasz go po kliknięciu instancji, jak wspomniano w poprzednim kroku; rysunek 4.11).

☐	Name	Instance ID	Instance Type	Availability Zone	Instance State	Sta
☒	prod-server	i-0594353aca4ca68e6	t2.micro	us-west-2a	running	
☐	PacktPub	i-0a0ea0cdbf242c5be	t2.micro	us-west-2b	running	

Instance: **i-0594353aca4ca68e6 (prod-server)** Public IP: 54.185.177.217

...



Rysunek 4.11. Uzyskiwanie publicznego adresu IP instancji

13. Przed zalogowaniem się do instancji w celu poprawy bezpieczeństwa klucza musisz zmienić wartość jego uprawnień dostępu na 400 (co oznacza, że prawo do jego odczytu będzie miał tylko użytkownik). Jest to ten sam klucz, który został pobrany w kroku 10. Z zasady powinien się znajdować w lokalizacji pobranych plików:

```
chmod 400 vpc-prod.pem
```

14. Aby zalogować się do instancji, potrzebujesz klucza (utworzonego tak, jak pokazano na rysunku 4.9) i nazwy użytkownika właściwej dla obrazu AMI. W przypadku używanego w tym przykładzie obrazu AWS Linux tą nazwą jest `ec2-user`. Różni się ona u poszczególnych dostawców systemów operacyjnych, na przykład w dystrybucji CentOS i Ubuntu nazwy użytkownika to odpowiednio `centos` i `ubuntu`. Aby zalogować się do instancji, podaj klucz, nazwę użytkownika i publiczny adres IP:

```
ssh -i <nazwa_klucza> <nazwa_uzytkownika>@<publiczny_adres_IP>
ssh -i vpc-prod.pem ec2-user@54.185.177.217
```

Wicie już, jak tworzyć instancje w konsoli AWS. Jest to wygodne, gdy chce się uruchomić tylko kilka instancji, ale jeśli ma ich być wiele, lepiej to zrobić w sposób zautomatyzowany — i właśnie wtedy przydaje się usługa AWS CloudFormation. W następnym punkcie zobaczycie, jak wykonać to samo zadanie uruchomienia instancji za pomocą tej usługi.

Tworzenie instancji EC2 przy użyciu usługi AWS CloudFormation

W tym punkcie nauczycie się, jak zautomatyzować proces tworzenia instancji EC2 za pomocą usługi CloudFormation. Przeanalizujemy kod CloudFormation i sprawdzimy krok po kroku, jak on działa:

1. Najpierw musisz utworzyć grupę zabezpieczeń działającą jak wirtualna zapora sieciowa, która będzie zezwalała na ruch z dowolnego miejsca na porcie 22 (SSH). W tym przykładzie użyty zostanie zasób `AWS::EC2::SecurityGroup`, port 22 i protokół `tcp`. Wartość `CidrIp (0.0.0.0)` wskazuje, że dozwolony jest ruch z dowolnego miejsca, co przydaje się w warunkach demonstracyjnych, ale w środowisku produkcyjnym zezwalaj wyłącznie na ruch z określonych zakresów podsieci. W ostatniej sekcji w celu zapewnienia bezpieczeństwa w istniejącej sieci VPC zostanie utworzona grupa zabezpieczeń. Wartość identyfikatora bieżącej sieci VPC trzeba będzie zaimportować, używając funkcji `ImportValue`:

```
Description: Tworzenie instancji EC2 w usłudze CloudFormation
Parameters:
  NetworkStack:
    Type: "String"
    Description: "Tworzenie stosu sieciowego na zasoby"
Resources:
  SecurityGroupptoallowsshtraffic:
```

```
Type: AWS::EC2::SecurityGroup
Properties:
  GroupName: prod-sg
  SecurityGroupIngress:
    - IpProtocol: tcp
      FromPort: 22
      ToPort: 22
      CidrIp: 0.0.0.0/0
  Description: W celu zezwalania na ruch ssh z dowolnego miejsca
  GroupDescription: Grupa zabezpieczen dla srodowiska produkcyjnego
  VpcId:
    Fn::ImportValue:
      !Sub ${NetworkStack}-VpcId
```

2. Grupa zabezpieczeń jest gotowa. Następnym krokiem będzie utworzenie instancji typu `AWS::EC2::Instance` i podanie wszystkich wymaganych parametrów, takich jak: strefa dostępności, wolumin główny, typ obrazu AMI, nazwa klucza (wszystkie te parametry zostały omówione w poprzednim punkcie). Tak jak wcześniej wartość podsieci trzeba zaimportować z istniejącej sieci VPC, należy także podać referencję do utworzonej w poprzednim kroku grupy zabezpieczeń:

```
EC2InstanceProdEnv:
  Type: AWS::EC2::Instance
  Properties:
    AvailabilityZone: "us-east-2a"
    BlockDeviceMappings:
      - DeviceName: "/dev/sda1"
        Ebs:
          DeleteOnTermination: 'true'
          VolumeSize: '8'
          VolumeType: gp2
    ImageId: "ami-07c8bc5c1ce9598c3"
    InstanceType: "t2.micro"

    NetworkInterfaces:
      - Description: "Podstawowy interfejs sieciowy"
        DeviceIndex: "0"
        SubnetId:
          Fn::ImportValue:
            !Sub ${NetworkStack}-PublicsubnetA
        GroupSet:
          - Ref: SecurityGroupptoallowsshtraffic
```

Rozumiecie już, jak jest zbudowany kod CloudFormation. Następnym krokiem będzie utworzenie stosu, w którym powstanie faktyczny zasób chmury AWS:

1. W pierwszym kroku utwórz plik o nazwie *ec2-instance.yml* i wklej do niego powyższy kod CloudFormation. Cały kod można znaleźć w archiwum dostępnym pod adresem <https://ftp.helion.pl/przyklady/awssadm.zip>, w pliku *r04/cloudformation/ec2-instance.yml*. Aby utworzyć te zasoby w regionie Ohio (us-east-2), trzeba wyeksportować region w wierszu poleceń systemu Linux:

```
export AWS_DEFAULT_REGION=us-east-2
```

2. Sprawdź poprawność szablonu, aby upewnić się, że nie zawiera błędów składni. W tym celu musisz wydać polecenie `validate-template`, które zweryfikuje plik `ec2-instance.yml`:

```
aws cloudformation validate-template --template-body file://ec2-instance.yml
{
  "Parameters": [
    {
      "ParameterKey": "NetworkStack",
      "NoEcho": false,
      "Description": "Tworzenie stosu sieciowego na zasoby"
    }
  ],
  "Description": " Tworzenie instancji EC2 w usłudze CloudFormation"
}
```

3. Aby utworzyć stos o nazwie `ec2-dr`, wydaj polecenie `create-stack`:

```
aws cloudformation create-stack --stack-name ec2-dr --template-body
file://ec2-instance.yml --parameters
ParameterKey=NetworkStack,ParameterValue=vpc-dr
{
  "StackId": "arn:aws:cloudformation:us-east-2:XXXXXXX:stack/ec2-
dr/263d0d50-d393-11ea-ada1-06eba8d6f1ae"
}
```

4. Aby zweryfikować, czy instancja została poprawnie utworzona, przejdź do konsoli EC2 (<https://us-east-2.console.aws.amazon.com/ec2/v2/home?region=useast-2#Instances:sort=instanceId>), pokazanej na rysunku 4.12, i sprawdzaj stan instancji w karcie *Status Checks* (kontrola stanu) — w ciągu kilku minut stan *Initializing* (inicjowana) powinien zmienić się na *Running* (uruchomiona).



Rysunek 4.12. Konsola instancji AWS EC2

Teraz wiecie już, jak utworzyć instancję EC2 w konsoli AWS, a także jak zautomatyzować ten proces przy użyciu usługi CloudFormation.

Tworzenie w chmurze AWS alarmów dotyczących rozliczeń

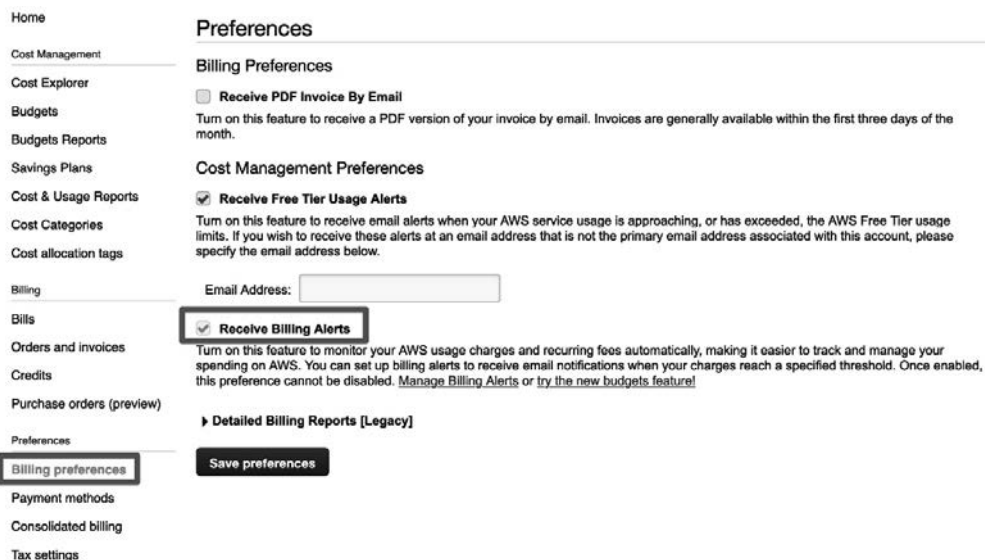
Jednym z najistotniejszych zadań administratora systemów i inżyniera DevOps jest tworzenie alarmu dotyczącego rozliczeń (ang. *billing alarm*), po to by zespół był powiadamiany o przekroczeniu pewnego progu opłat. Może się on też przydać, jeśli ktoś zapomni zamknąć używane

zasoby AWS. W takich przypadkach po przekroczeniu wartości progowej zostanie wyzwołany alarm dotyczący rozliczeń, działający niczym kontrola bezpieczeństwa. Można wtedy wrócić na konkretne konto i zobaczyć, które zasoby AWS trzeba wyłączyć lub uprzątnąć, by ograniczyć koszty. Aby utworzyć alarm dotyczący rozliczeń, trzeba spełnić pewne wymagania wstępne:

