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

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.

Spis treści

Część III. Nadawanie infrastrukturze skalowalności i elastyczności

Część IV. Warstwy monitorowania, wskaźników i kopii zapasowych

4

ROZDZIAŁ

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.

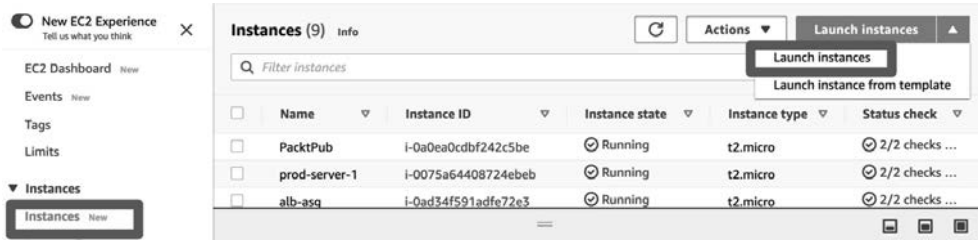
Kolejną zaletą korzystania z chmury jest to, że można w zależności od wymagań łatwo przejść na korzystanie z zasobów wyższej lub niższej klasy. Jeśli na przykład na początku do działania aplikacji wystarczy instancja o jednym procesorze i 1 GB pamięci, to później, kiedy użycie wzrośnie, można przejść (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.

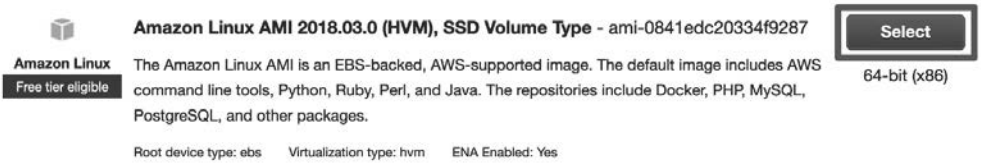
Aby uruchomić instancję w konsoli AWS, musisz wykonać serię kroków:

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 instancje; rysunek 4.1). W górnym prawym rogu następnej strony kliknij przycisk *Opt out to the old experience* (wycofaj się do starego środowiska).



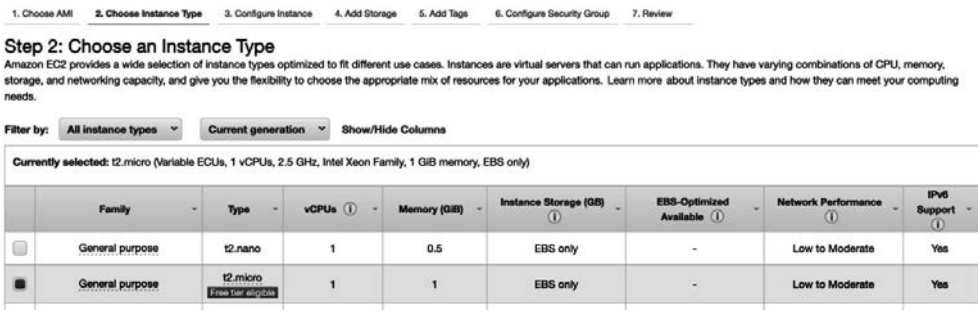
Rysunek 4.1. Uruchamianie pierwszej instancji EC2 w konsoli AWS

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).

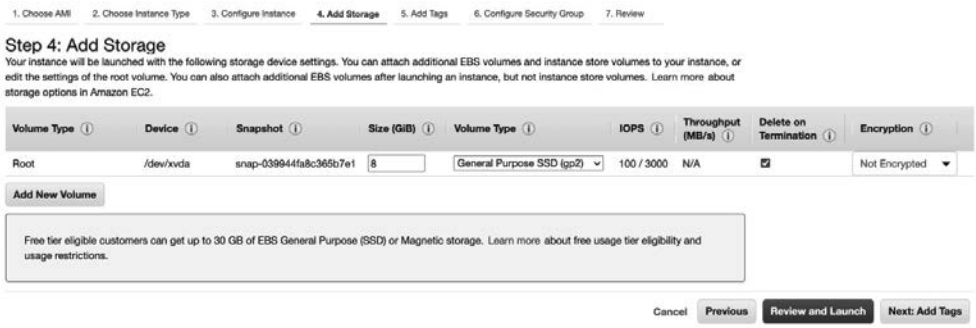


Rysunek 4.3. Wybór typu instancji

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/awsadm.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).

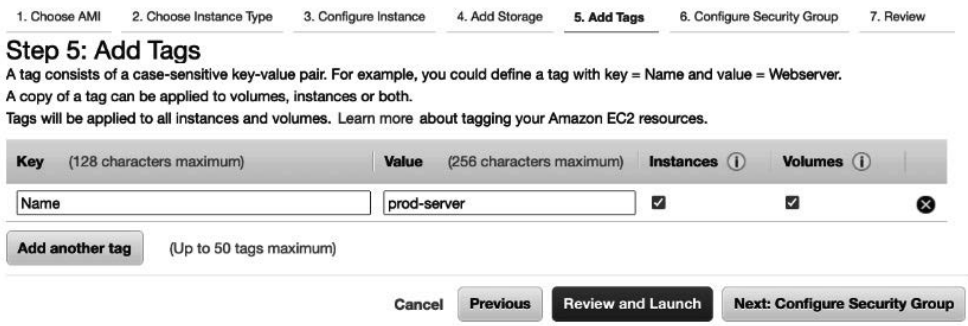
Rysunek 4.4. Konfigurowanie szczegółów instancji

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).



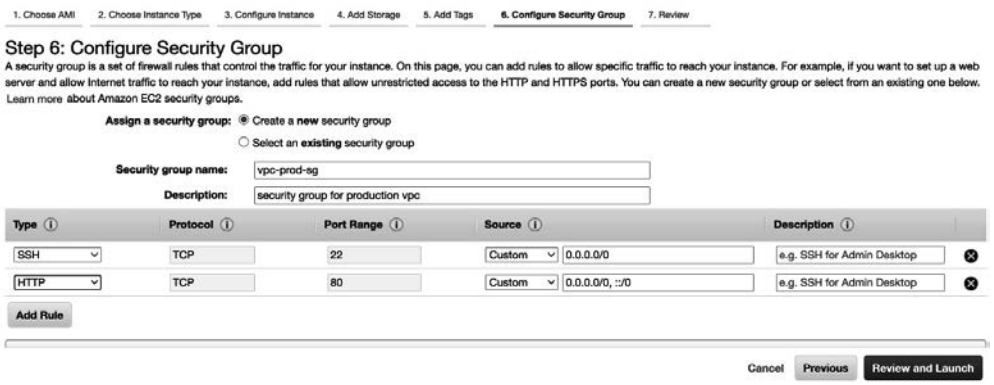
Rysunek 4.5. Dodawanie przestrzeni dyskowej

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).



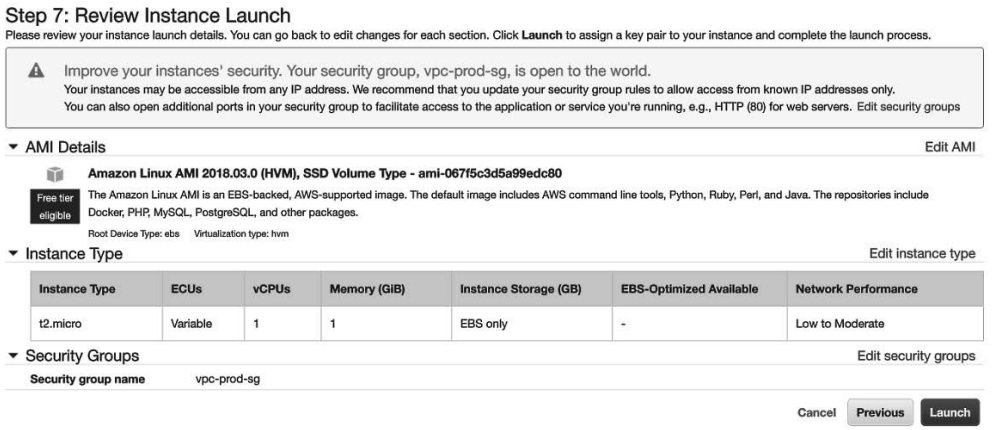
Rysunek 4.6. Dodawanie tagów

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).



