

# Przetwarzanie w chmurze

## Podsumowanie wdrożenia na stosie usług Amazon Web Services

Łukasz Brewczyński

Filip Dziedzic

Michał Pałka

Rafał Studnicki

Informatyka

Wydział Elektrotechniki, Automatyki, Informatyki i Inżynierii Biomedycznej  
Akademia Górniczo-Hutnicza im. St. Staszica w Krakowie

---

### 1. Wprowadzenie

Jednym z elementów projektu z przedmiotu *Przetwarzanie w chmurze* było wdrożenie przykładowej aplikacji na jednym ze stosów technologicznych udostępniających usługi w chmurze. Wybrany przez nas zestawem usług był **Amazon Web Services**. Z kolei wdrażaną aplikacją był system śledzenia pojazdów składający się z aplikacji udostępniającej logikę biznesową, klienta webowego oraz klienta mobilnego.

### 2. Instancje EC2

Aplikacja do działania potrzebuje maszyny wirtualnej Javy w wersji przynajmniej 1.7 oraz serwera HTTP **Netty**. Jedynym wymaganiem klienta webowego jest serwer HTTP serwujący statyczne pliki, w naszym przypadku był to **nginx**.

Zarówno **JVM** w wdrażaną aplikacją jak i **nginx** zostały uruchomione na dwóch instancjach typu **micro EC2** w regionie EU-West zlokalizowanym w Irlandii.

Na rysunku 1 widać zrzut ekranu z panelu zarządzania uruchomionymi instancjami.

### 3. Elastic Load Balancer

Celem uruchomienia większej liczby instancji jest wysoka dostępność uruchomionej aplikacji. W przypadku awarii jednej z nich, druga może wziąć na siebie obsługę zapytań. W sytuacji normalnego funkcjonowania instancji, *load balancer* decyduje do której instancji przekierować zapytanie, dzięki czemu ruch jest rozłożony na więcej niż jedną maszynę.

W idealnym przypadku zatem instancje powinny być uruchomione w różnych strefach (znajdujących się w tym samym *datacenter*, jednak gwarantowana jest ich fizyczna separacja) a nawet regionach (Amazon posiada kilka *datacenter* w różnych częściach świata). W ten sposób uniezależniamy działanie aplikacji od awarii regionu czy nawet całego datacenter.

Na rysunku 2 została umieszczona lista load balancerów w panelu administracyjnym **Amazon Web Services** z load balancerem uruchomionym na potrzeby aplikacji o nazwie *vehicle*.

W panelu administracyjnym load balancera możliwe jest podglądanie i zmienianie aktualnie działających instancji w obrębie danej instancji load balancera. Przykład tego możliwy jest do zaobserwowania na rysunku 3.

Launch Instance Connect Actions

Filter: All instances All instance types Search Instances 1 to 2 of 2 Instances

Name	Instance ID	Instance Type	Availability Zone	Instance State	Status Checks	Alarm Status	Public DNS
<input checked="" type="checkbox"/>	i-cead798e	t1.micro	eu-west-1a	running	2/2 checks...	None	ec2-54-74-28-209.eu-west-1.compute.amazonaws.com
<input type="checkbox"/>	i-cff82c8f	t1.micro	eu-west-1a	running	2/2 checks...	None	ec2-54-22-...

Instance: i-cead798e Public DNS: ec2-54-74-28-209.eu-west-1.compute.amazonaws.com

Description Status Checks Monitoring Tags

Instance ID	i-cead798e	Public DNS	ec2-54-74-28-209.eu-west-1.compute.amazonaws.com
Instance state	running	Public IP	54.74.28.209
Instance type	t1.micro	Elastic IP	-
Private DNS	ip-10-35-183-118.eu-west-1.compute.internal	Availability zone	eu-west-1a
Private IPs	10.35.183.118	Security groups	all . view rules
Secondary private IPs	-	Scheduled events	No scheduled events
VPC ID	-	AMI ID	amzn-ami-pv-2014.03.1.x86_64-eb3 (ami-2918e35e)
Subnet ID	-	Platform	-
Network interfaces	-	IAM role	-
Source/dest. check	False	Key pair name	tracking
		Owner	700563900662

Figure 1: Podsumowanie uruchomionych instancji EC2

Create Load Balancer Actions

Filter: Search Load Balancers 1 to 1 of 1 Load Balancers

Load B	DNS Name	Port Configuration	Availability	Instance C	Health Check
<input checked="" type="checkbox"/> vehicle	vehicle-2032696057.eu-west-1a	80 (HTTP) forwarding to 80 (...)	eu-west-1a	2 Instances	HTTP:9000/i

Figure 2: Lista instancji ELB w panelu AWS

Load balancer: **vehicle**

Description Instances Health Check Monitoring Security Listeners

Connection Draining: Enabled, 300 seconds (Edit)