- Musisz się zalogować jak użytkownik główny (ang. *root user*) albo użytkownik IAM mający uprawnienia do przeglądania informacji rozliczeniowych.
- W konsoli AWS musisz wybrać region *us-east (N. Virginia)*, bo w tym regionie przechowywane są dane pomiarowe dotyczące rozliczeń.
- Musisz włączyć opcję *Receive Billing Alerts* (otrzymuj alerty dotyczące rozliczeń), co oznacza, że gdy wysokość opłat przekroczy pewien próg, otrzymasz powiadomienie pocztą e-mail.

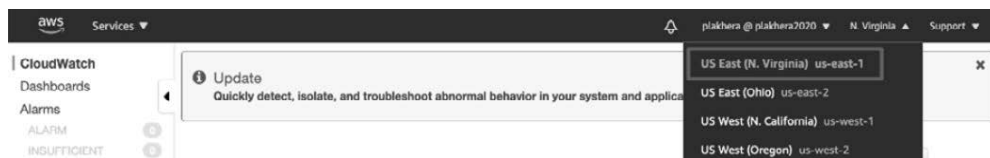
Oto kroki, które trzeba wykonać, by włączyć alarm dotyczący rozliczeń:

1. Po zalogowaniu się w konsoli AWS na stronie <https://aws.amazon.com/console/> na koncie użytkownika głównego (utworzonym podczas pierwszej konfiguracji konta AWS) przejdź do konsoli rozliczeń i zarządzania kosztami pod adres <https://console.aws.amazon.com/billing/> (rysunek 4.13). Kliknij pozycję *Billing preferences* (preferencje dotyczące rozliczeń), a potem zaznacz opcję *Receive Billing Alerts* (otrzymuj alerty dotyczące rozliczeń). Kliknij przycisk *Save preferences* (zapisz preferencje).



Rysunek 4.13. Konsola rozliczeń AWS

2. Przejdź do konsoli CloudWatch pod adres <https://console.aws.amazon.com/cloudwatch/home?region=us-east-1> i upewnij się, że znajdujesz się w regionie *N. Virginia (us-east-1)*, jak to pokazano na rysunku 4.14.



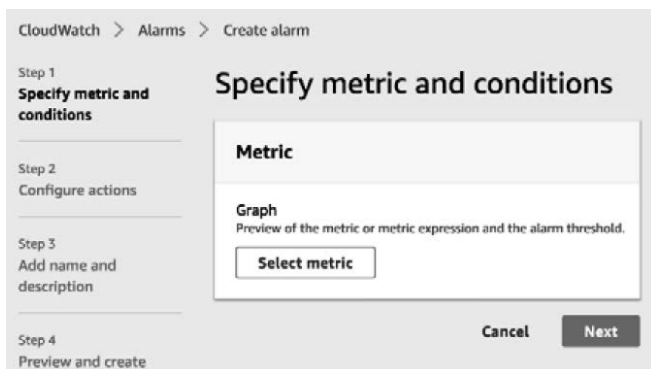
Rysunek 4.14. Wybór regionu N. Virginia do rozliczeń

3. Kliknij pozycję *Alarms*, a potem przycisk *Create alarm* (utwórz alarm; rysunek 4.15).



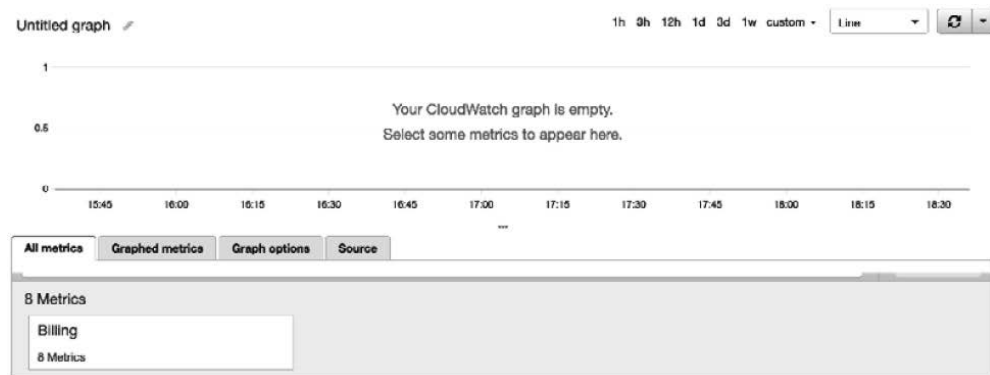
Rysunek 4.15. Tworzenie alarmu CloudWatch

4. Następnie kliknij przycisk *Select metric* (wybierz wskaźnik), pokazany na rysunku 4.16.



Rysunek 4.16. Określanie wskaźnika i warunku

5. Na kolejnym ekranie (rysunek 4.17) wybierz grupę *Billing* (rozliczenia).



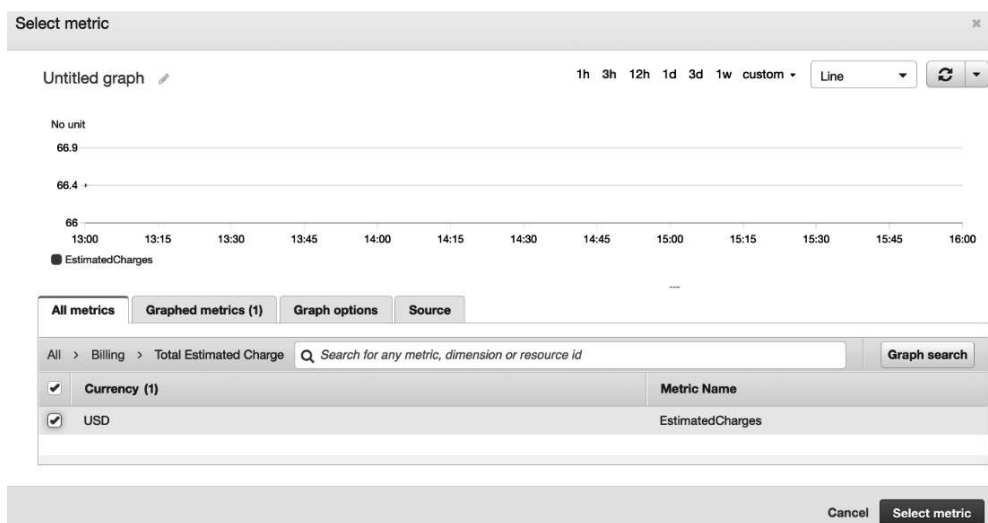
Rysunek 4.17. Wybór wskaźników z grupy Billing

6. Na następnym ekranie (rysunek 4.18) wybierz *Total Estimated Charge* (łącznie przewidywane obciążenie).



Rysunek 4.18. Wybór opcji Total Estimated Charge

7. Zaznacz pole *USD*¹ (dolary amerykańskie), a potem kliknij przycisk *Select metric* (wybierz wskaźnik; rysunek 4.19).



Rysunek 4.19. Wybór dolara jako waluty

Zachowaj wszystkie ustawienia jako domyślne i wybierz dostosowaną do swoich potrzeb wartość progową (w tym przypadku możesz na przykład wybrać 5 USD, co oznacza, że jeśli wartość rachunku z AWS przekroczy pięć dolarów, otrzymasz powiadomienie). Zmień tę wartość zgodnie ze swoimi preferencjami i kliknij przycisk *Next* (dalej; rysunek 4.20).

¹ Zależy to od waluty, w której rozliczane jest konto. Domyślnie są to dolary amerykańskie.— *przyp. tłum.*

CloudWatch > Alarms > Create alarm

Step 1
Specify metric and conditions

Step 2
Configure actions

Step 3
Add name and description

Step 4
Preview and create

Specify metric and conditions

Metric Edit

Graph
This alarm will trigger when the blue line goes above the red line for 1 datapoints within 6 hours.

No unit

Namespace
AWS/Billing

Metric name
EstimatedCharges

Currency
USD

Statistic
Maximum

Period
6 hours

Conditions

Threshold type

☒ Static
Use a value as a threshold

☐ Anomaly detection
Use a band as a threshold

Whenever EstimatedCharges is...

Define the alarm condition.

☒ Greater
> threshold

☐ Greater/Equal
≥ threshold

☐ Lower/Equal
≤ threshold

☐ Lower
< threshold

than...
Define the threshold value.