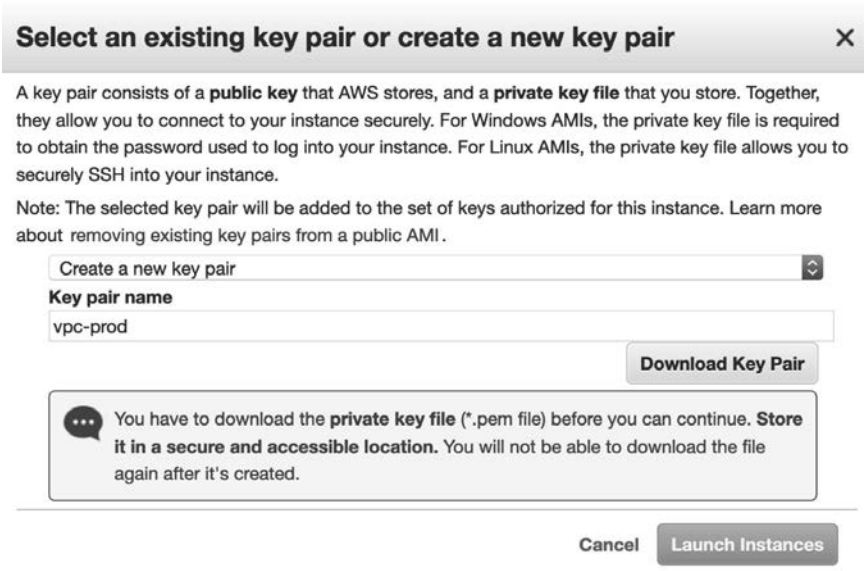
Rysunek 4.7. Konfigurowanie grupy zabezpieczeń

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

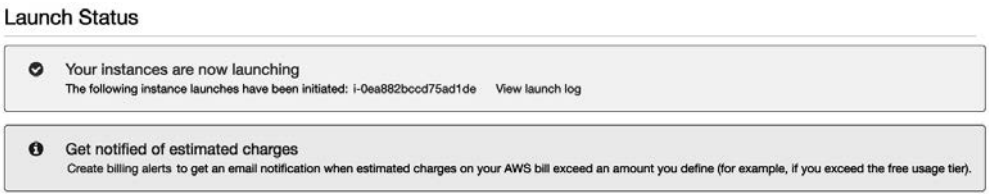


Rysunek 4.8. Przegląd konfiguracji uruchomieniowej

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).



Rysunek 4.9. Tworzenie pary kluczy



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).

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

Wiecie 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. Przeanalizujmy kod CloudFormation i sprawdźmy 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 usludze CloudFormation
    Parameters:
      NetworkStack:
        Type: "String"
        Description: "Tworzenie stosu sieciowego na zasoby"
    Resources:
      SecurityGrouptoallowsshtraffic:
    ```

```
    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 produkycjnego
      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: SecurityGrouptoallowsshtraffic
```

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/awsadm.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 usludze 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-east2:XXXXXXXXXXXX: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

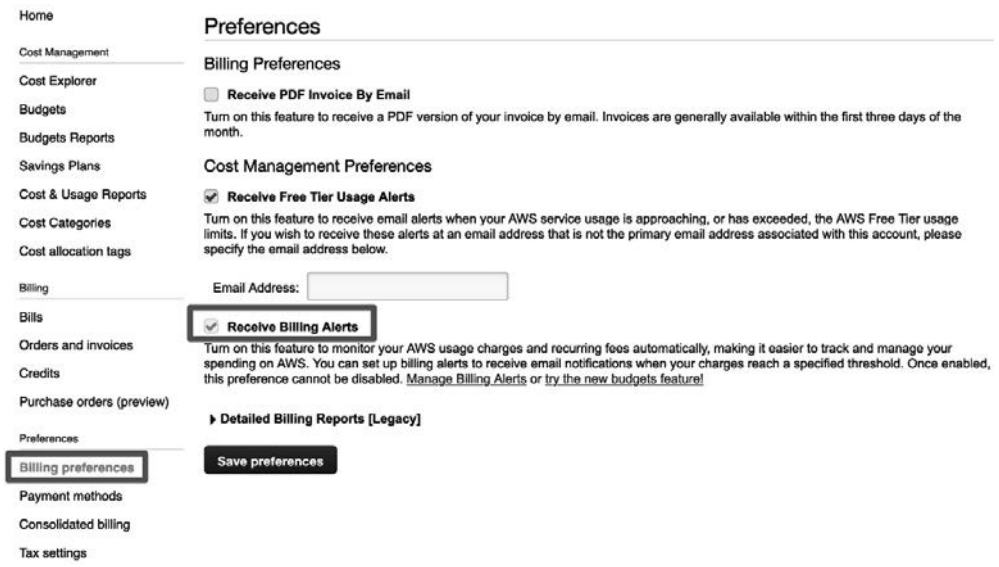
Jednym z najistotniejszych zadań administratora systemów i inżyniera DevOps jest tworzenie alarmu dotyczącego rozliczeń (ang. *billing alarm*), po to by zespół był powiadamiany o przekroczeniu pewnego progu opłat. Może się on też przydać, jeśli ktoś zapomni zamknąć używane

zasoby AWS. W takich przypadkach po przekroczeniu wartości progowej zostanie wyzwolony 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.

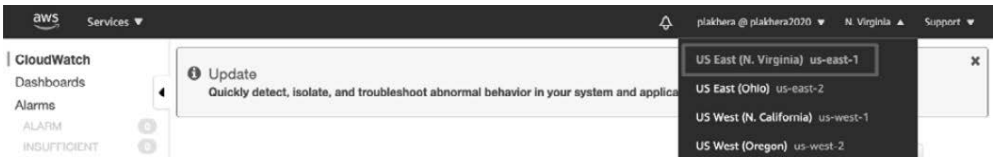
Oto kroki, które trzeba wykonać, by włączyć alarm dotyczący rozliczeń:

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).



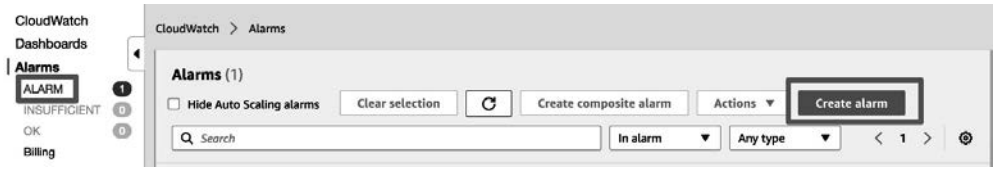
Rysunek 4.13. Konsola rozliczeń AWS

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.



