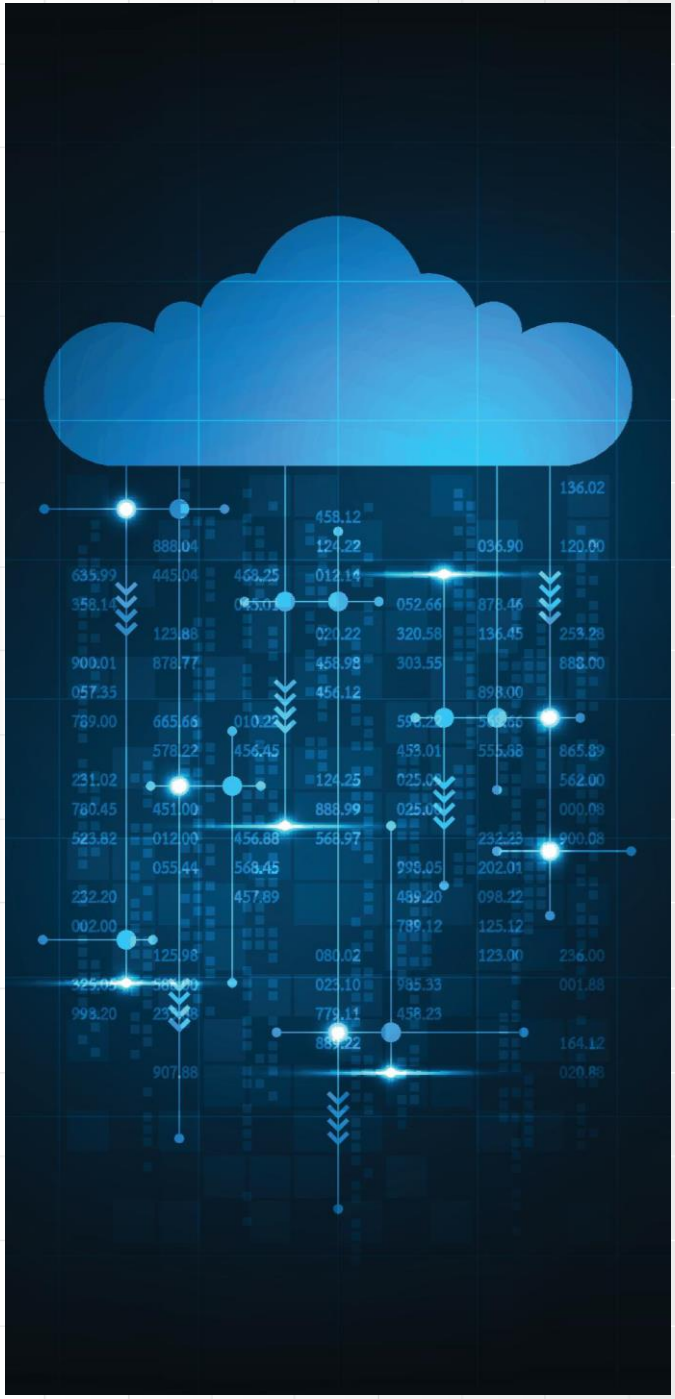


Programowanie w chmurze

Rafał Palak

Politechnika Wrocławska

Wrocław
University
of Science
and Technology

Przekazywanie danych

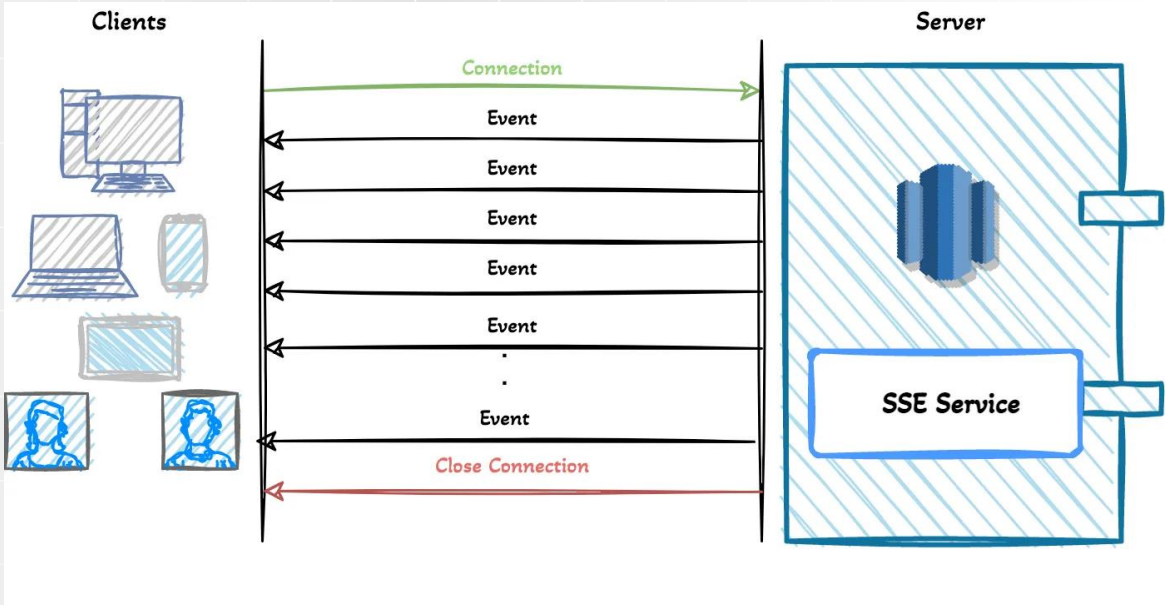
Sposoby przekazywanie danych

- HTTP Polling
- Long Polling
- Server-Sent Events (SSE)
- WebSockets

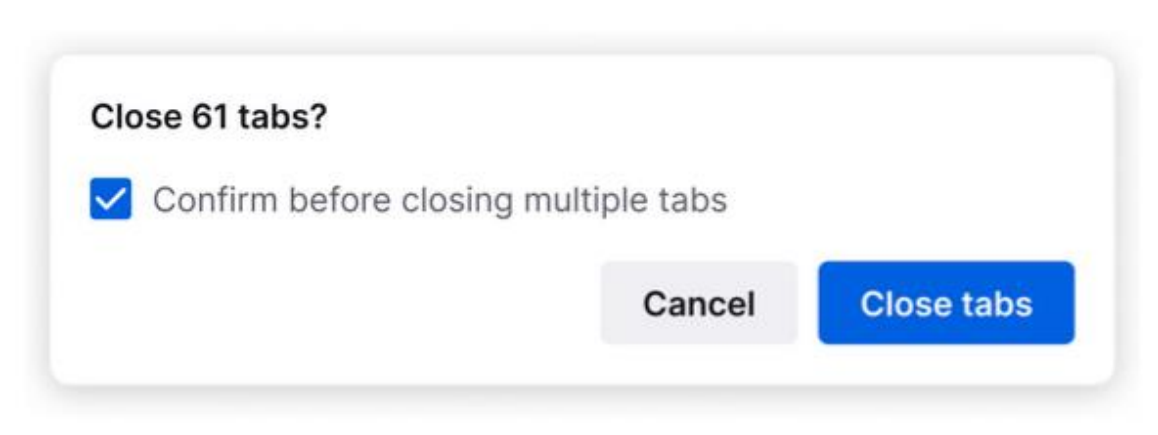
```java
import org.springframework.http.MediaType;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RestController;
import org.springframework.web.servlet.mvc.method.annotation.SseEmitter;

import java.io.IOException;
import java.util.concurrent.CopyOnWriteArrayList;

@RestController
public class SseController {

    private final CopyOnWriteArrayList<SseEmitter> emitters = new CopyOnWriteArrayList<>();

    @GetMapping("/subscribe")
    public SseEmitter subscribe() {
        SseEmitter emitter = new SseEmitter(Long.MAX_VALUE);
        this.emitters.add(emitter);

        emitter.onCompletion(() -> this.emitters.remove(emitter));
        emitter.onTimeout(() -> this.emitters.remove(emitter));

        return emitter;
    }

    // Metoda do wysyłania aktualizacji do wszystkich subskrybentów
    public void sendUpdateToClients(String update) {
        for (SseEmitter emitter : this.emitters) {
            try {
                emitter.send(update, MediaType.APPLICATION_JSON);
            } catch (IOException e) {
                emitter.completeWithError(e);
            }
        }
    }
}
```

Server-Sent Events (SSE)