5 USD

Must be a number

Additional configuration

Cancel Next

Rysunek 4.20. Wybór wartości progowej rozliczeń

- Na następnym ekranie (rysunek 4.21) wybierz opcję *Create new topic* (utwórz nowy temat) albo wybierz któryś z tematów usługi **Simple Notification Service** (SNS) — AWS SNS to całkowicie zarządzana przez dostawcę chmury usługa dostarczania komunikatów, która wyśle pocztą elektroniczną powiadomienie, jeśli przekroczona zostanie pewna wartość progowa. Kliknij przycisk *Create topic* (utwórz temat). Na dole ekranu kliknij przycisk *Next* (dalej).
- Otrzymasz wiadomość e-mail, taką jak na rysunku 4.22. Koniecznie kliknij łącze *Confirm subscription* (potwierdź subskrypcję), bo tylko wtedy będziesz otrzymywać powiadomienia o przekroczeniu wartości progowej.

Step 1
Specify metric and conditions
Step 2
Configure actions
Step 3
Add name and description
Step 4
Preview and create

Configure actions

Notification

Alarm state trigger
Define the alarm state that will trigger this action.

☒ **In alarm**
The metric or expression is outside of the defined threshold.
☐ **OK**
The metric or expression is within the defined threshold.
☐ **Insufficient data**
The alarm has just started or not enough data is available.

Select an SNS topic
Define the SNS (Simple Notification Service) topic that will receive the notification.

☐ Select an existing SNS topic
☒ Create new topic
☐ Use topic ARN

Create a new topic...
The topic name must be unique.

Billing_alarm1

SNS topic names can contain only alphanumeric characters, hyphens (-) and underscores (_).

Email endpoints that will receive the notification...
Add a comma-separated list of email addresses. Each address will be added as a subscription to the topic above.

user1@example.com, user2@example.com

Create topic

Add notification

Auto Scaling action

Add Auto Scaling action

EC2 action

This action is only available for EC2 Per-Instance Metrics.

Add EC2 action

Cancel Previous Next

Rysunek 4.21. Temat usługi SNS

AWS Notifications <no-reply@sns.amazonaws.com>
to me ▾

Thu, Aug 6, 11:47 AM ☆ ↩ ⋮

You have chosen to subscribe to the topic:
arn:aws:sns:us-east-1: Billing_alarm

To confirm this subscription, click or visit the link below (If this was in error no action is necessary):
[Confirm subscription](#)

Please do not reply directly to this email. If you wish to remove yourself from receiving all future SNS subscription confirmation requests please send an email to [sns-opt-out](#)

Rysunek 4.22. Potwierdzenie subskrypcji w usłudze SNS

125

Kup książkę

Poleć książkę

10. Na kolejnym ekranie (rysunek 4.23) uzupełnij pole *Alarm name* (nazwa alarmu), nadając alarmowi sensowną nazwę (na przykład *Billing_Alarm*), i *Alarm description* (opis alarmu), podając zrozumiały opis (na przykład *Billing Alarm when threshold reached 5 dollars* — alarm dotyczący rozliczeń uruchamiany po osiągnięciu wartości progowej 5 dolarów). Gdy już to zrobisz, kliknij przycisk *Next* (dalej).

CloudWatch > Alarms > Create alarm

Step 1
Specify metric and conditions

Step 2
Configure actions

Step 3
Add name and description

Step 4
Preview and create

Add name and description

Name and description

Alarm name
Define a unique name.

Billing_Alarm

Alarm description - optional
Define a description for this alarm.

Billing Alarm when threshold reached 5 dollars

Up to 1024 characters (45/1024)

Cancel Previous Next

Rysunek 4.23. Nazwa i opis alarmu SNS

11. Na etapie podglądu (rysunek 4.24) przejrzyj wszystkie ustawienia i kliknij przycisk *Create alarm* (utwórz alarm).

Proces został już zautomatyzowany: jeśli zostanie przekroczony próg pięciu dolarów, otrzymasz powiadomienie. To doskonały sposób na to, by mieć kontrolę nad kwestiami dotyczącymi bezpieczeństwa i uniknąć przekroczenia budżetu. W następnych podrozdziałach nauczycie się, jak zmniejszyć rachunek za korzystanie z chmury AWS dzięki sprzątaniu nieużywanych zasobów.

CloudWatch > Alarms > Create alarm

Step 1

Specify metric and conditions

Step 2

Configure actions

Step 3

Add name and description

Step 4

Preview and create

Preview and create

Step 1: Specify metric and conditions

Edit

Metric

Graph

This alarm will trigger when the blue line goes above the red line for 1 datapoints within 6 hours.

No unit

EstimatedCharges

Namespace
AWS/Billing

Metric name
EstimatedCharges

Currency
USD

Statistic
Maximum

Period
6 hours

Conditions

Threshold type
Static

Whenever **EstimatedCharges** is
Greater (>)
than...
5

► Additional configuration

Step 2: Configure actions

Edit

Actions

Notification

When in alarm, send a notification to "Billing_alarm"

Step 3: Add name and description

Edit

Name and description

Name
Billing_alarm

Description
Billing Alarm when threshold reached 5 dollar

Cancel

Previous

Create alarm

Rysunek 4.24. Podgląd ustawień alarmu dotyczącego rozliczeń

Praktyczny przykład sprzątania nieużywanych obrazów AMI

Jednym ze sposobów ograniczania kosztów korzystania z chmury AWS jest sprzątanie lub usuwanie starych, nieużywanych obrazów AMI. Proces ten nosi nazwę wyrejestrowywania obrazu AMI. Nie ma wpływu na już działające instancje, ale z wyrejestrowanego obrazu nie można uruchomić nowych.

Istnieje kilka sposobów osiągnięcia tego celu (sprzątnięcia obrazów AMI). My zrobimy to, łącząc funkcję Lambda i reguły CloudWatch. Oto kroki, które trzeba wykonać:

1. Pierwszym z nich jest utworzenie funkcji Lambda. Przejdź do konsoli tej usługi na stronę <https://us-west-2.console.aws.amazon.com/lambda/home> (rysunek 4.25) i kliknij przycisk *Create function* (utwórz funkcję).



Rysunek 4.25. Konsola AWS Lambda

2. Wprowadź dane, tak jak na rysunku 4.26:
 - *Function name* (nazwa funkcji) — nadaj funkcji Lambda nazwę, na przykład `cleanupunusedami`,
 - *Runtime* (środowisko uruchomieniowe) — *Python 3.7*,
 - *Execution role* (rola wykonawcza) — wybierz opcję *Create a new role with basic Lambda permissions* (utwórz nową rolę z podstawowymi uprawnieniami usługi Lambda).
 Kliknij przycisk *Create function* (utwórz funkcję).
3. Na następnym ekranie (rysunek 4.27) kliknij kartę *Configuration* (konfiguracja).
4. Na kolejnym ekranie (rysunek 4.28) w bocznym menu kliknij pozycję *Permissions* (uprawnienia), a potem przycisk *Edit* (edytuj).
5. Kliknij nazwę roli (na przykład `cleanupunusedami-role-8mkhxx5f` — u Ciebie będzie ona inna; rysunek 4.29).

Lambda > Functions > Create function

Create function Info

Choose one of the following options to create your function.

Author from scratch ☒

Start with a simple Hello World example.

Use a blueprint ☐

Build a Lambda application from sample code and configuration presets for common use cases.

Container image ☐

Select a container image to deploy for your function.

Browse serverless app repository ☐

Deploy a sample Lambda application from the AWS Serverless Application Repository.

Basic information

Function name
Enter a name that describes the purpose of your function.
cleanupunusedami
Use only letters, numbers, hyphens, or underscores with no spaces.

Runtime Info
Choose the language to use to write your function. Note that the console code editor supports only Node.js, Python, and Ruby.
Python 3.7

Architecture Info
Choose the instruction set architecture you want for your function code.
☒ x86_64
☐ arm64

Permissions Info
By default, Lambda will create an execution role with permissions to upload logs to Amazon CloudWatch Logs. You can customize this default role later when adding triggers.

▼ Change default execution role

Execution role
Choose a role that defines the permissions of your function. To create a custom role, go to the IAM console.

☒ Create a new role with basic Lambda permissions
☐ Use an existing role
☐ Create a new role from AWS policy templates

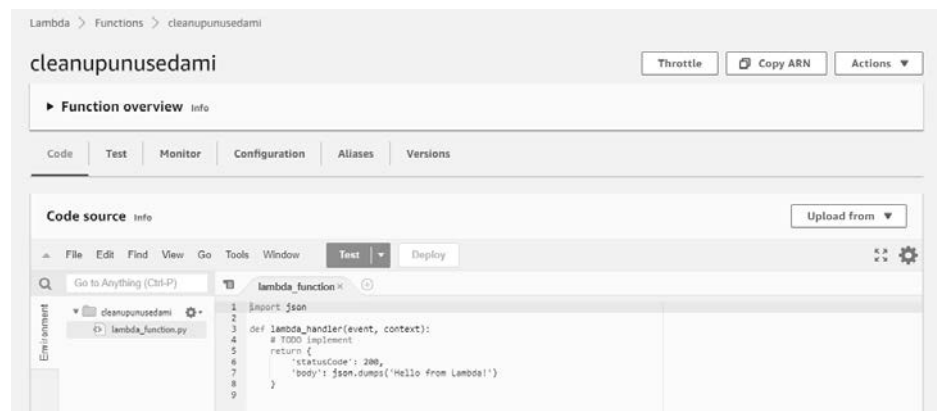
① Role creation might take a few minutes. Please do not delete the role or edit the trust or permissions policies in this role.