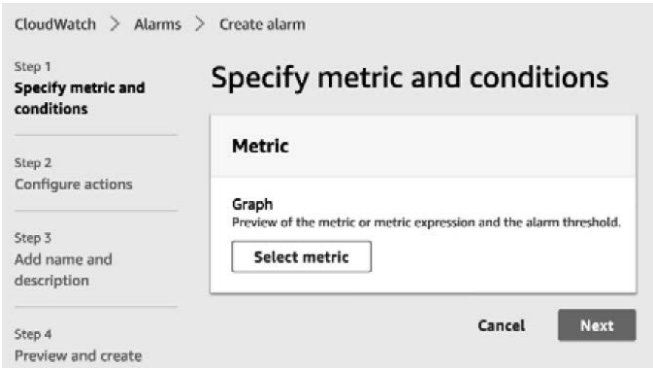
Rysunek 4.14. Wybór regionu N. Virginia do rozliczeń

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



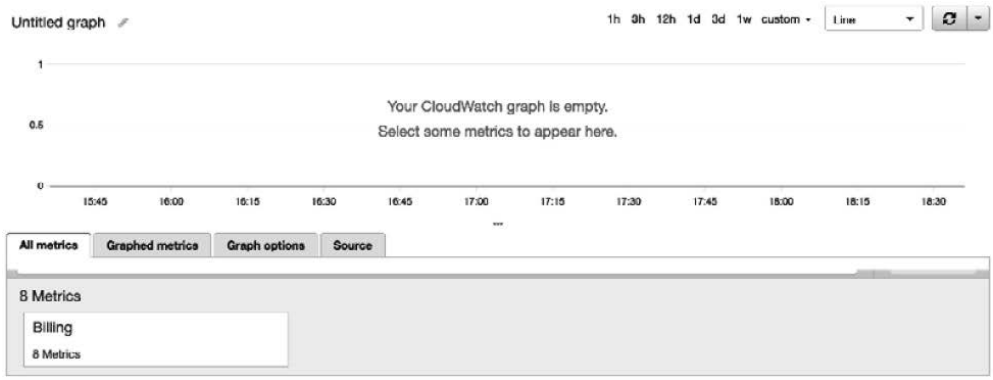
Rysunek 4.15. Tworzenie alarmu CloudWatch

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).

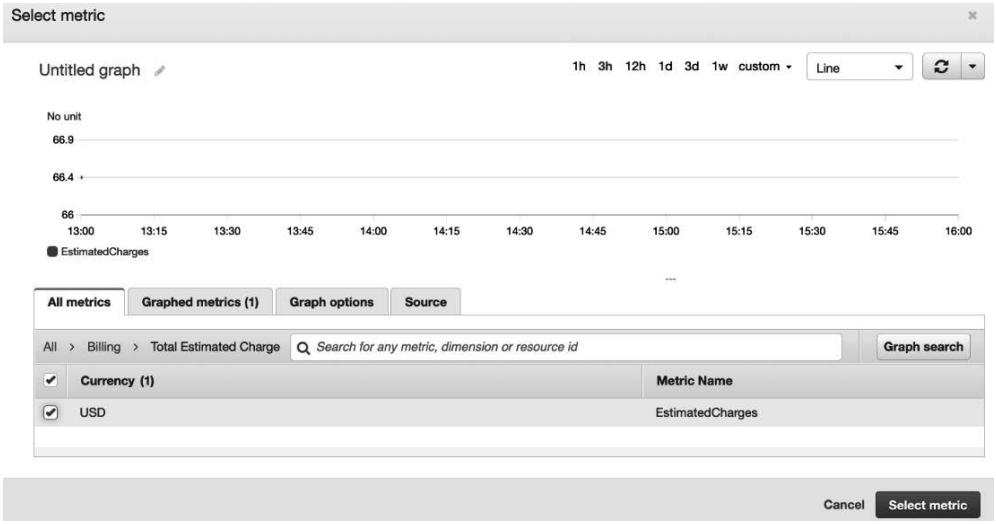


Rysunek 4.17. Wybór wskaźników z grupy Billing

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

Rysunek 4.18. Wybór opcji Total Estimated Charge

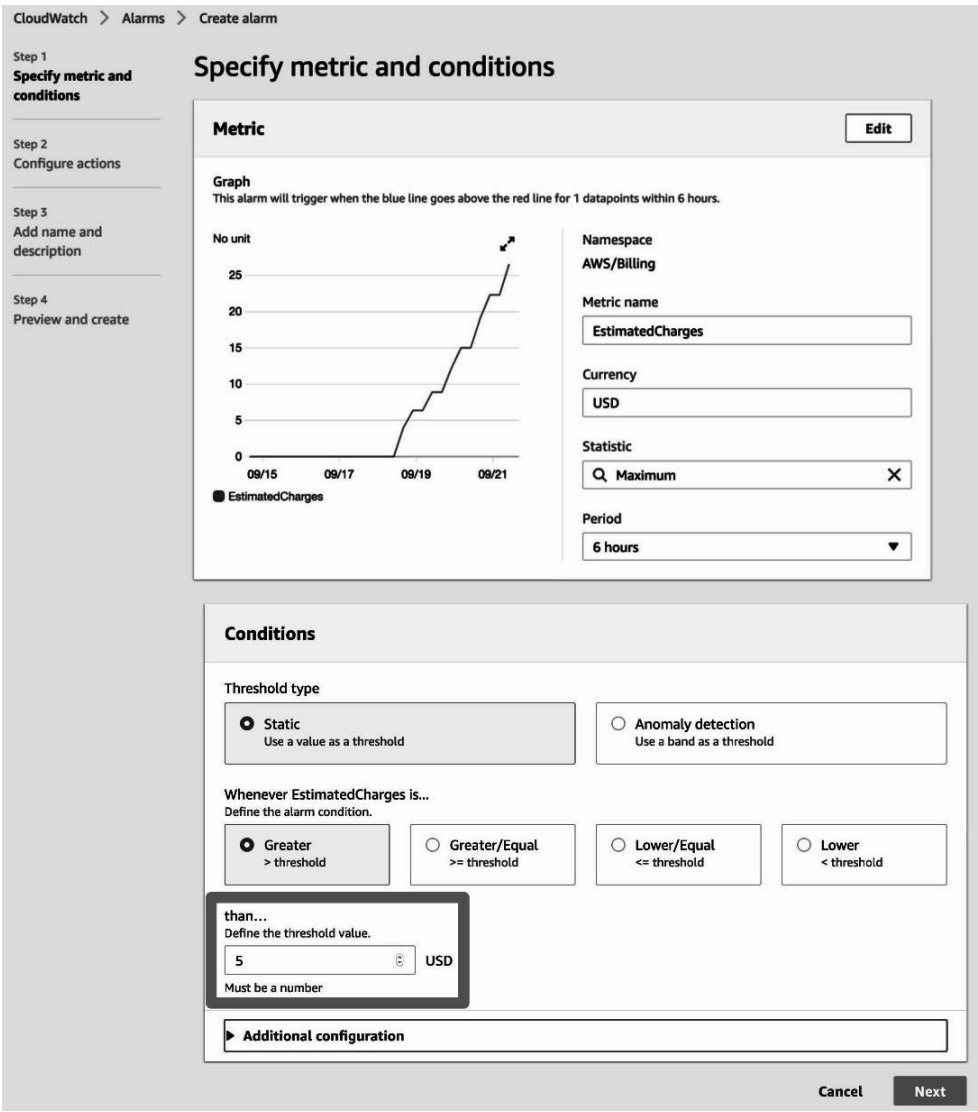
7. Zaznacz pole *USD* [1](dolary amerykańskie), a potem kliknij przycisk *Select metric* (wybierz wskaźnik; rysunek 4.19).