- Komunikacja jednokierunkowa



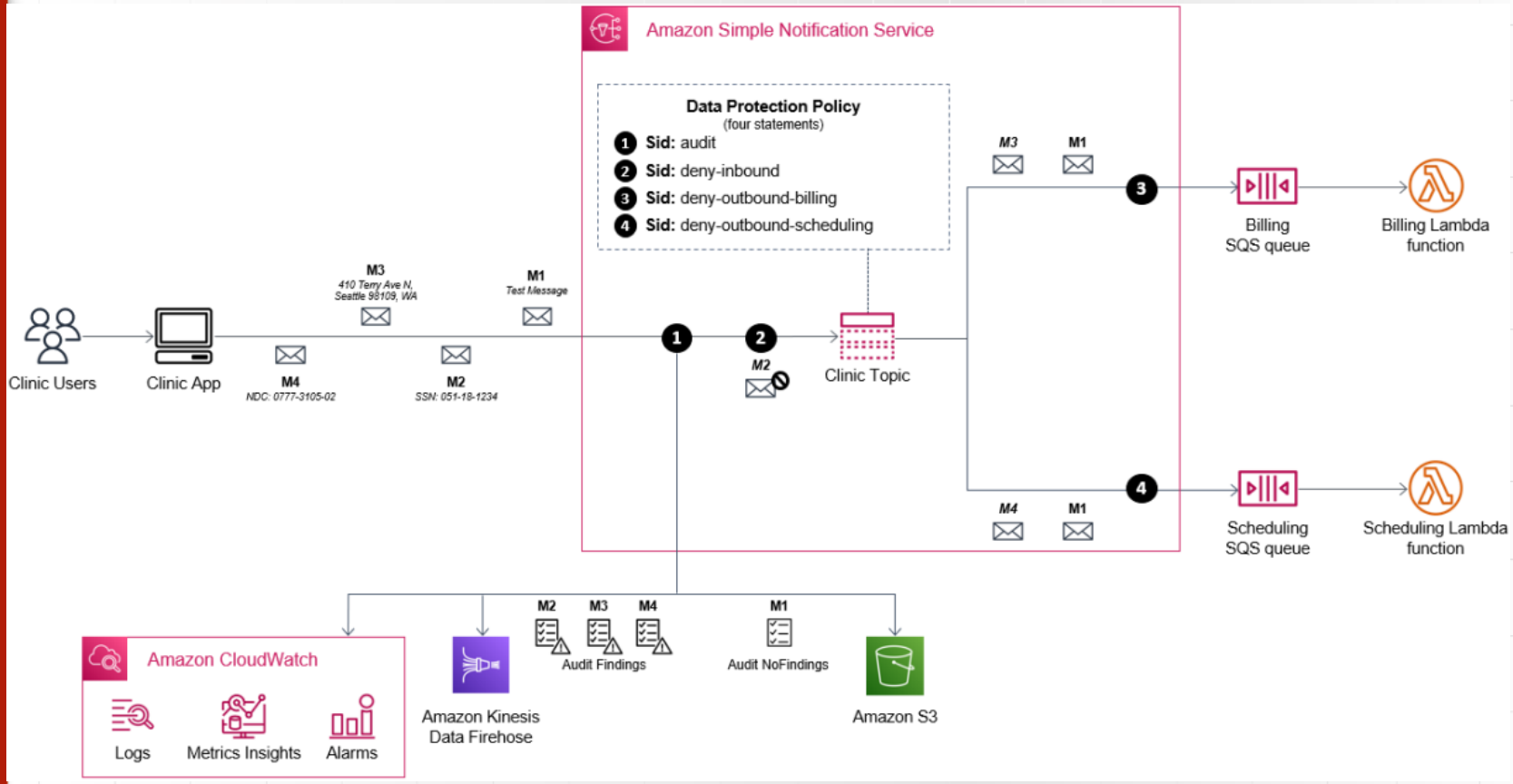
Co zrobić gdy karta przeglądarki została zamknięta?



AWS SNS- działanie

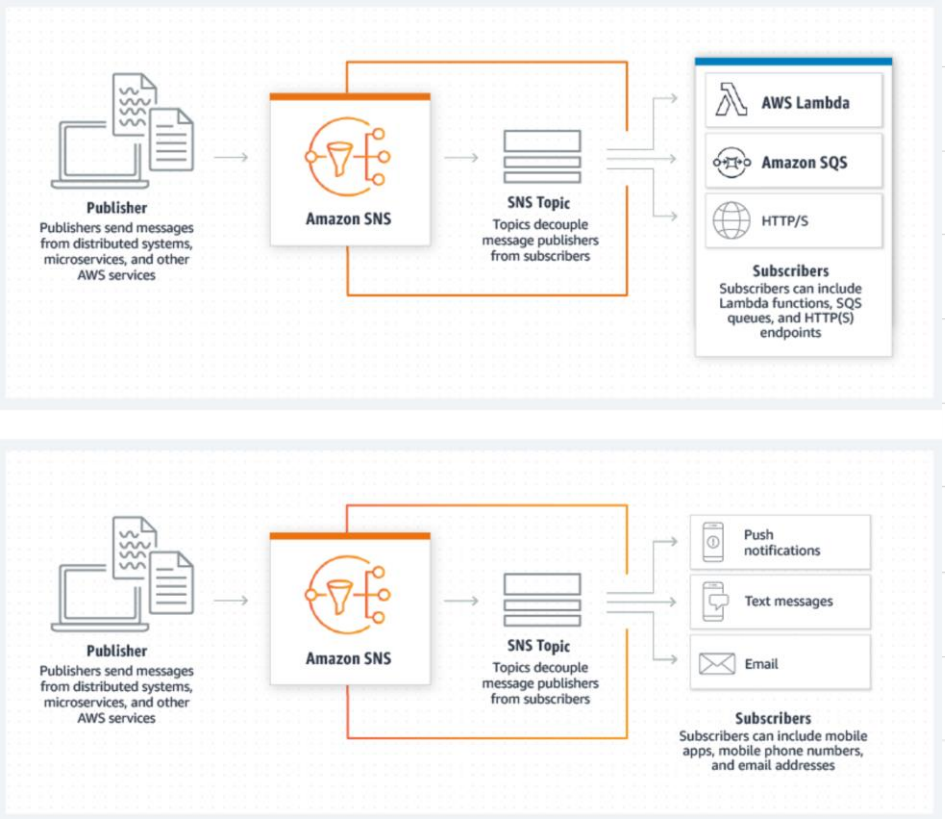
- Tematy (topics)
- Subskrybenci (subscribers)

AWS SNS- bezpieczeństwo



AWS SNS

- Tworzenie Tematów
- Zarządzanie Subskrypcjami
- Filtrowanie Wiadomości
- Różne Typy Subskrypcji - HTTP/S, E-mail, SMS, AWS SQS
- Skalowalność i Elastyczność
- Proste i Elastyczne Integracje
- Bezpieczeństwo
- Wsparcie dla Wiadomości Strukturalnych
- Wieloplatformowość