Lambda will create an execution role named cleanupunusedami-role-qxk61f6z, with permission to upload logs to Amazon CloudWatch Logs.

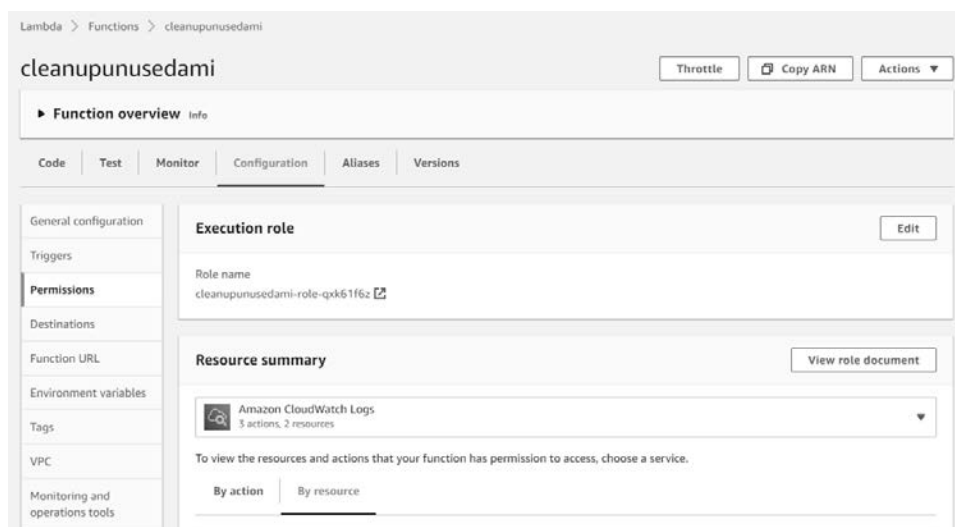
► Advanced settings

Cancel Create function

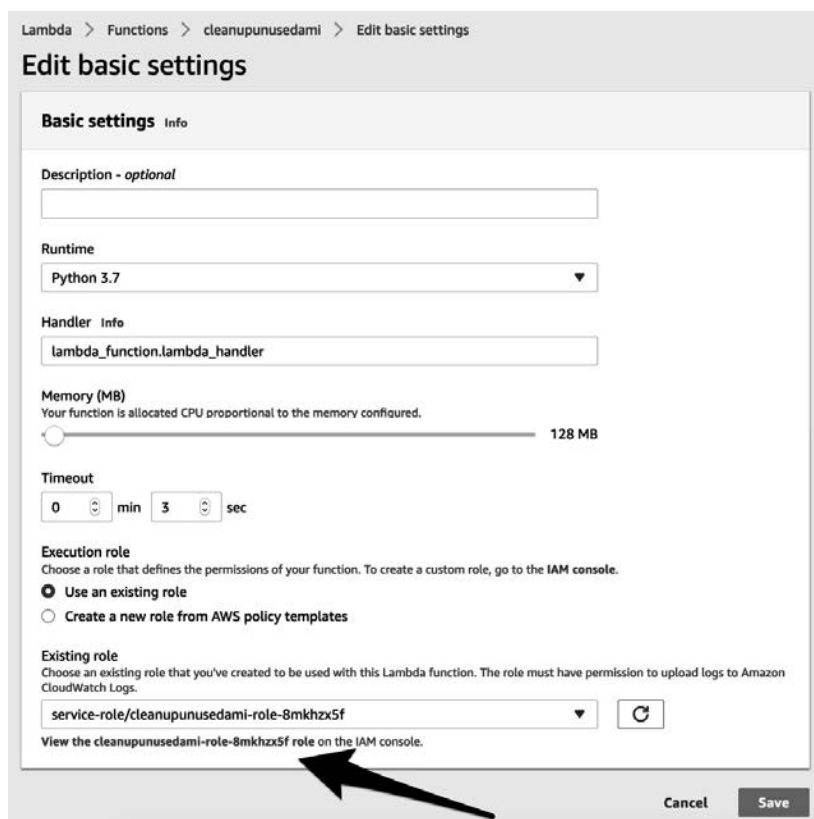
Rysunek 4.26. Tworzenie funkcji Lambda



Rysunek 4.27. Przegląd funkcji Lambda

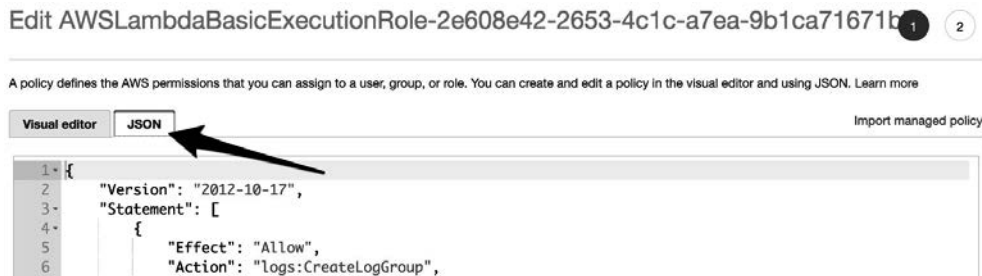


Rysunek 4.28. Edytowanie uprawnień IAM funkcji Lambda



Rysunek 4.29. Rola IAM funkcji Lambda

6. Kliknij nazwę zasady (w tym przypadku na przykład `AWSLambdaBasicExecutionRole-2e608e42-2653-4c1c-a7ea-9b1ca71671b`, ale u Ciebie będzie inna), a potem przycisk *Edit policy* (edytuj zasady). Po przejściu do edycji zasad kliknij kartę *JSON* (rysunek 4.30).

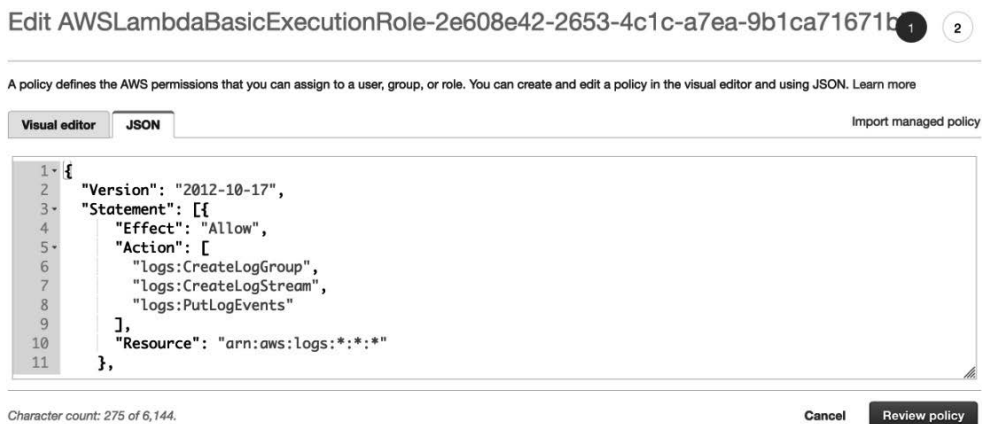


Rysunek 4.30. Zasada dla funkcji Lambda w formacie JSON

7. Zastąp kod na karcie *JSON* tym przedstawionym poniżej. Zmieniamy domyślne zasady IAM, ponieważ skrypt języka Python wymaga pewnych dodatkowych uprawnień, między innymi: `DescribeImages`, które pozwala utworzyć listę wszystkich obrazów, `ec2:DescribeRegions`, służącego do pobrania listy wszystkich regionów AWS, i `ec2:DeregisterImage` — faktycznego uprawnienia do wyrejestrowywania i czyszczenia obrazów AMI. Wymaganymi uprawnieniami IAM, oprócz domyślnych, są: uprawnienie do tworzenia grupy dzienników CloudWatch (`logs:CreateLogGroup`), uprawnienie do tworzenia strumienia dziennika CloudWatch przeznaczonego do zapisywania dzienników funkcji Lambda (`logs:CreateLogStream`) i uprawnienie do umieszczania wpisów w dziennikach CloudWatch (`logs:PutLogEvents`):

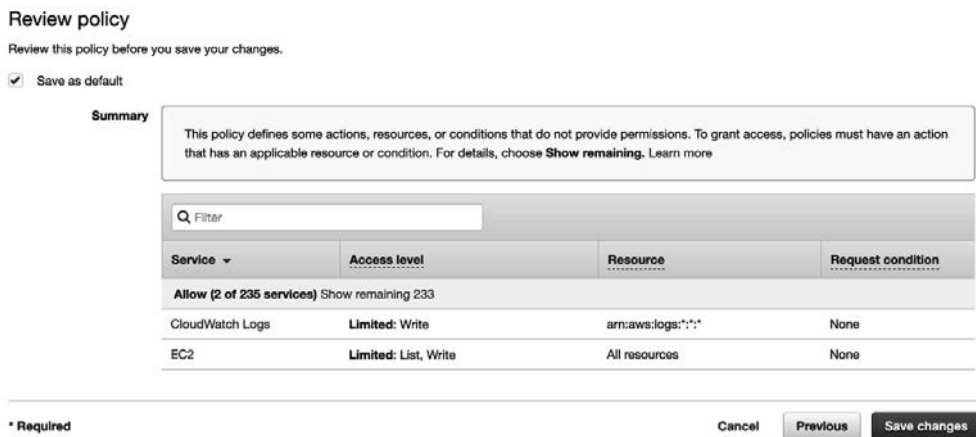
```
{
  "Version": "2012-10-17",
  "Statement": [{
    "Effect": "Allow",
    "Action": [
      "logs:CreateLogGroup",
      "logs:CreateLogStream",
      "logs:PutLogEvents"
    ],
    "Resource": "arn:aws:logs:*:*:*"
  }],
  {
    "Effect": "Allow",
    "Action": [
      "ec2:DescribeImages",
      "ec2:DescribeRegions",
      "ec2:DeregisterImage"
    ],
    "Resource": "*"
  }
]
```