Rysunek 4.19. Wybór dolara jako waluty

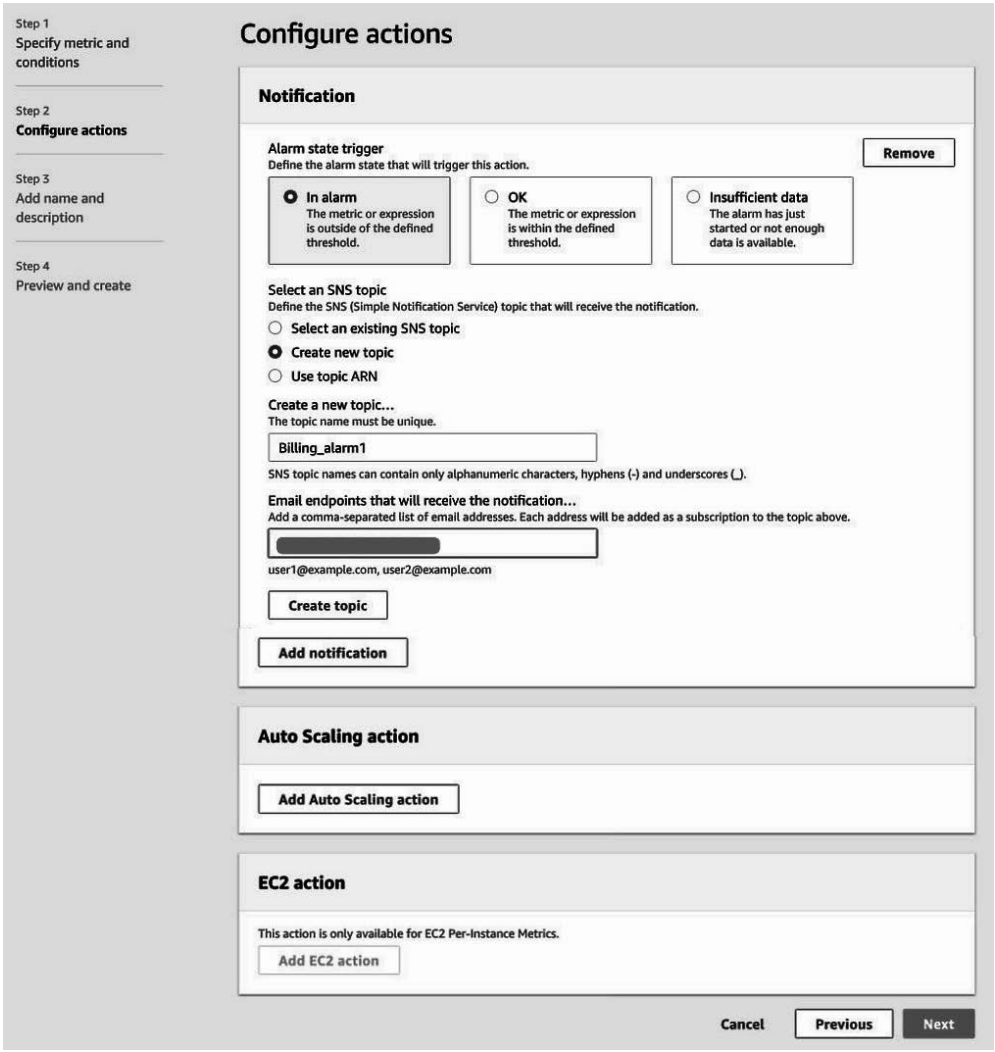
Zachowaj wszystkie ustawienia jako domyślne i wybierz dostosowaną do swoich potrzeb wartość progową (w tym przypadku możesz na przykład wybrać 5 USD, co oznacza, że jeśli wartość rachunku z AWS przekroczy pięć dolarów, otrzymasz powiadomienie). Zmień tę wartość zgodnie ze swoimi preferencjami i kliknij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

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

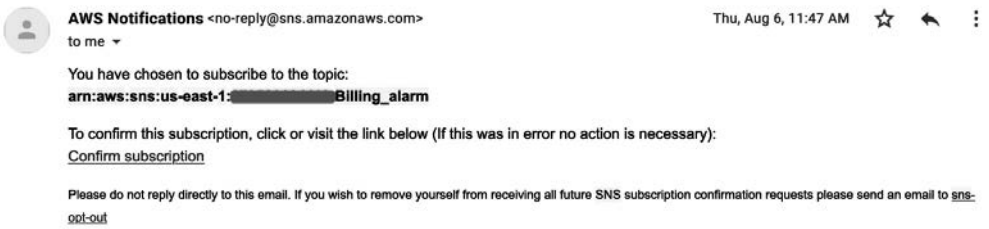


Rysunek 4.20. Wybór wartości progowej rozliczeń

8. 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).
9. 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.