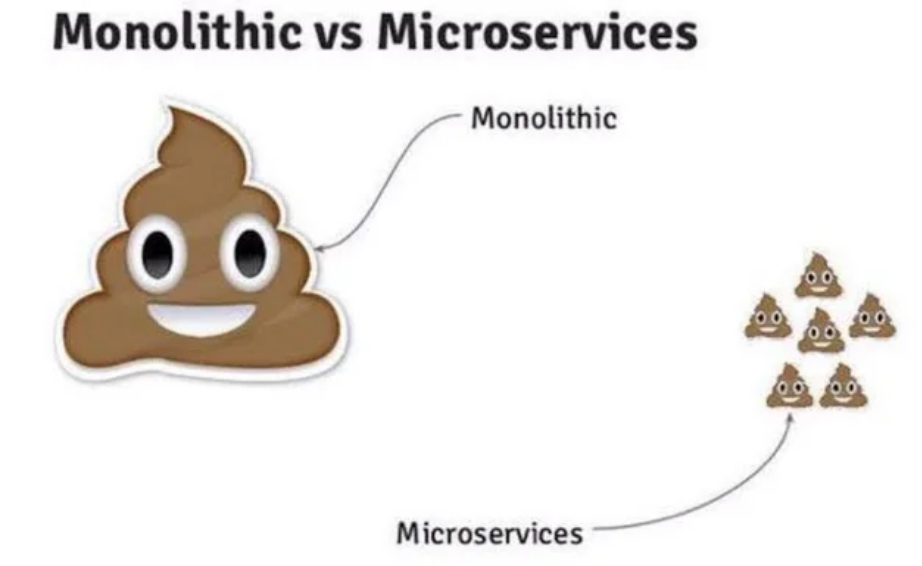


Wrocław
University
of Science
and Technology

Lambda

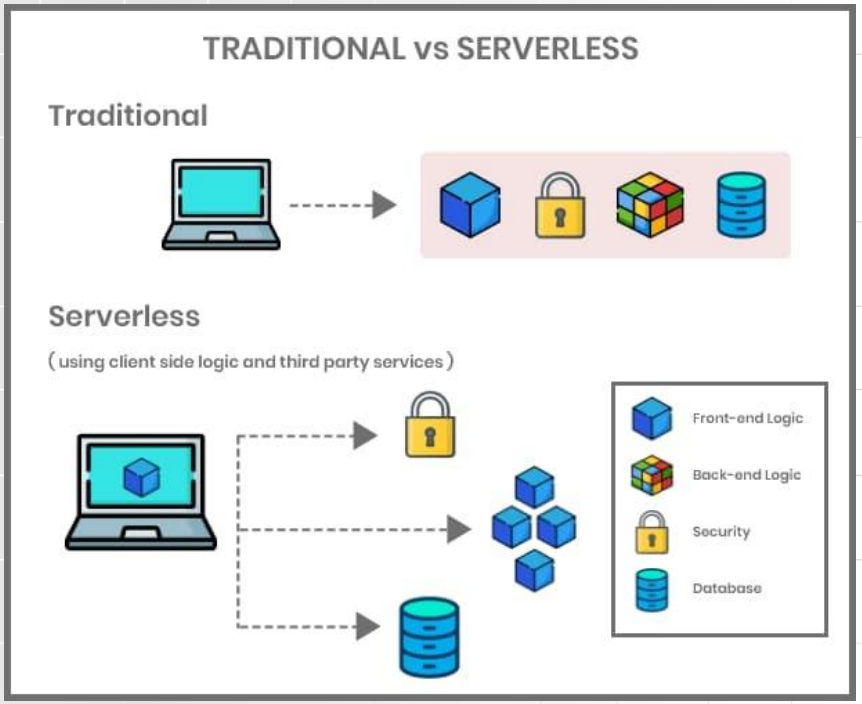
Architektura mikroserwisowa

- Styl architektoniczny, który tworzy strukturę aplikacji jako zbiór luźno połączonych ze sobą niewielkich serwisów komunikujących się poprzez lekkie protokoły komunikacyjne
- Celem jest zapewnienie niezależności poszczególnych komponentów
- Komponenty mogą być rozwijane niezależnie od pozostałych elementów składowych systemu
- Wyraźny podział komponentów tak, by realizowały jedną, konkretną logikę biznesową lub programową.