8. Kliknij przycisk *Review policy* (przejrzyj zasady; rysunek 4.31).



Rysunek 4.31. Przegląd zasad IAM funkcji Lambda

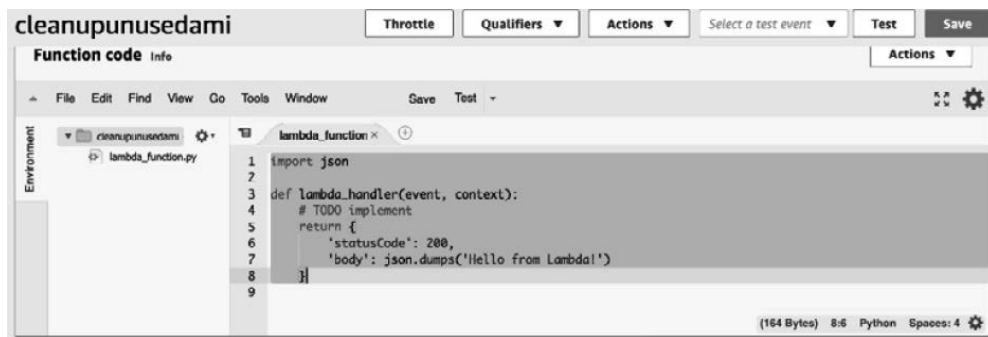
9. Kliknij przycisk *Save changes* (zapisz zmiany; rysunek 4.32).



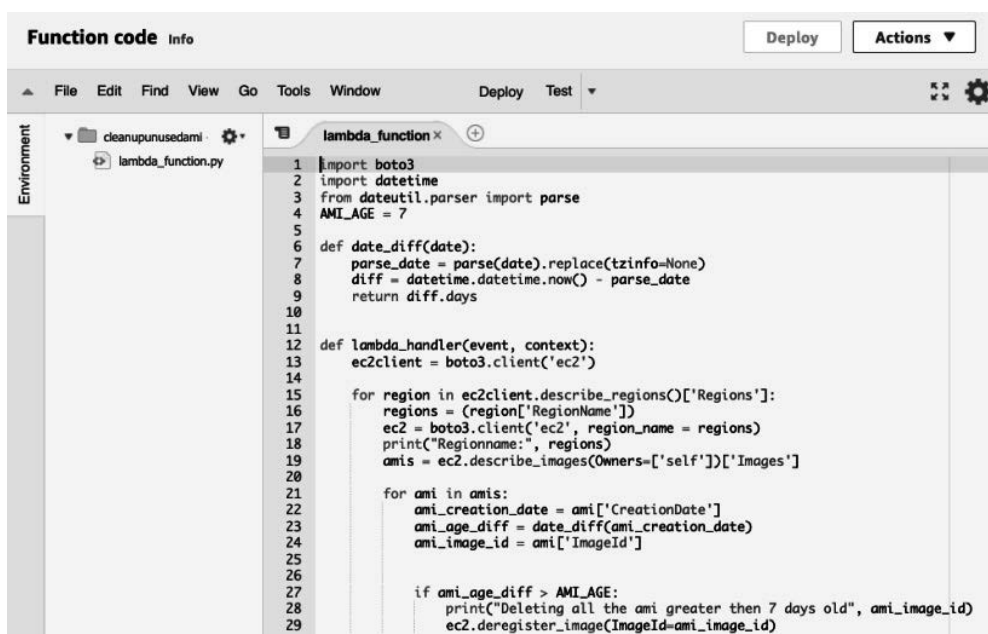
Rysunek 4.32. Zapisywanie zasad IAM funkcji Lambda

10. Wróć do konsoli Lambda, wyczyść cały domyślny kod (rysunek 4.33) i skopiuj ten z archiwum dostępnego pod adresem <https://ftp.helion.pl/przyklady/awsadm.zip>, z pliku `r04/python/cleanup_unused_ami`. W kilku następnych akapitach prześledzimy ten kod krok po kroku.

11. Zastąp kod domyślny tym z rysunku 4.34.



Rysunek 4.33. Edytor kodu w konsoli Lambda



Rysunek 4.34. Kod w języku Python do czyszczenia obrazów AMI

12. Spróbujmy teraz przeanalizować, jak działa ten kod. Najpierw importowane są wszystkie standardowe moduły języka Python (zestaw SDK boto3 oraz moduły datetime i dateutil, służące do pobierania bieżącej daty i liczby dni, które upłynęły od utworzenia obrazu AMI):

```

import boto3
import datetime
from dateutil.parser import parse

```

13. W następnym kroku trzeba utworzyć funkcję zwracającą różnicę (w dniach) pomiędzy datą utworzenia obrazu AMI a bieżącą:

```
def date_diff(date):
    parse_date = parse(date).replace(tzinfo=None)
    diff = datetime.datetime.now() - parse_date
    return diff.days
```

14. Teraz pobierana jest lista wszystkich regionów (ponieważ obraz AMI jest w tej usłudze jednostką regionalną, wartość jego identyfikatora — AMI ID — jest inna w różnych regionach):

```
for region in ec2client.describe_regions()['Regions']:
    regions = (region['RegionName'])
    ec2 = boto3.client('ec2', region_name = regions)
    print("Regionname:", regions)
```

15. Następnie pobierane są posiadane (utworzone) przez Ciebie obrazy AMI oraz daty ich powstania i identyfikatory. Za pomocą funkcji `date_diff` określana jest liczba dni, które upłynęły od momentu utworzenia obrazu AMI do chwili obecnej:

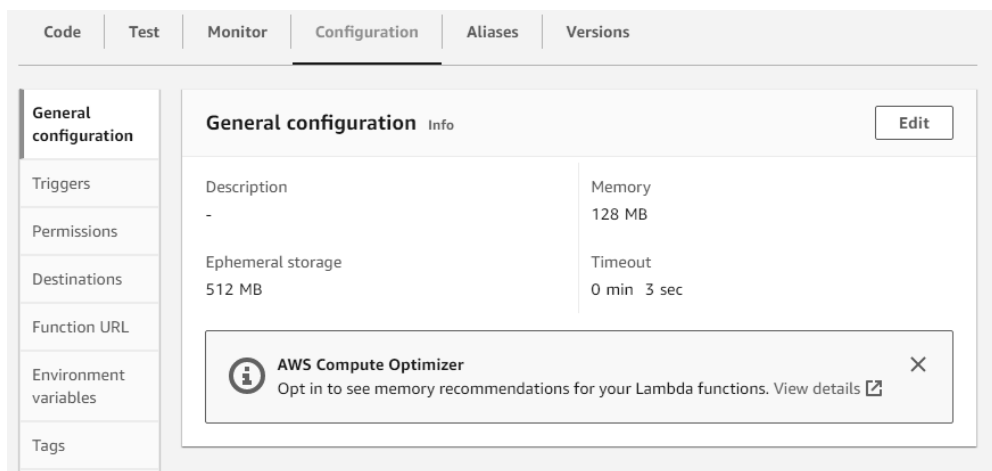
```
amis = ec2.describe_images(Owners=['self'])['Images']

for ami in amis:
    ami_creation_date = ami['CreationDate']
    ami_age_diff = date_diff(ami_creation_date)
    ami_image_id = ami['ImageId']
```

16. W kolejnym kroku obraz AMI zostaje wyrejestrowany, jeśli jest starszy niż 7 dni. Czas ten zależy od wymagań i na początek idealną wartością będzie 60 albo 90 dni, ale w tym przykładzie przyjmijmy 7:

```
if ami_age_diff > AMI_AGE:
    print("Deleting all the ami greater than 7 days old", ami_image_id)
    ec2.deregister_image(ImageId=ami_image_id)
```

17. W sekcji *General configuration* (ogólna konfiguracja), pokazanej na rysunku 4.35, kliknij przycisk *Edit* (edytuj).



Rysunek 4.35. Podstawowe ustawienia funkcji Lambda

18. Zwiększ wartość *Timeout* (limit czasu) z domyślnych 3 *sec* do ponad 2 *min*. Zmieniamy to ustawienie, ponieważ na koncie może być wiele obrazów AMI, a trzy sekundy na wykonanie funkcji Lambda to bardzo mało czasu. Kliknij przycisk *Save* (zapisz; rysunek 4.36).

Lambda > Functions > cleanupunusedami > Edit basic settings

Edit basic settings

Basic settings Info

Description - optional

Runtime

Python 3.7

Handler Info

lambda_function.lambda_handler

Memory (MB)

Your function is allocated CPU proportional to the memory configured.

128 MB

Timeout

2 min 3 sec

Execution role

Choose a role that defines the permissions of your function. To create a custom role, go to the [IAM console](#).