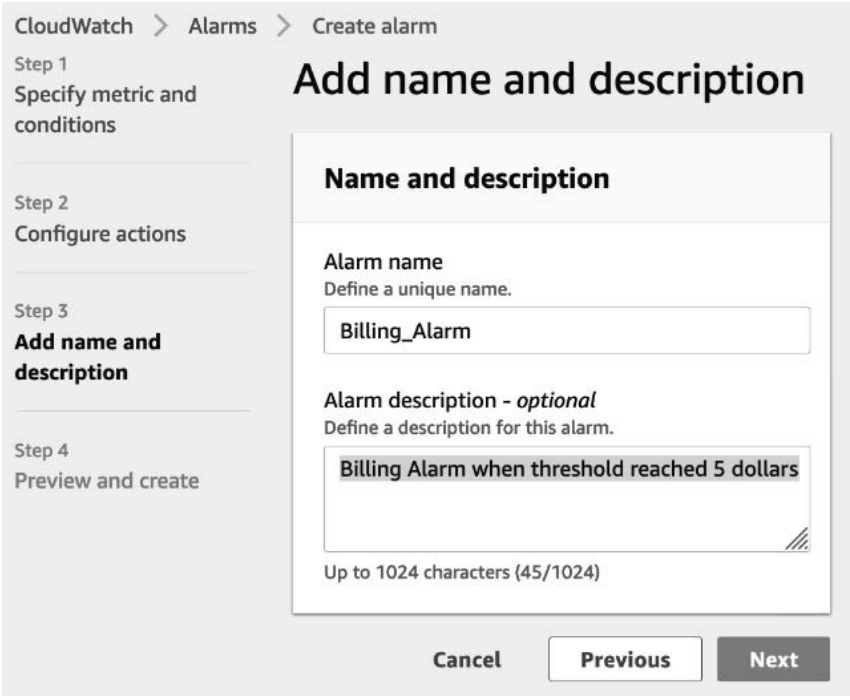


Rysunek 4.21. Temat usługi SNS



Rysunek 4.22. Potwierdzenie subskrypcji w usłudze SNS

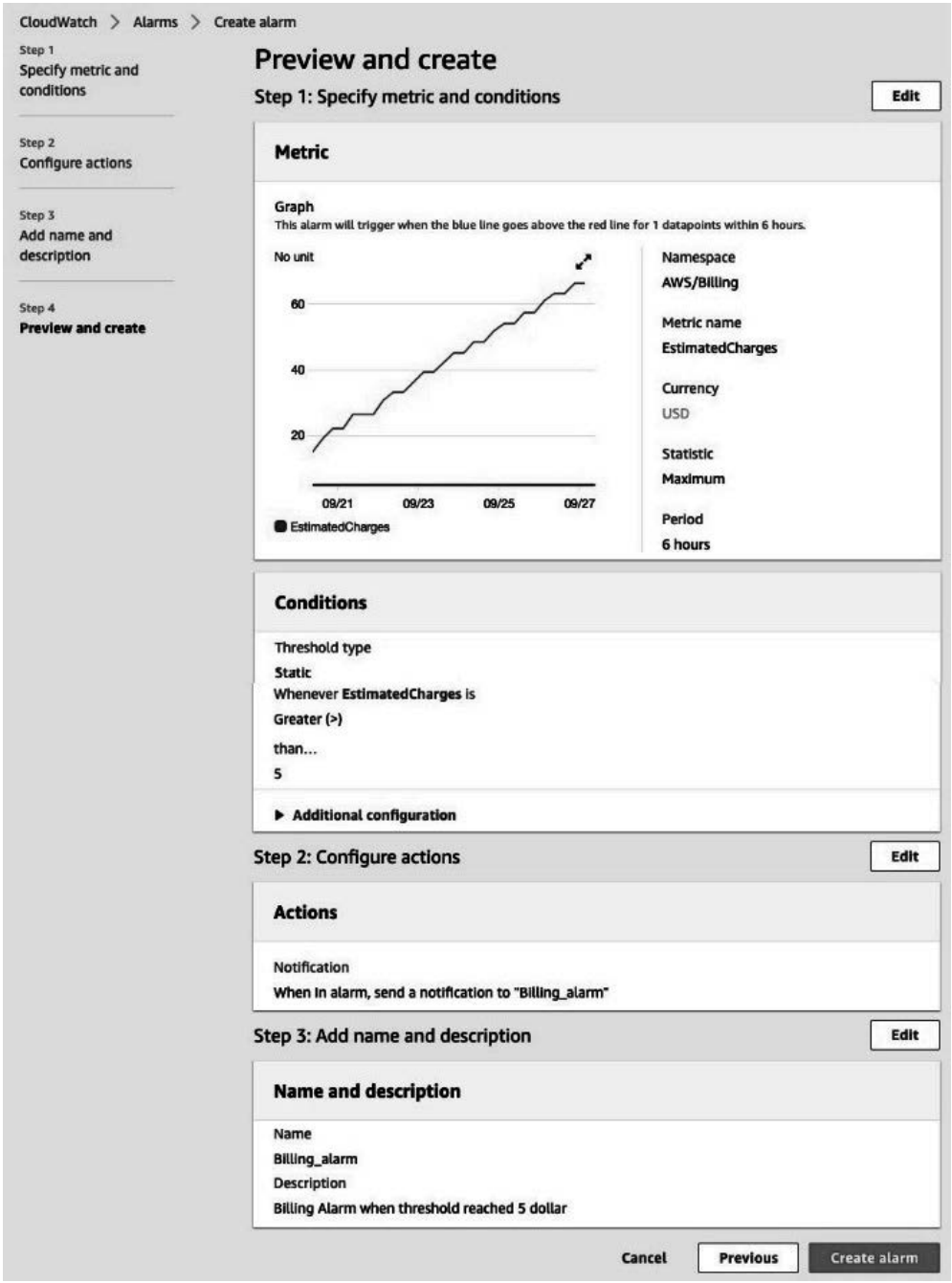
10. Na kolejnym ekranie (rysunek 4.23) uzupełnij pole *Alarm name* (nazwa alarmu), nadając alarmowi sensowną nazwę (na przykład `Billing_Alarm`), i *Alarm description* (opis alarmu), podając zrozumiały opis (na przykład `Billing Alarm when threshold reached 5 dollars` — alarm dotyczący rozliczeń uruchamiany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



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.



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.

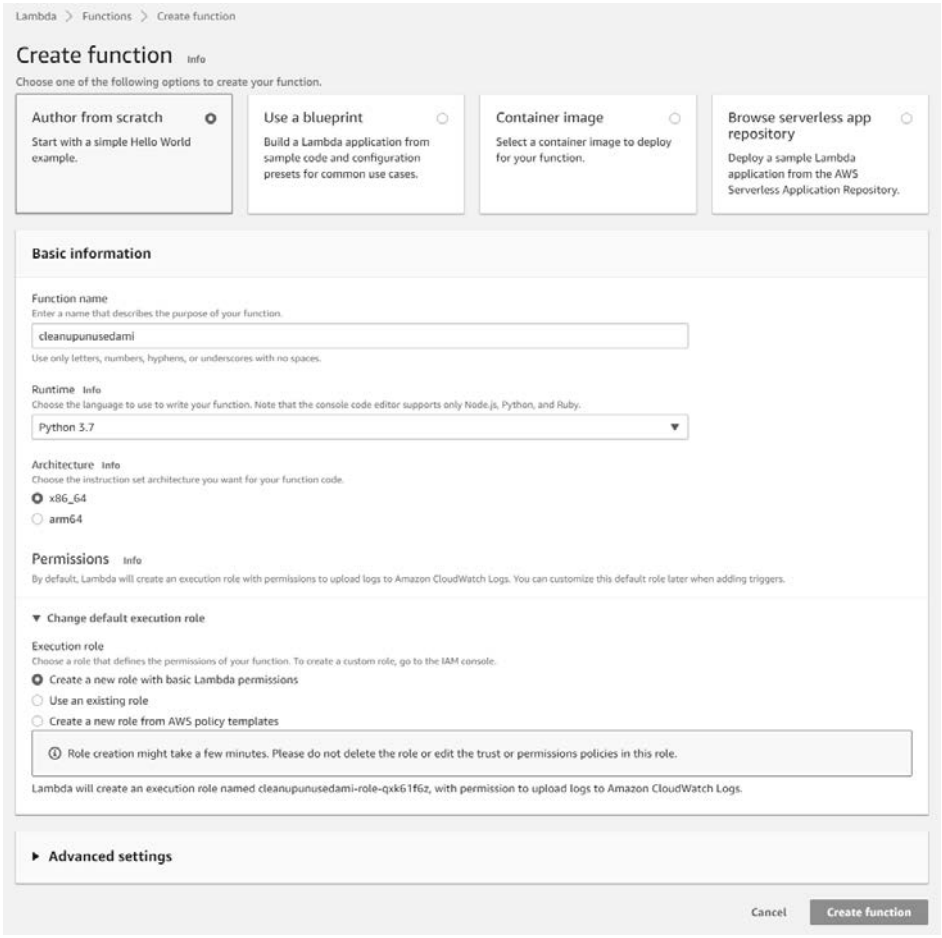
Istnieje kilka sposobów osiągnięcia tego celu (sprzątnięcia obrazów AMI). My zrobimy to, łącząc funkcję Lambda i reguły CloudWatch. Oto kroki, które trzeba wykonać:

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).

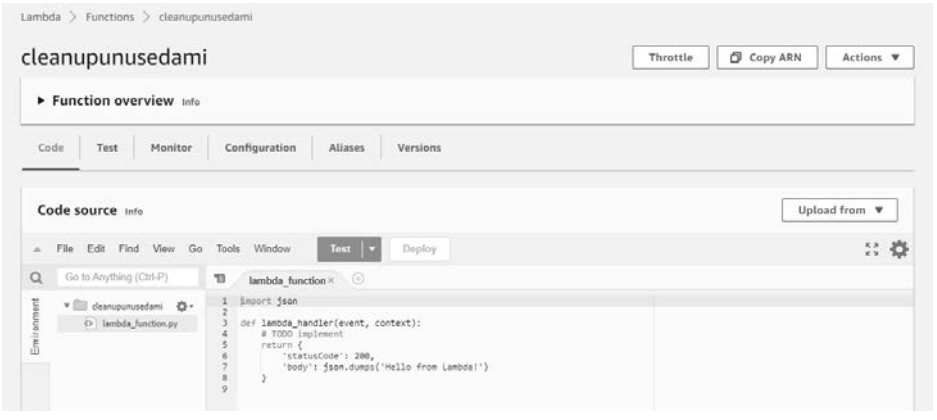
Rysunek 4.25. Konsola AWS Lambda

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).