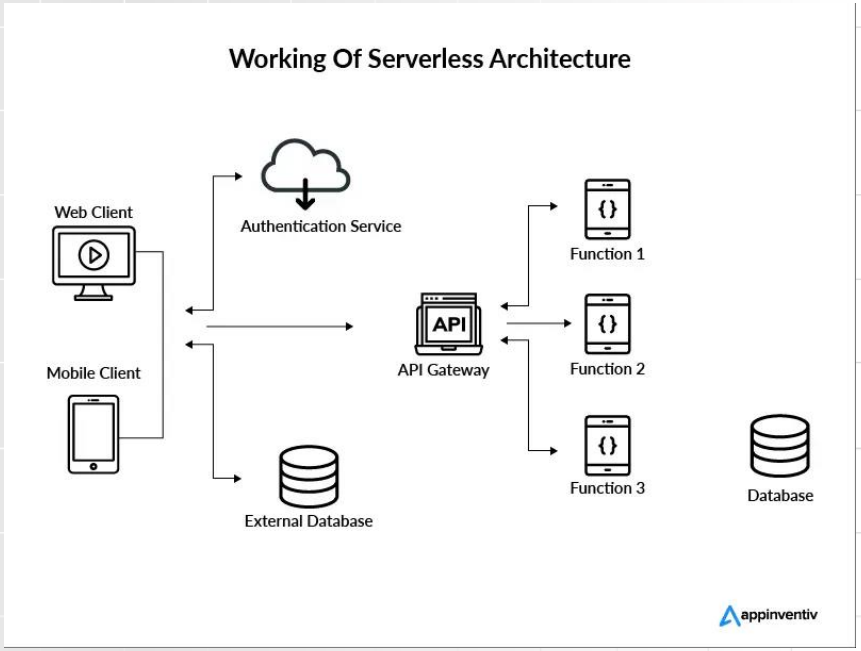
Architektura serverless

- Sposób na tworzenie i uruchamianie aplikacji i usług bez konieczności zarządzania infrastrukturą
- Aplikacja nadal działa na serwerach, ale całe zarządzanie serwerem jest wykonywane przez AWS
- Nie ma konieczności udostępniać, skalować i utrzymywać serwerów, aby uruchamiać aplikacje, bazy danych i systemy pamięci masowej



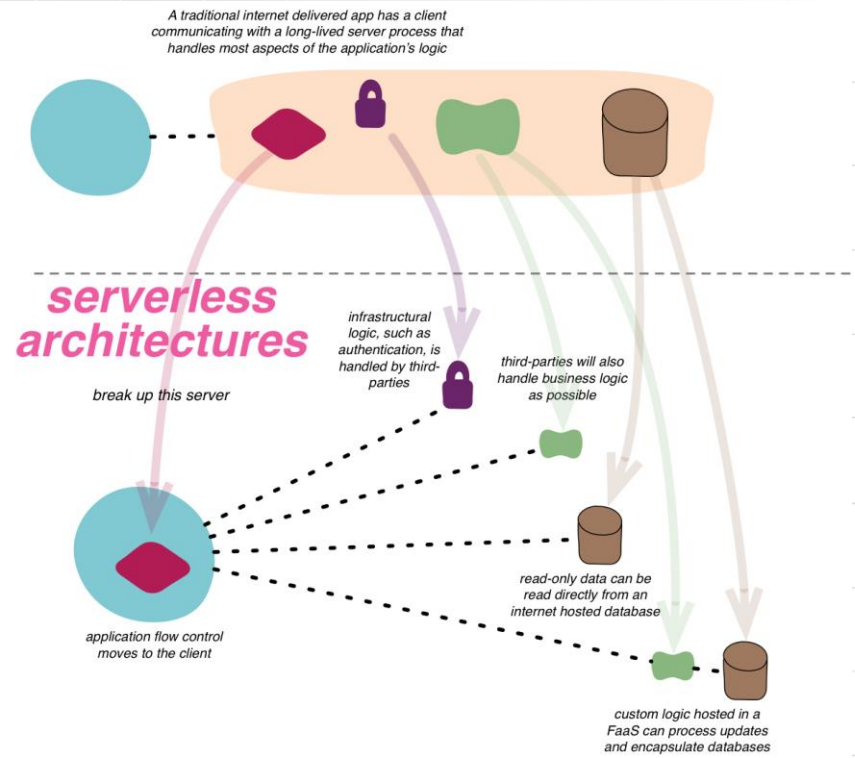
Lambda [1]

- Pozwala uruchamiać kod bez udostępniania lub zarządzania serwerami
- Płatność tylko za zużyty czas obliczeniowy — nie ma opłat, gdy Twój kod nie jest uruchomiony
- Płatność dopiero po pierwszym milionie żądań miesięcznie na AWS Free Tier
- Dzięki Lambdzie można uruchamiać kod dla praktycznie każdego typu aplikacji lub usługi (wszystko to bez konieczności administrowania).



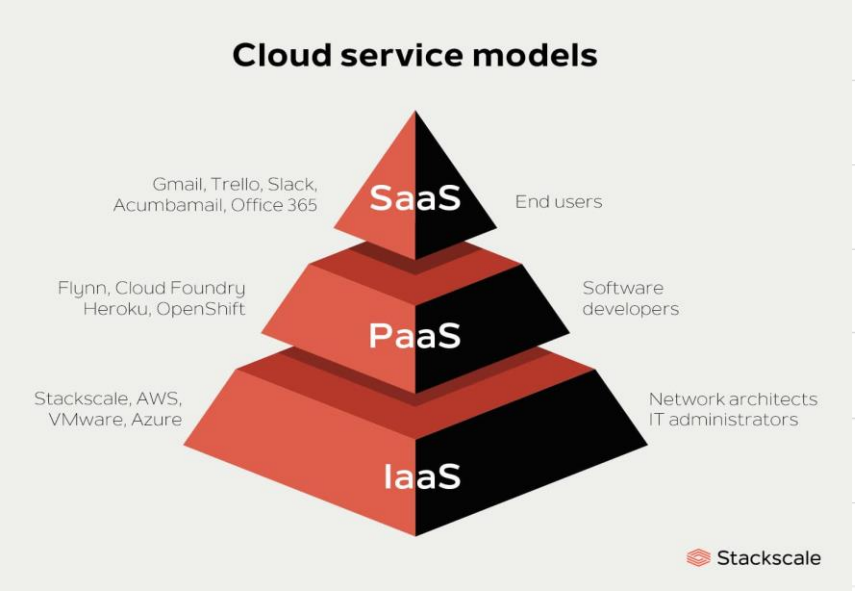
Lambda [2]