Edit Instances

Instance ID	Name	Availability Zone	Status	Actions
i-cead798e		eu-west-1a	InService ⓘ	<a href="#">Remove from Load Balancer</a>
i-cff82c8f		eu-west-1a	InService ⓘ	<a href="#">Remove from Load Balancer</a>

Edit Availability Zones

Availability Zone	Instance Count	Healthy?	Actions
eu-west-1a	2	Yes	-

Figure 3: Spis instancji obsługiwanych przez load balancer *vehicle*

Load balancer nie potrafi dokonać automatycznej oceny czy dana instancja nadaje się do obsługi ruchu przychodzącego, gdyż nawet jeśli instancja jest poprawnie uruchomiona, aplikacja może działać błędnie.

Dlatego też konieczne jest zdefiniowanie własnego interfejsu sprawdzającego czy aplikacja na danym węźle działa poprawnie. Możliwe jest zdefiniowanie jej za pomocą protokołu HTTP, HTTPS, TCP lub SSL. Dodatkowo możliwe jest zdefiniowanie czasu co jaki instancja powinna być sprawdzana, a także liczba błędnych i poprawnych odpytań z rzędu która konieczna jest do stwierdzenia, że aplikacja nie działa lub działa ponownie.

Sposób konfiguracji load balancera w naszej aplikacji został przedstawiony na rysunku 4.

Jak wspomniano zadaniem load balancera jest przekierowywanie zapytań przychodzących do load balancera do działających instancji. Możliwe jest przekierowanie połączeń w protokołach HTTP, HTTPS, TCP lub SSL.

Sposób konfiguracji przekierowań w naszym projekcie został przedstawiony na rysunku 5.

W **Elastic Load Balancerze** możliwe jest skonfigurowanie *lepkości* sesji. W przypadku przekierowań protokołu HTTP lub HTTPS funkcjonalność ta pozwala na przypisanie sesji użytkownika do danej instancji. Jest to konieczne w przypadku, gdy informacje o sesji przechowywane są tylko na docelowej instancji i nie podlegają klastrowaniu.

W **ELB** *lepkość* można skonfigurować na 2 sposoby: przez *cookie* dołączane automatycznie przez load balancer, lub *cookie* dołączane przez aplikację użytkownika. Druga opcja powinna zostać wybrana, jeśli przekazujemy identyfikator sesji właśnie w postaci ciasteczek.

W obu przypadkach klient powinien przy każdym kolejnym zapytaniu użyć zwróconych wartości *cookie*, które tracą ważność po zadanym okresie czasu.

Sposób konfiguracji *lepkości* sesji w naszym projekcie został przedstawiony na rysunku 6.

Load balancer: **vehicle**

Description Instances **Health Check** Monitoring Security Listeners

**Ping Target** HTTP:9000/itineraries

**Timeout** 5 seconds

**Interval** 30 seconds

**Unhealthy Threshold** 2

**Healthy Threshold** 2

Edit Health Check

Figure 4: Reguły sprawdzania stanu instancji przez load balancer *vehicle*

Load balancer: **vehicle**

Description Instances Health Check Monitoring Security **Listeners**

The following listeners are currently configured for this load balancer:

Load Balancer Protocol	Load Balancer Port	Instance Protocol	Instance Port	Cipher	SSL Certificate
HTTP	80	HTTP	80	N/A	N/A
HTTP	9000	HTTP	9000	N/A	N/A

Edit

Figure 5: Reguły przekierowania zapytań do instancji w load balancerze *vehicle*

**Scheme:** internet-facing

**Status:** 2 of 2 instances in service

**Port Configuration:** 80 (HTTP) forwarding to 80 (HTTP)  
Stickiness: Disabled (Edit)

9000 (HTTP) forwarding to 9000 (HTTP)  
Stickiness: LBCookieStickinessPolicy, expirationPeriod='900' (Edit)

**Availability Zones:** eu-west-1a

**Cross-Zone Load Balancing:** Enabled (Edit)

Figure 6: Konfiguracja *lepkości* sesji w load balancerze *vehicle*

## 4. Relational Database Service

Komponentem wymaganym do działania przykładowej aplikacji śledzenia jest relacyjna baza danych. **Relational Database Service** zapewnia silniki bazodanowe dla MySQL, PostgreSQL, Oracle oraz Microsoft SQL Server. Wybrany przez nas silnikiem jest PostgreSQL.

Warto dodać, że **AWS** zapewnia wysoką dostępność usługi **RDS**, dlatego też uruchomienie jednej instancji bazy danych nie jest równoważne z wprowadzeniem do systemu *Single Point of Failure*.