☒ Use an existing role

☐ Create a new role from AWS policy templates

Existing role

Choose an existing role that you've created to be used with this Lambda function. The role must have permission to upload logs to Amazon CloudWatch Logs.

service-role/cleanupunusedami-role-8mkhxx5f

View the cleanupunusedami-role-8mkhxx5f role on the IAM console.

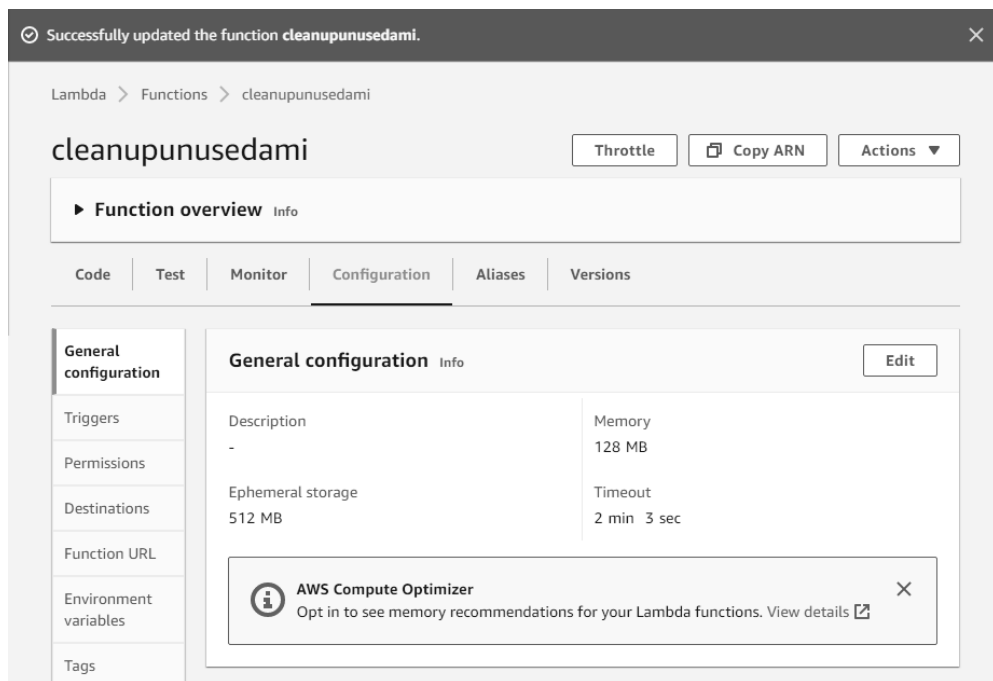
Cancel Save

Rysunek 4.36. Ustawienia limitu czasu działania funkcji Lambda

19. Jak widać na rysunku 4.37, funkcja Lambda została zaktualizowana.

Do wyzwolenia funkcji Lambda będzie potrzebne zdarzenie CloudWatch². W zdarzeniach tej usługi, używając wyrażeń cron lub rate, można utworzyć samowyzwalającą się regułę opartą na harmonogramie. W tym przypadku definiowana jest stała wartość (ang. *fixed rate*) jednego dnia, dzięki czemu funkcja Lambda będzie wyzwalana codziennie.

² Usługę CloudWatch Events zastępuje usługa EventBridge i jest rekomendowanym sposobem obsługi zdarzeń.— *przyp. tłum.*



Rysunek 4.37. Aktualizacja funkcji Lambda

Aby to skonfigurować, wykonaj następującą serię kroków:

1. Przejdź do konsoli CloudWatch na stronę <https://us-west-2.console.aws.amazon.com/cloudwatch/home>, kliknij pozycję **Rules** (reguły), znajdujący się u góry ekranu przycisk **Back to CloudWatch Events** (wróć do usługi CloudWatch Events), a potem przycisk **Create rule** (utwórz regułę; rysunek 4.38).

Step 1: Create rule

Create rules to invoke Targets based on Events happening in your AWS environment.

Event Source

Build or customize an Event Pattern or set a Schedule to invoke Targets.

☐ Event Pattern ⓘ ☒ **Schedule** ⓘ

☒ **Fixed rate of** **Days** ▼

☐ **Cron expression**

Learn more about CloudWatch Events schedules.

► Show sample event(s)

Targets

Select Target to invoke when an event matches your Event Pattern or when schedule is triggered.

Lambda function ▼ ⓘ

Function*

► Configure version/alias

► Configure input

[Add target*]

* Required

[Cancel] [Configure details]

Rysunek 4.38. Reguła zdarzenia CloudWatch

W sekcji *Event Source* (źródło zdarzeń) wybierz opcję *Schedule* (harmonogram), a następnie opcję *Fixed rate of* (stała wartość) i wprowadź liczbę 1 oraz jednostkę *Days* (dni). Pod nagłówkiem *Targets* (elementy docelowe) wybierz z listy rozwijanej utworzoną w poprzednim kroku funkcję Lambda (na przykład *cleanupunusedami*). Kliknij przycisk *Configure details* (konfiguruj szczegóły).

2. W sekcji *Configure rule details* (konfigurowanie szczegółów roli), pokazanej na rysunku 4.39, nadaj regule sensowną nazwę (na przykład *amicleanup*) i dodaj opis (na przykład *Lambda Function to cleanup unused ami on daily basis* — funkcja Lambda do codziennego sprzątania nieużywanych obrazów AMI). Kliknij przycisk *Create rule* (utwórz regułę).

Step 2: Configure rule details

Rule definition

Name*

Description

State ☒ Enabled

CloudWatch Events will add necessary permissions for target(s) so they can be invoked when this rule is triggered.

* Required Cancel Back Create rule

Rysunek 4.39. Tworzenie reguły CloudWatch

Mamy już skonfigurowaną regułę CloudWatch i funkcję Lambda, która, uruchamiana codziennie, będzie sprzątać obrazy AMI starsze niż siedem dni. Pozwoli to zmniejszyć koszty korzystania z chmury AWS.

Praktyczny przykład usuwania nieużywanych woluminów EBS

W tym podrozdziale dowiecie się, jak ograniczyć wydatki dzięki sprzątaniu nieużywanych woluminów usługi **Elastic Block Storage (EBS)**. O kosztach utrzymywania woluminów EBS stale się zapomina, ponieważ cykl użytkowy takiego woluminu jest niezależny od cyklu działania instancji. Nawet jeśli zostanie ona usunięta, jej wolumin EBS wciąż istnieje i generuje koszty.

Cel ten można osiągnąć na wiele sposobów, ale my, tak jak w przykładzie opisanym w poprzednim rozdziale, połączymy funkcję Lambda z regułami CloudWatch. Zastosujemy takie rozwiązanie, ponieważ zapewnia dużą elastyczność: można na przykład filtrować woluminy na podstawie tego, czy są używane, czy też nie.

Aby utworzyć funkcję Lambda, wykonaj te same kroki (1. – 5.), które zostały opisane w podrozdziale „Praktyczny przykład sprzątania nieużywanych obrazów AMI”. Trzeba tylko zmienić kilka parametrów:

1. Krok ten będzie podobny do tych z wcześniejszego przykładu. Wystarczy, że zmienisz nazwę funkcji, na przykład na `cleanupunattachedebsvol`. Reszta kroków, od 2. do 6., będzie taka sama, ale w kroku 7. zmień zasady IAM na te pokazane poniżej. Zastępujemy domyślne zasady IAM, ponieważ skrypt języka Python wymaga pewnych dodatkowych uprawnień, między innymi: `DeleteVolume` do usuwania nieużywanych woluminów, `ec2:DescribeRegions` do pobierania listy wszystkich regionów AWS i `ec2:DescribeVolumes` do dodawania do woluminu opisu.

Do wymaganych uprawnień IAM, poza domyślnym, należy także uprawnienie do tworzenia grupy dzienników CloudWatch (`logs:CreateLogGroup`), uprawnienie do tworzenia strumienia dziennika CloudWatch przeznaczonego do zapisywania dzienników funkcji Lambda (`logs:CreateLogStream`) i uprawnienie do umieszczania wpisów w dziennikach CloudWatch (`logs:PutLogEvents`):

```
{
  "Version": "2012-10-17",
  "Statement": [{
    "Effect": "Allow",
    "Action": [
      "logs:CreateLogGroup",
      "logs:CreateLogStream",
      "logs:PutLogEvents"
    ],
    "Resource": "arn:aws:logs:*:*:*"
  },
  {
    "Effect": "Allow",
    "Action": [
      "ec2:DeleteVolume",
      "ec2:DescribeRegions",
      "ec2:DescribeVolumes"
    ],
    "Resource": "*"
  }
]
```

2. Kliknij przycisk *Review policy* (przejrzyj zasady; rysunek 4.40).
3. Kliknij przycisk *Save changes* (zapisz zmiany; rysunek 4.41).
4. Wróć do konsoli Lambda, wyczyść cały domyślny kod i wklej ten z archiwum dostępnego pod adresem <https://ftp.helion.pl/przyklady/awssadm.zip>, z pliku `r04/python/cleanup_unattached_ebs_vol`. W kilku następnych akapitach prześledzimy go krok po kroku. Cały kod będzie wyglądać tak jak na rysunku 4.42.