- Przesyłamy kod, a Lambda zajmuje się wszystkim, co jest wymagane do uruchomienia i skalowania kodu z wysoką dostępnością.
- Można skonfigurować kod tak, aby był automatycznie inicjowany z innych usług lub zdarzeń AWS, lub można go skonfigurować tak, aby odpowiadał bezpośrednio na żądanie HTTP lub HTTPS.



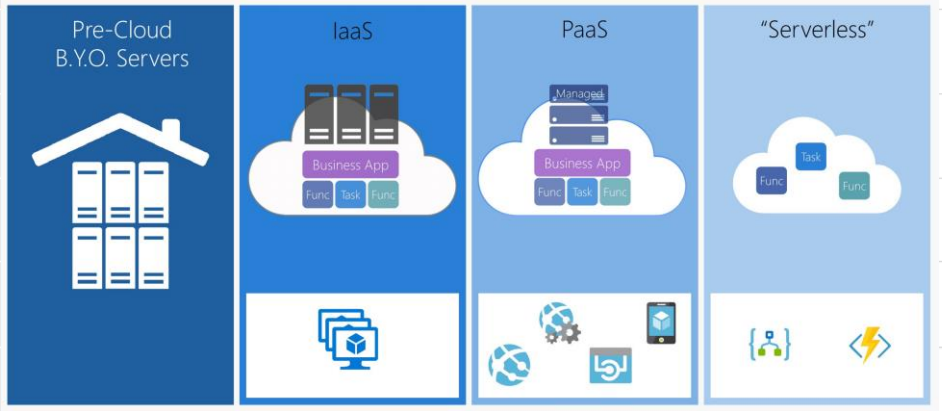
SaaS, PaaS czy IaaS?

- Lambda zaliczana jest do:
- Oprogramowanie jako usługa (SaaS)
- Platforma jako usługa (PaaS)
- Infrastruktura jako usługa (IaaS)



Funkcjonować jako usługa (Function as a Service - FaaS)

- Model przetwarzania w chmurze, który umożliwia klientom w chmurze opracowywanie aplikacji i wdrażanie funkcjonalności oraz naliczanie opłat tylko wtedy, gdy funkcja jest wykonywana.
- Jest często używany do wdrażania mikrousług i może być również określany jako przetwarzanie bezserwerowe (serverless computing)

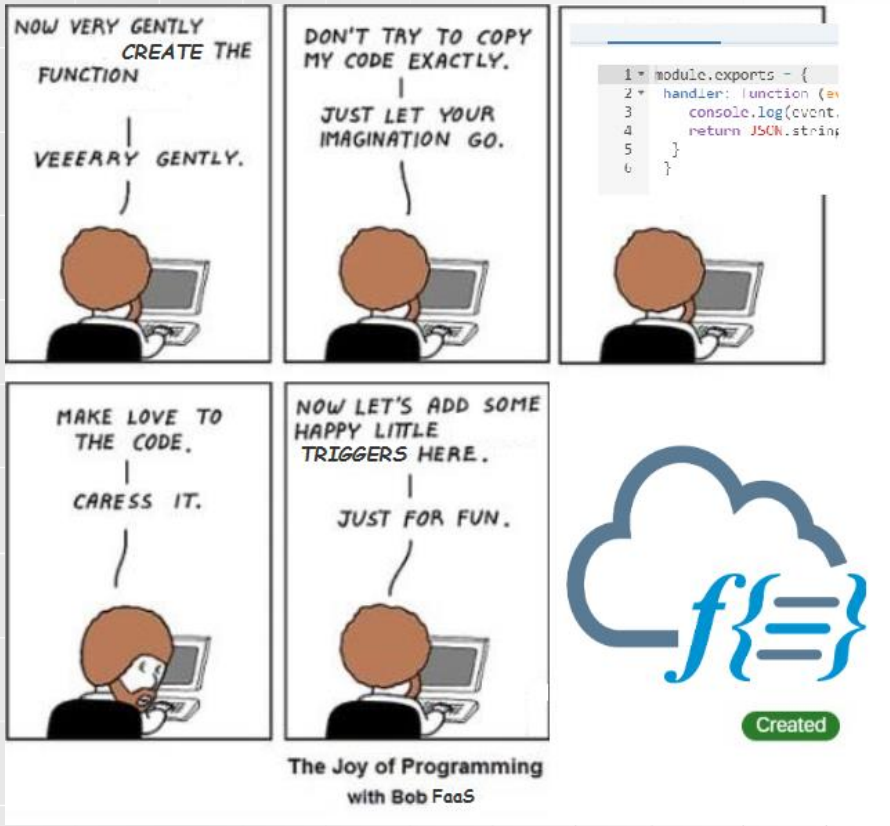


Funkcjonować jako usługa (Function as a Service - FaaS)