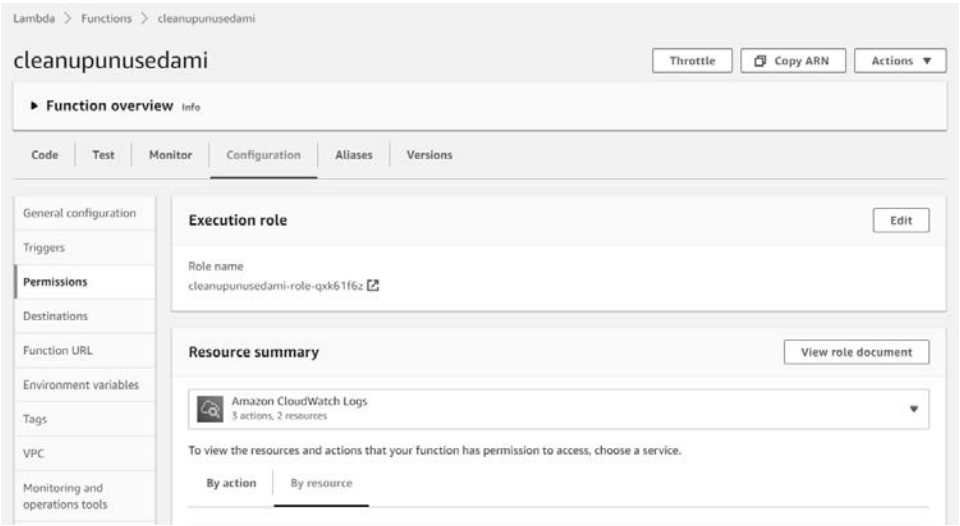
   Kliknij przycisk *Create function* (utwórz funkcję).
3. Na następnym ekranie (rysunek 4.27) kliknij kartę *Configuration* (konfiguracja).
4. Na kolejnym ekranie (rysunek 4.28) w bocznym menu kliknij pozycję *Permissions* (uprawnienia), a potem przycisk *Edit* (edytuj).
5. Kliknij nazwę roli (na przykład `cleanupunusedami-role-8mkhzx5f` — u Ciebie będzie ona inna; rysunek 4.29).



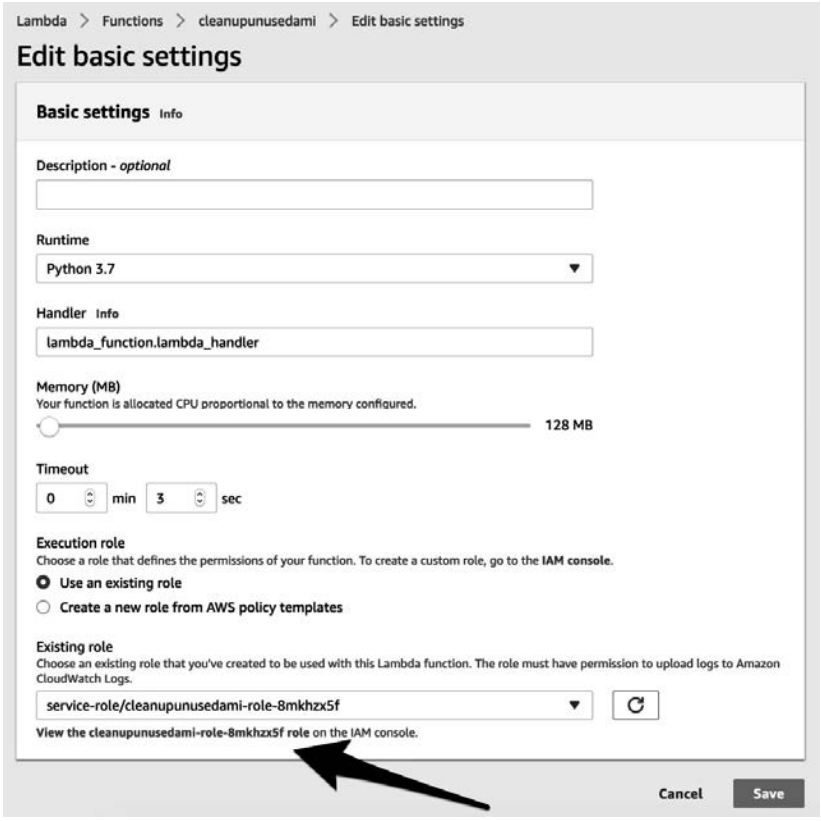
Rysunek 4.26. Tworzenie funkcji Lambda



Rysunek 4.27. Przegląd funkcji Lambda



Rysunek 4.28. Edytowanie uprawnień IAM funkcji Lambda



Rysunek 4.29. Rola IAM funkcji Lambda

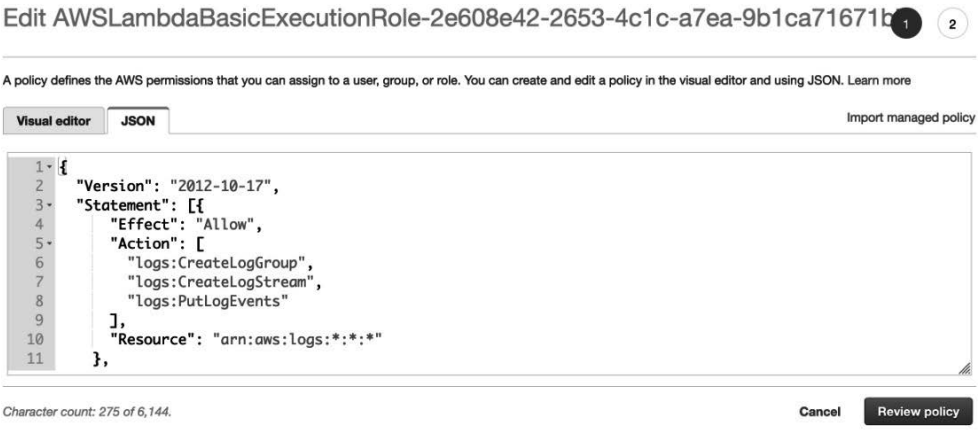
6. Kliknij nazwę zasady (w tym przypadku na przykład `AWSLambdaBasicExecutionRole-2e608e42-2653-4c1c-a7ea-9b1ca71671b`, ale u Ciebie będzie inna), a potem przycisk *Edit policy* (edytuj zasady). Po przejściu do edycji zasad kliknij kartę *JSON* (rysunek 4.30).

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`):