Edit AWSLambdaBasicExecutionRole-c5126003-b20d-4ecd-a1fd-78f62 1 2

A policy defines the AWS permissions that you can assign to a user, group, or role. You can create and edit a policy in the visual editor and using JSON. [Learn more](#)

Visual editor

JSON

[Import managed policy](#)

```

1 {
2   "Version": "2012-10-17",
3   "Statement": [{
4     "Effect": "Allow",
5     "Action": [
6       "logs:CreateLogGroup",
7       "logs:CreateLogStream",
8       "logs:PutLogEvents"
9     ],
10  }],

```

Character count: 273 of 6,144.

Cancel

Review policy

Rysunek 4.40. Przegląd zasad IAM funkcji Lambda

Review this policy before you save your changes.

☒ Save as default

Summary

This policy defines some actions, resources, or conditions that do not provide permissions. To grant access, policies must have an action that has an applicable resource or condition. For details, choose **Show remaining**. [Learn more](#)

Q Filter

Service ▾

Access level

Resource

Request con

Allow (2 of 235 services) Show remaining 233

CloudWatch Logs

Limited: Write

arn:aws:logs:*:*

None

EC2

Limited: List, Write

All resources

None

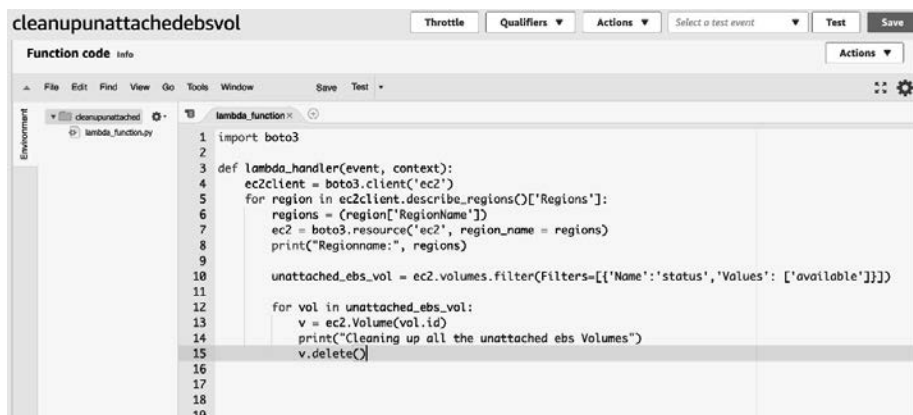
* Required

Cancel

Previous

Save changes

Rysunek 4.41. Zapis zasad IAM funkcji Lambda



Rysunek 4.42. Kod w języku Python do sprzątania woluminów EBS

5. Spróbujcie zrozumieć, jak działa ten kod. Najpierw importowane są wszystkie moduły standardowe języka Python (w tym przypadku zestaw SDK boto3):

```
import boto3
```

Następnie pobierana jest lista wszystkich regionów (trzeba w nich znaleźć wszystkie nieużywane woluminy EBS):

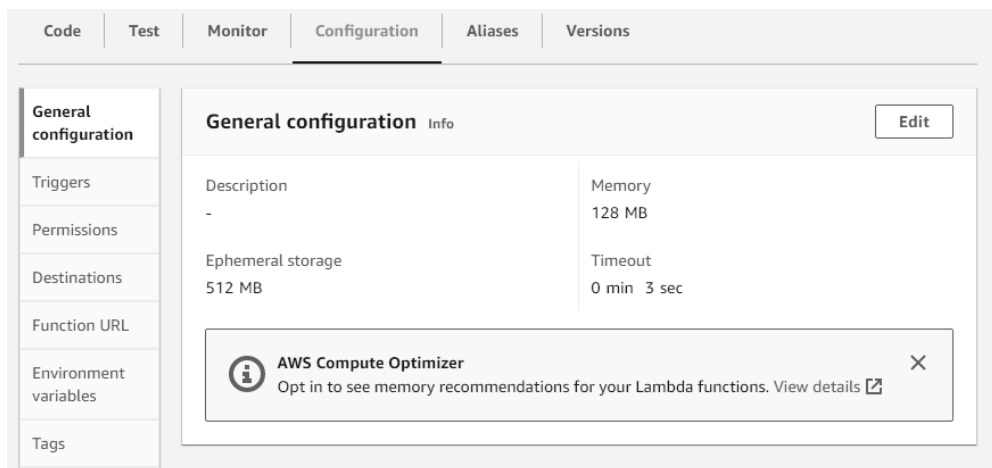
```
ec2client = boto3.client('ec2')
```

```
for region in ec2client.describe_regions()['Regions']:
    regions = (region['RegionName'])
    ec2 = boto3.resource('ec2', region_name = regions)
    print("Regionname:", regions)
```

Potrzebny jest teraz filtr do znajdowania wszystkich woluminów, które mają stan `available` (dostępny). W tym celu tworzony jest filtr z kryteriami: `Name` (nazwa) jako `status` i `Values` (wartości) jako `available`, który odfiltruje tylko dostępne woluminy:

```
unattached_ebs_vol =
ec2.volumes.filter(Filters=[{'Name':'status','Values':['available']}])
for vol in unattached_ebs_vol:
    v = ec2.Volume(vol.id)
    print("Cleaning up all the unattached ebs Volumes")
    v.delete()
```

6. Po przejściu na kartę *Configuration* (konfiguracja) znajdziesz nagłówek *General configuration* (ogólna konfiguracja), pokazany na rysunku 4.43. Kliknij tam przycisk *Edit* (edytuj).



Rysunek 4.43. Podstawowe ustawienia funkcji Lambda

7. Zwiększ wartość *Timeout* (limit czasu) z domyślnych 3 sec do ponad 1 min, ponieważ na koncie może być wiele woluminów EBS, a trzy sekundy na wykonanie funkcji Lambda to bardzo mało. Kliknij przycisk *Save* (zapisz; rysunek 4.44).

Basic settings Info

Description - optional

Runtime

Python 3.7

Handler Info

lambda_function.lambda_handler

Memory (MB)

Your function is allocated CPU proportional to the memory configured.

128 MB

Timeout

1 min 3 sec

Execution role

Choose a role that defines the permissions of your function. To create a custom role, go to the [IAM console](#).

☒ Use an existing role

☐ Create a new role from AWS policy templates

Existing role

Choose an existing role that you've created to be used with this Lambda function. The role must have permission to upload logs to Amazon CloudWatch Logs.

service-role/cleanupunattachedebsvol-role-bqcof7y8

View the [cleanupunattachedebsvol-role-bqcof7y8](#) role on the IAM console.

Cancel Save

Rysunek 4.44. Ustawienia limitu czasu działania funkcji Lambda

Funkcja Lambda jest już gotowa. Aby utworzyć regułę zdarzenia CloudWatch do wyzwalania tej funkcji, zastosuj kroki 1. – 3. z podrozdziału „Praktyczny przykład sprzątania nieużywanych obrazów AMI”. W kroku 2. musisz zmienić kilka parametrów. W tym przykładzie wybierz dla elementów docelowych inną funkcję Lambda (na przykład `cleanupunattachedebsvol`). Tak jak to zdefiniowano wcześniej w kroku 3., skonfiguruj regułę, ale tym razem nadaj jej inną nazwę, na przykład `cleanupunattachedebsvol`.

Reguła CloudWatch i funkcja Lambda, która będzie uruchamiana codziennie, żeby sprzątać nieużywane woluminy EBS, są już skonfigurowane. Pozwoli to zmniejszyć wydatki na chmurę AWS.

Praktyczny przykład codziennego wyłączania instancji

Na kolejnym przykładzie nauczycie się, jak codziennie wyłączać instancje, na przykład o 21:00, w celu ograniczania kosztów. Takie rozwiązanie najbardziej przydaje się w środowisku nieprodukcyjnym albo rozwojowym.

Z dotychczasowych przykładów dowiedzieliście się, jak korzystać z konsoli AWS. Tym razem zautomatyzujemy proces w narzędziu Terraform.

Musisz wykonać następujące kroki:

1. Pierwszym z nich jest tworzenie roli IAM. Używa się w tym celu zasobu Terraform `aws_iam_role`. Zasady te nadadzą funkcji Lambda uprawnienia do przyjmowania roli:

```
"aws_iam_role" "iam_for_lambda" {
  name = "iam_for_lambda"

  assume_role_policy = <<EOF
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": "sts:AssumeRole",
      "Principal": {
        "Service": "lambda.amazonaws.com"
      },
      "Effect": "Allow",
      "Sid": ""
    }
  ]
}
EOF
}
```

2. Następnie przy użyciu zasobu `aws_iam_policy` musisz utworzyć zasadę IAM, która nada funkcji uprawnienia niezbędne do uruchamiania i zatrzymywania instancji (`ec2:Start/ec2:Stop`), tworzenia grupy dzienników CloudWatch (`logs:CreateLogGroup` i `logs:CreateLogStream`) oraz umieszczania tam zdarzeń (`logs:PutLogEvents`):

```
resource "aws_iam_policy" "lambda_logging" {
  name       = "lambda_logging"
  path       = "/"
  description = "Zasada IAM do rejestrowania zdarzeń z funkcji Lambda"

  policy = <<EOF
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "logs:CreateLogGroup",
        "logs:CreateLogStream",
        "logs:PutLogEvents"
      ],
      "Resource": "arn:aws:logs:*:*:*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "ec2:Start*",
        "ec2:Stop*"
      ],

```

```

    ],
    "Resource": "*"
  }
]
}
EOF
}

```

3. Dołącz zasadę IAM do roli IAM za pomocą zasobu `aws_iam_role_policy_attachment`:

```

resource "aws_iam_role_policy_attachment" "lambda_logs" {
  role      = aws_iam_role.iam_for_lambda.name
  policy_arn = aws_iam_policy.lambda_logging.arn
}

```

4. W następnym kroku, zanim przekażesz kod do usługi Lambda, musisz spakować go do archiwum. Sprawdźmy jednak wcześniej, co on robi.