Podsumowanie konfiguracji uruchomionej przez nas instancji relacyjnej bazy danych zostało zaprezentowane na zrzucie ekranowym na rysunku 7.

The screenshot displays the AWS Management Console interface for a PostgreSQL database instance. At the top, there are buttons for 'Launch DB Instance', 'Show Monitoring', and 'Instance Actions'. Below these is a filter section showing 'All Instances' and a search bar. The main table lists the instance 'vehicle-tracking' with details like VPC ID, Multi-AZ status, Class (db.t1.micro), Status (available), and Storage (5 GB). Below the table, the 'Endpoint' is shown as 'vehicle-tracking.crmxbjkieek8.eu-west-1.rds.amazonaws.com:5432' with an 'authorized' status. The configuration details are organized into several sections: 'Configuration Details' (Engine: postgres 9.3.3, DB Name: tracking, Username: tracking, Option Group: default:postgres-9-3, Parameter Group: default.postgres9.3), 'Security and Network' (Availability Zone: eu-west-1b, Security Groups: default (active), Port: 5432), 'Instance and IOPS' (Instance Class: db.t1.micro, IOPS: disabled, Storage: 5GB), 'Availability and Durability' (DB Instance Status: available, Multi AZ: No, Automated Backups: Enabled (1 Day), Latest Restore Time: June 10, 2014 9:32:24 PM UTC+2), and 'Maintenance Details' (Auto Minor Version Upgrade: Yes, Maintenance Window: fri:05:30-fri:06:00, Backup Window: 02:07-02:37). At the bottom, there are buttons for 'Instance Actions', 'Events', 'Tags', and 'Logs'.

DB Instance Identifier	VPC ID	Multi-AZ	Class	Status	Storage
vehicle-tracking		No	db.t1.micro	available	5 GB

**Endpoint:** vehicle-tracking.crmxbjkieek8.eu-west-1.rds.amazonaws.com:5432 (authorized)

**Configuration Details**

- Engine: postgres (9.3.3)
- DB Name: tracking
- Username: tracking
- Option Group(s): default:postgres-9-3 (in-sync)
- Parameter Group: default.postgres9.3 (in-sync)

**Security and Network**

- Availability Zone: eu-west-1b
- Security Groups: default (active)
- Port: 5432

**Instance and IOPS**

- Instance Class: db.t1.micro
- IOPS: disabled
- Storage: 5GB

**Availability and Durability**

- DB Instance Status: available
- Multi AZ: No
- Automated Backups: Enabled (1 Day)
- Latest Restore Time: June 10, 2014 9:32:24 PM UTC+2

**Maintenance Details**

- Auto Minor Version Upgrade: Yes
- Maintenance Window: fri:05:30-fri:06:00
- Backup Window: 02:07-02:37

Figure 7: Instancja bazy danych PostgreSQL w AWS

## 5. Route 53

Ostatnim elementem wdrożenia aplikacji było uruchomienie strefy DNS dla wybranej przez nas domeny w usłudze **Route 53**, działającej jak serwer DNS.

Podobnie jak w przypadku usługi **RDS**, Amazon zapewnia wysoką dostępność usługi i nie ma konieczności wprowadzania nadmiarowości w celu zapewnienia wysokiej dostępności na własną rękę.

Lista wpisów DNS została przedstawiona na rysunku 8. Jak można zauważyć, wpis A domeny wskazuje na uruchomiony przez nas wcześniej load balancer.

The screenshot shows the AWS Route 53 console interface for the hosted zone 'trackme.malopolska.pl'. On the left, a table lists three record sets:

Name	Type	Value
trackme.malopolska.pl.	A	ALIAS vehicle-
trackme.malopolska.pl.	NS	ns-382.awsdns, ns-1733.awsdns, ns-613.awsdns, ns-1432.awsdns
trackme.malopolska.pl.	SOA	ns-382.awsdns

The 'A' record is selected. The right-hand pane shows the 'Edit Record Set' details:

- Name:** trackme.malopolska.pl.
- Type:** A – IPv4 address
- Alias:** Yes (selected)
- Alias Target:** vehicle-2032696057.eu-west-1.elb.amaz
- Alias Hosted Zone ID:** Z32O12XQLNTSW2
- Routing Policy:** Simple
- Evaluate Target Health:** No (selected)

Figure 8: Konfiguracja strefy DNS w **Route 53**

## 6. Podsumowanie

Stos usług **Amazon Web Services** pozwala na uruchomienie wymaganych przez nas do wdrożenia aplikacji usług w bardzo łatwy sposób. Wszystkie funkcjonalności dostępne są z konsoli webowej (sposób używany przez nas) lub API REST.

Aplikacja kliencka będąca efektem wdrożenia dostępna jest pod adresem <http://trackme.malopolska.pl/>.