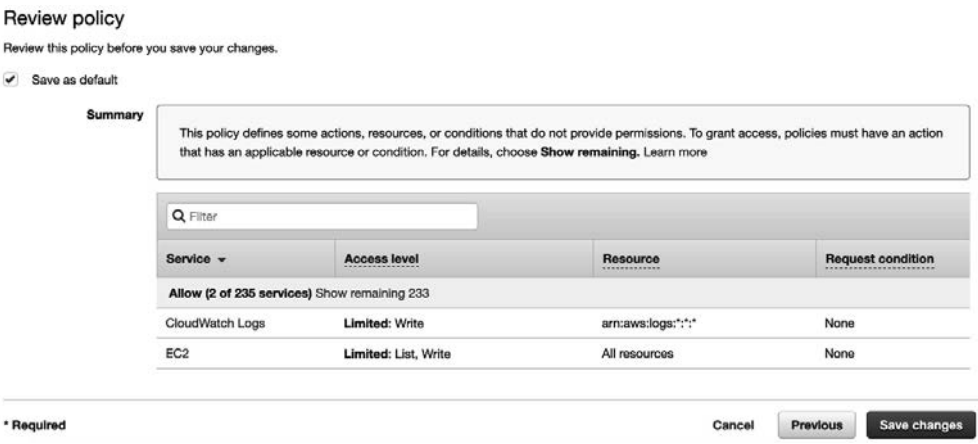
```
{
  "Version": "2012-10-17",
  "Statement": [{
      "Effect": "Allow",
      "Action": [
        "logs:CreateLogGroup",
        "logs:CreateLogStream",
        "logs:PutLogEvents"
      ],
      "Resource": "arn:aws:logs:*:*:*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "ec2:DescribeImages",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

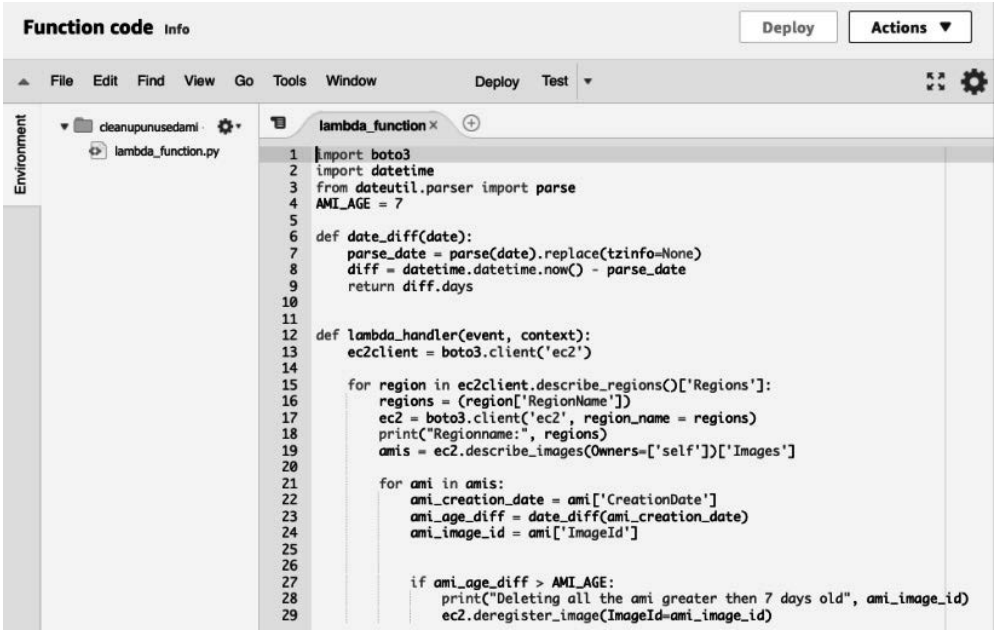


Rysunek 4.32. Zapisywanie zasad IAM funkcji Lambda

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:

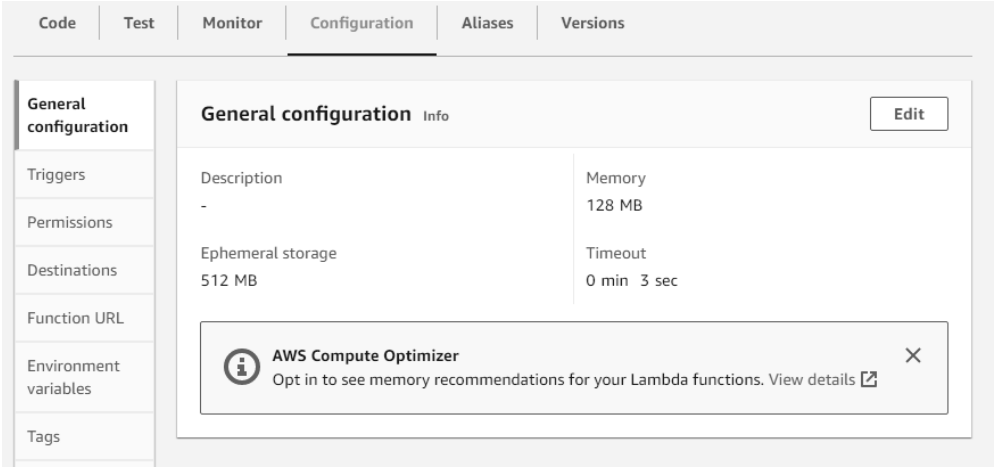
```
amis = ec2.describe_images(Owners=['self'])['Images']