5. Najpierw importujemy wszystkie moduły standardowe języka Python (w tym przypadku zestaw SDK boto3):

```
import boto3
```

6. Następnie pozyskujemy listę wszystkich regionów:

```

for region in ec2client.describe_regions()['Regions']:
    regions = (region['RegionName'])
    ec2 = boto3.resource('ec2', region_name = regions)
    print("Nazwy regionów:", regions)

```

7. Ponieważ trzeba zatrzymać tylko te instancje, które są uruchomione, należy je odfiltrować:

```

running_instances = ec2.instances.filter(Filters=[{'Name':
↳ 'instance-state-name', 'Values': ['running']}])

```

8. Wreszcie na podstawie filtra dokonuje się iteracji po instancjach i są one zatrzymywane:

```

for instance in running_instances:
    print("Zatrzymywanie instancji: ", instance.id)
    instance.stop()

```

9. Zapisz ten skrypt języka Python w pliku `ec2_stop.py` i spakuj go do archiwum `lambda.zip`:

```

zip lambda.zip ec2_stop.py
adding: ec2_stop.py (deflated 45%)

```

10. W następnym kroku musimy utworzyć funkcję Lambda za pomocą narzędzia Terraform. W tym celu użyjemy zasobu `aws_lambda_function`, który ma następujące parametry:

- `filename` (nazwa pliku) — spakuj utworzony wcześniej plik z kodem (`lambda.zip`).
- `function_name` (nazwa funkcji) — nadaj funkcji sensowną nazwę (`stop_ec2_nightly`).
- `role` (rola) — jest to rola IAM utworzona w pierwszym kroku (`aws_iam_role.iam_for_lambda.arn`).

- handler (procedura obsługi) — gdy stworzysz funkcję Lambda, musisz określić procedurę obsługi, czyli zawartą w kodzie funkcję, którą usługa będzie wywoływać, gdy przejdzie do wykonania kodu.
- source_code_hash (wartość skrótu kodu źródłowego) — używa się jej do wyzwalania aktualizacji. W argumencie tym trzeba ustawić zakodowany algorytmem Base64 skrót SHA256 zawartości pakietu. Podaje się klucz obiektu s3_key albo nazwę pliku.
- runtime (środowisko uruchomieniowe) — w tym przykładzie używany jest język Python, ale Lambda obsługuje też inne środowiska, takie jak Node.js, Java, .NET Core, Go, Ruby, a także niestandardowe:

```
resource "aws_lambda_function" "test_lambda" {
  filename      = "lambda.zip"
  function_name = "stop_ec2_nightly"
  role          = aws_iam_role.iam_for_lambda.arn
  handler       = "lambda.lambda_handler"
  source_code_hash = base64sha256("lambda.zip")
  runtime       = "python3.7"
}
```

Do wyzwalania funkcji Lambda potrzebne jest jeszcze zdarzenie CloudWatch. Aby je utworzyć, wykorzystamy zasób Terraform `aws_cloudwatch_event_rule`, który będzie wyzwaliał regułę o 21:00 czasu UTC (można to zmienić albo ustawić inny czas, stosownie do wymagań). Potem przy użyciu `aws_cloudwatch_event_target` zostanie skonfigurowany element docelowy, czyli utworzona ostatnio funkcja Lambda:

```
resource "aws_cloudwatch_event_rule" "cron_expr" {
  name           = "cron-expression"
  description    = "Uruchamia się codziennie o 21:00 czasu UTC."
  schedule_expression = "cron(0 21 * * ? *)"
}

resource "aws_cloudwatch_event_target" "cron_expr_target" {
  {
    rule      = aws_cloudwatch_event_rule.cron_expr.name
    target_id = "lambda"
    arn       = aws_lambda_function.test_lambda.arn
  }
}

resource "aws_lambda_permission" "allow_cloudwatch_to_call_lambda" {
  statement_id = "AllowExecutionFromCloudWatch"
  action       = "lambda:InvokeFunction"
  function_name = aws_lambda_function.test_lambda.function_name
  principal    = "events.amazonaws.com"
  source_arn   = aws_cloudwatch_event_rule.cron_expr.arn
}
```

Tym sposobem dzięki codziennemu zatrzymywaniu instancji możesz zmniejszyć koszty, przynajmniej w środowisku nieprodukcyjnym. Jeśli firma wymaga, by uruchamiać ponownie te instancje o 9:00, zmodyfikuj kod funkcji Lambda i zamień w instrukcji `instance.start()` metodę `stop` na `start`.

Podsumowanie

W rozdziale tym zostało przedstawione jedno z najważniejszych rozwiązań AWS, czyli usługa EC2. Od tego większość użytkowników zaczyna pracę na tej platformie, wdrażając własną aplikację lub witrynę internetową.

Macie już wiedzę na temat tego rozwiązania obliczeniowego chmury AWS i umiecie tworzyć instancje EC2 w konsoli AWS oraz w usłudze CloudFormation. Przekonaliście się też, o ile prostsze jest zarządzanie wydajnością, gdy nie trzeba zamawiać z wyprzedzeniem żadnego sprzętu komputerowego. Nauczyliście się również na przykładach, jak realnie zmniejszać koszty użytkowania chmury AWS dzięki wyłączaniu lub sprzątaniu nieużywanych zasobów AWS.

W następnym rozdziale skupimy się na usłudze Elastic Load Balancing i różnych rozwiązaniach równoważenia obciążenia oferowanych przez AWS. Do tej pory tworzyliśmy tylko kilka serwerów, ale wraz ze wzrostem wykorzystania aplikacji będziemy musieli dodawać ich coraz więcej i zapewnić równomierne rozłożenie obciążenia między nimi, a równocześnie na wypadek awarii któregoś węzła — zablokować kierowanie do niego ruchu. W takich właśnie sytuacjach przydaje się moduł równoważenia obciążenia. Następny rozdział rozpoczniemy od skonfigurowania funkcji Application Load Balancer, najpierw w konsoli AWS, a potem za pomocą narzędzia Terraform.

PROGRAM PARTNERSKI

— GRUPY HELION —



1. ZAREJESTRUJ SIĘ
2. PREZENTUJ KSIĄŻKI
3. ZBIERAJ PROWIZJĘ

Zmień swoją stronę WWW w działający bankomat!

Dowiedz się więcej i dołącz już dzisiaj!

<http://program-partnerski.helion.pl>

GRUPA
Helion 

AWS, czyli dostępność, odporność i niezawodność aplikacji!

Amazon Web Services (**AWS**) zdobywa coraz większe uznanie. Platforma **AWS** udostępnia znakomite rozwiązania, w tym usługi obliczeniowe, magazyn danych, obsługę sieci i usług zarządzanych. Aplikacje korporacyjne wdrożone w chmurze **AWS** mogą być wyjątkowo odporne, skalowalne i niezawodne. Aby takie były, administrator systemu musi jednak zrozumieć koncepcje zaawansowanego zarządzania chmurą i nauczyć się stosować je w praktyce.

W tej książce omówiono techniki wdrażania systemów na platformie **AWS** i zasady zarządzania nimi. Zaprezentowano podstawy korzystania z usługi Identity and Access Management, narzędzia sieciowe i monitorujące chmury **AWS**. Poruszono tematy Virtual Private Cloud, Elastic Compute Cloud, równoważenia obciążenia, automatycznego skalowania i baz danych usługi Relational Database Service. Dokładnie przedstawiono zasady wdrażania aplikacji i zarządzania danymi. Pokazano też, jak zainicjować automatyczne tworzenie kopii zapasowych, a także jak śledzić i przechowywać pliki dzienników. Znalazły się tu również informacje na temat interfejsów API platformy **AWS** i sposobu ich użycia, automatyzacji infrastruktury z wykorzystaniem usługi CloudFormation, narzędzia Terraform i skryptów w języku Python z biblioteką Boto3.

W książce między innymi:

- zasady bezpieczeństwa w systemach chmurowych
- tworzenie usług Amazon Elastic Compute Cloud (EC2)
- konfiguracja centrum danych w chmurze **AWS** za pomocą sieci VPC
- automatyczne skalowanie aplikacji
- praca z dziennikami scentralizowanymi CloudWatch
- wykonywanie kopii zapasowych danych

Prashant Lakhera jest X-RHCA (Red Hat Certified Architect). Od ponad 15 lat zajmuje się wdrażaniem rozwiązań biznesowych w systemach Linux i z wykorzystaniem oprogramowania open source. Chce mieć pozytywny wpływ na świat i dlatego dzieli się wiedzą z innymi na swojej stronie WWW, we wpisach na blogu i na kanale w serwisie YouTube.

Helion 	KOD KORZYŚCI Sięgnij po więcej! 	
 helion.pl	ISBN 978-83-283-9657-9	
 HELION SA ul. Kościuszki 1c 44-100 Gliwice tel.: 32 230 98 63 helion@helion.pl	 9 788328 396579	
Cena: 79,00 zł		

Packt