- Tradycyjne korzystanie z chmury wymaga od użytkowników udostępniania infrastruktury chmurowej — w tym serwerów wirtualnych, pamięci masowej i usług — obsługującej kod aplikacji. Ten kod działa w sposób ciągły i generuje regularne powtarzające się koszty biznesowe.
- Działa tylko wtedy, gdy wykonywana jest funkcja, a następnie się wyłącza.

Funkcjonować jako usługa (Function as a Service - FaaS)

- FaaS zapewnia programistom możliwość uruchomienia jednej funkcji, fragmentu logiki lub części lub całości aplikacji i ponoszenia opłat tylko wtedy, gdy kod jest wykonywany.
- Pierwszy model FaaS został wydany w 2014 roku przez hook.io, platformę typu open source, która obsługuje webhooki i mikrousługi. Następnie pojawiły się oferty Amazon Web Services (AWS), Google, IBM, Microsoft i Oracle.



PaaS vs FaaS - Konfiguracja

- Ze względu na prosty przypadek użycia i całkowicie zarządzane rozwiązania, FaaS oferuje wyjątkowo krótki czas obsługi administracyjnej.
- Mimo że PaaS oferuje więcej abstrakcji niż inne modele przetwarzania w chmurze, nadal wymaga pewnej konfiguracji. Może to być zaletą lub wadą w zależności od pożądanego poziomu kontroli.

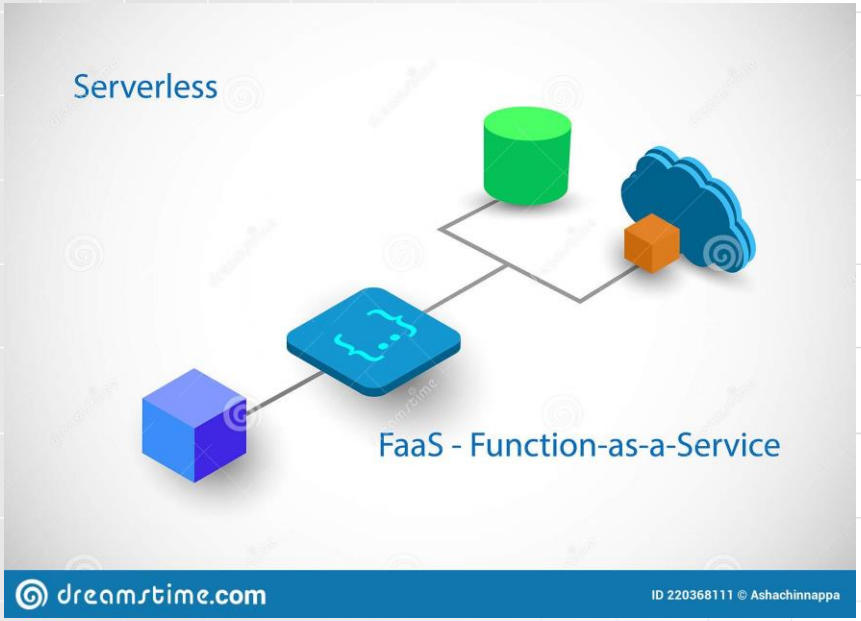
Cloud Computing Complex Magic

PaaS vs FaaS - Skalowalność

- PaaS skaluje się w górę iw dół w zależności od potrzeb, ale wymaga przemyślenia i konfiguracji od programisty.
- FaaS nie wymaga żadnego planowania pojemności. Skaluje się łatwo i łatwo w razie potrzeby.

PaaS vs FaaS - Cennik

- PaaS ma różne modele cenowe w zależności od dostawcy, ale często jest bardziej kosztowny niż FaaS, jeśli masz małe obciążenie zasobów.
- FaaS pozwala programistom płacić za każde wywołanie funkcji, oszczędzając czas, jeśli wyzwalaczy zdarzeń jest niewiele.
- PaaS może być lepszą opcją w przypadku stałych lub dużych obciążeń pod względem ceny.