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)
```

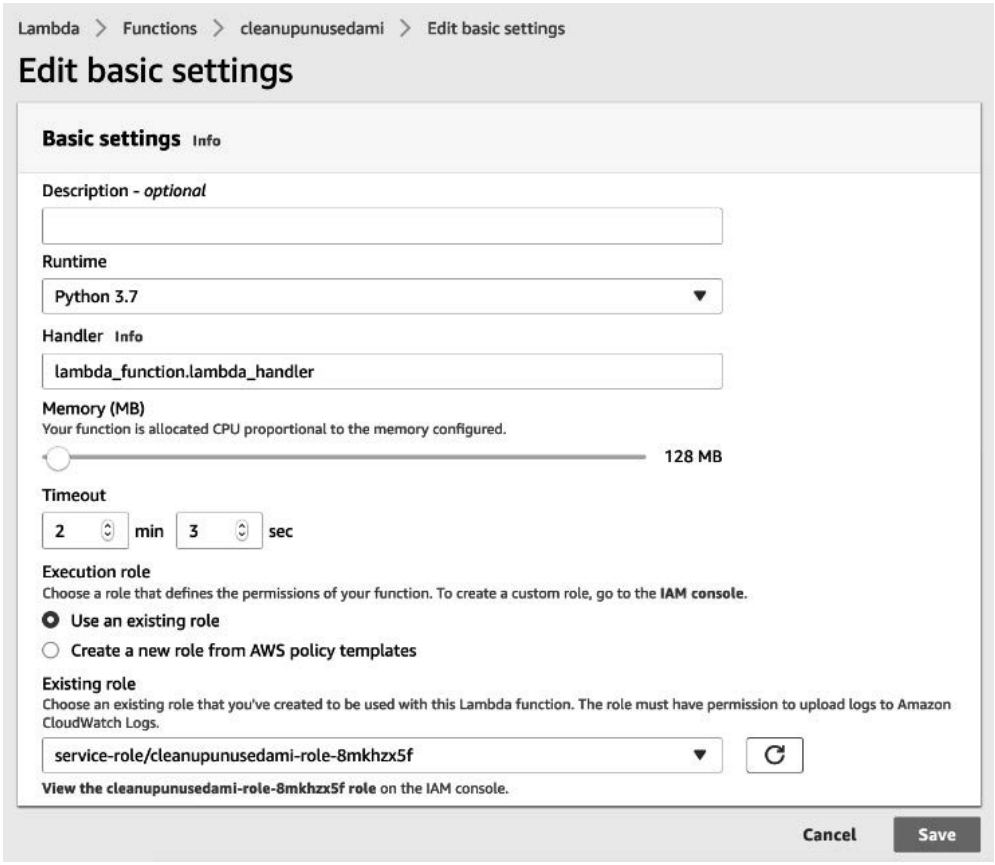
17. W sekcji *General configuration* (ogólna konfiguracja), pokazanej na rysunku 4.35, kliknij przycisk *Edit* (edytuj).



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).

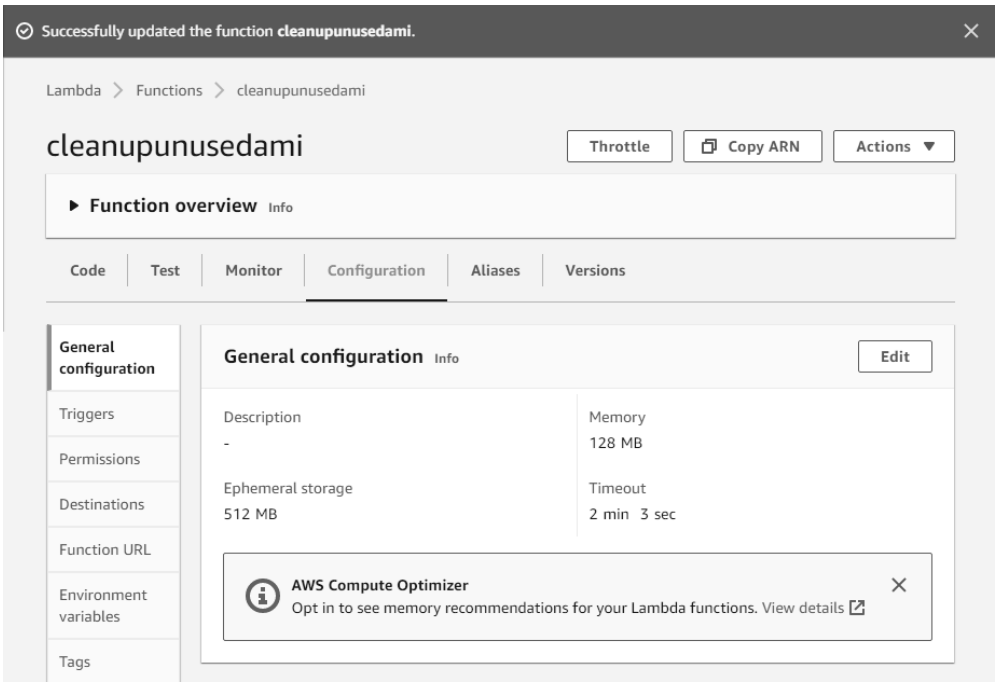


Rysunek 4.36. Ustawienia limitu czasu działania funkcji Lambda

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

Do wyzwolenia funkcji Lambda będzie potrzebne zdarzenie CloudWatch[2]. 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.

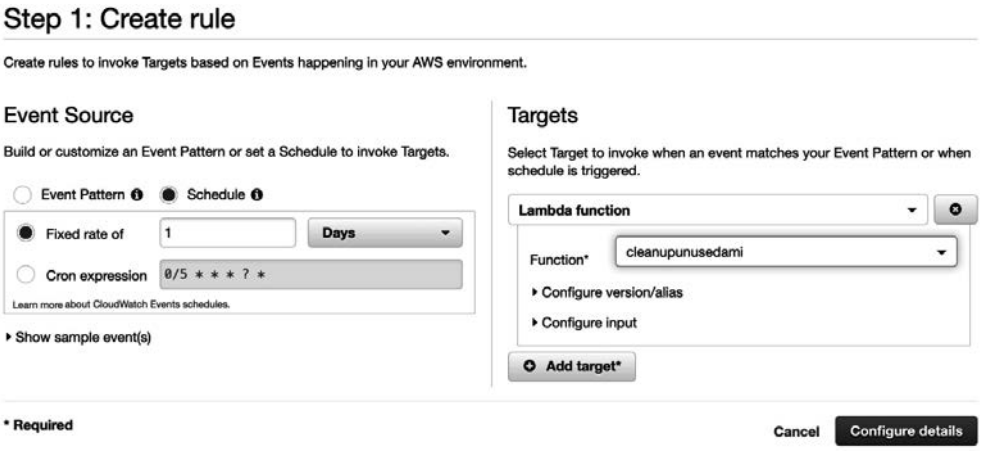
[2] Usługę CloudWatch Events zastępuje usługa EventBridge i jest rekomendowanym sposobem obsługi zdarzeń.— *przyp. tłum.*



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).



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).

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:

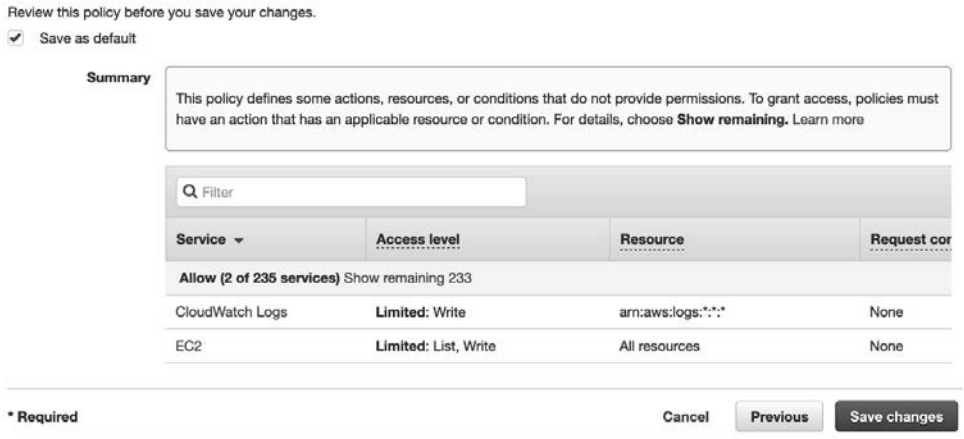
1. Krok ten będzie podobny do tych z wcześniejszego przykładu. Wystarczy, że zmienisz nazwę funkcji, na przykład na `cleanupunattachedebsvol`. Reszta kroków, od 2. do 6., będzie taka sama, ale w kroku 7. zmień zasady IAM na te pokazane poniżej. Zastępujemy domyślne zasady IAM, ponieważ skrypt języka 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`):

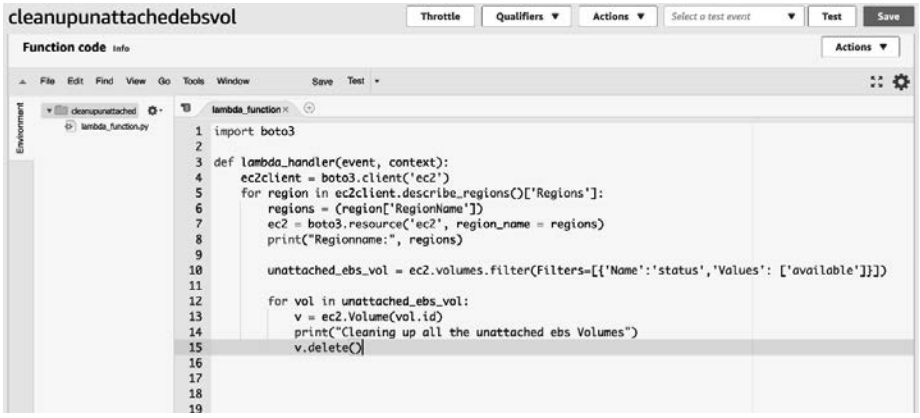
```
{
  "Version": "2012-10-17",
  "Statement": [{
      "Effect": "Allow",
      "Action": [
        "logs:CreateLogGroup",
        "logs:CreateLogStream",
        "logs:PutLogEvents"
      ],
      "Resource": "arn:aws:logs:*:*:*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "ec2:DeleteVolume",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

Rysunek 4.40. Przegląd zasad IAM funkcji Lambda



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):

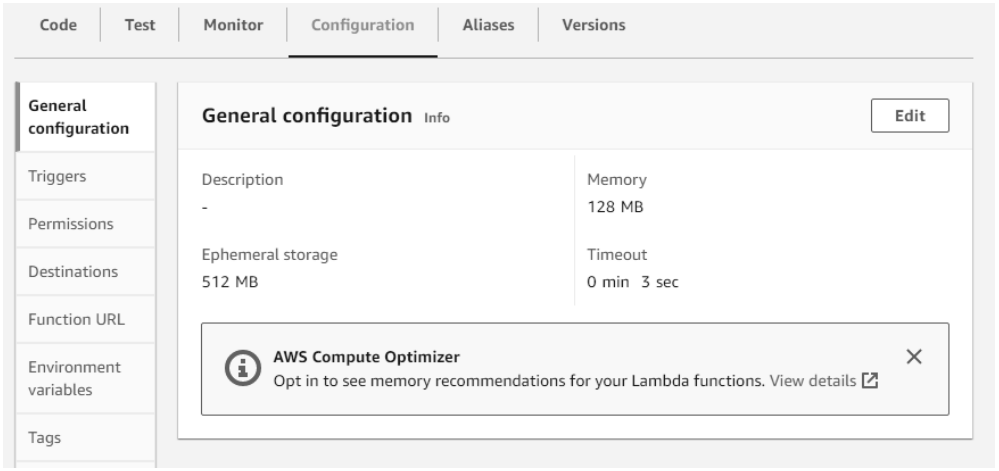
```
ec2client = boto3.client('ec2')

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).

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 zdarzen 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 tworzysz 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 wyzwalał 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 sie 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.

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

helion.pl

HELION SA
ul. Kościuszki 1c
44-100 Gliwice
tel.: 32 230 98 63
helion@helion.pl

KOD KORZYŚCI
Sięgnij po więcej!

ISBN 978-83-283-9657-9

Cena: 79,00 zł

Packt