Wrocław
University
of Science
and Technology

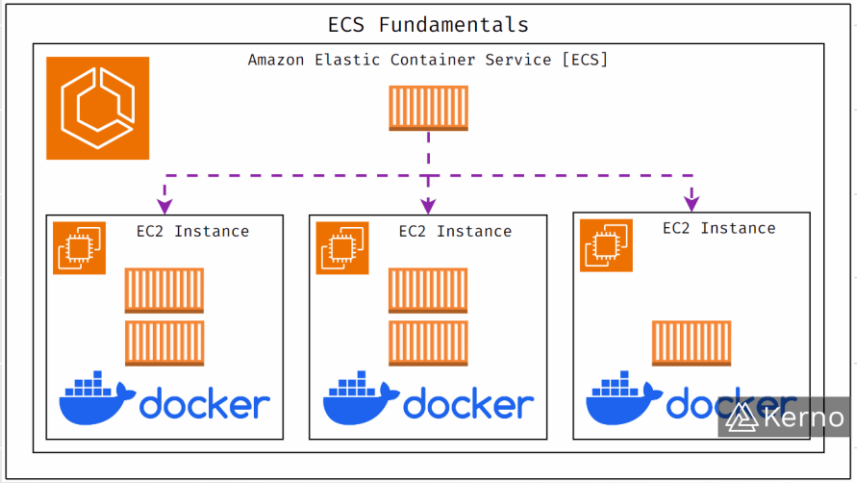
AWS Fargate

AWS Fargate

- AWS Fargate to zarządzana usługa obliczeniowa serverless umożliwiająca uruchamianie kontenerów bez potrzeby zarządzania infrastrukturą serwerową.
- Fargate automatycznie uruchamia i zarządza infrastrukturą niezbędną do działania kontenerów.
- Użytkownik definiuje tylko zadanie (task), określając zasoby CPU, RAM i konfigurację kontenerów.

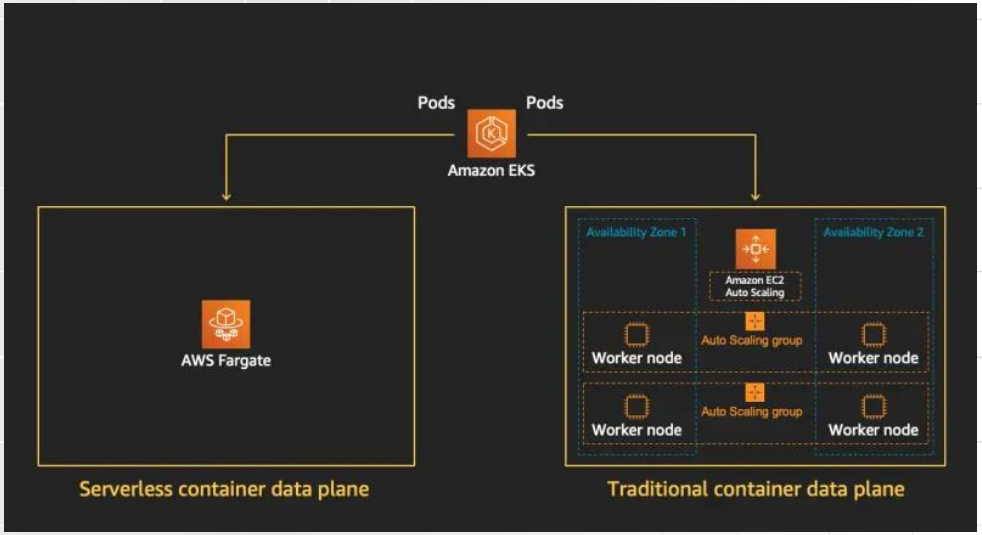
AWS Fargate - kluczowe cechy

- **Brak zarządzania klastrami -** Fargate automatycznie zarządza infrastrukturą obliczeniową.
- **Elastyczność i skalowalność -** Automatyczne skalowanie na podstawie wymagań aplikacji.
- **Płatność za wykorzystane zasoby -** Koszty naliczane za CPU i RAM używane przez zadania.



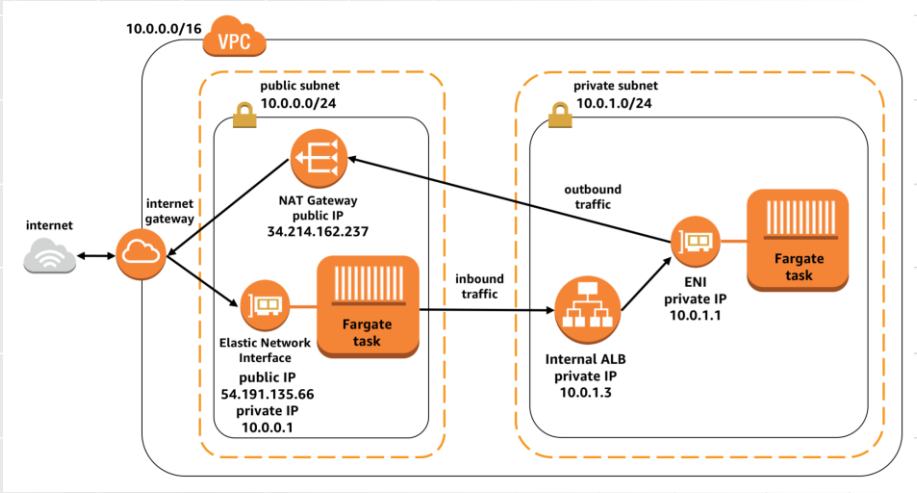
AWS Fargate - zarządzanie zasobami

- Dynamiczne przydzielanie zasobów: Fargate alokuje CPU i pamięć RAM dla każdego zadania lub kontenera indywidualnie.
- Izolacja zadań: Każdy kontener działa w swoim dedykowanym środowisku, co zwiększa bezpieczeństwo.



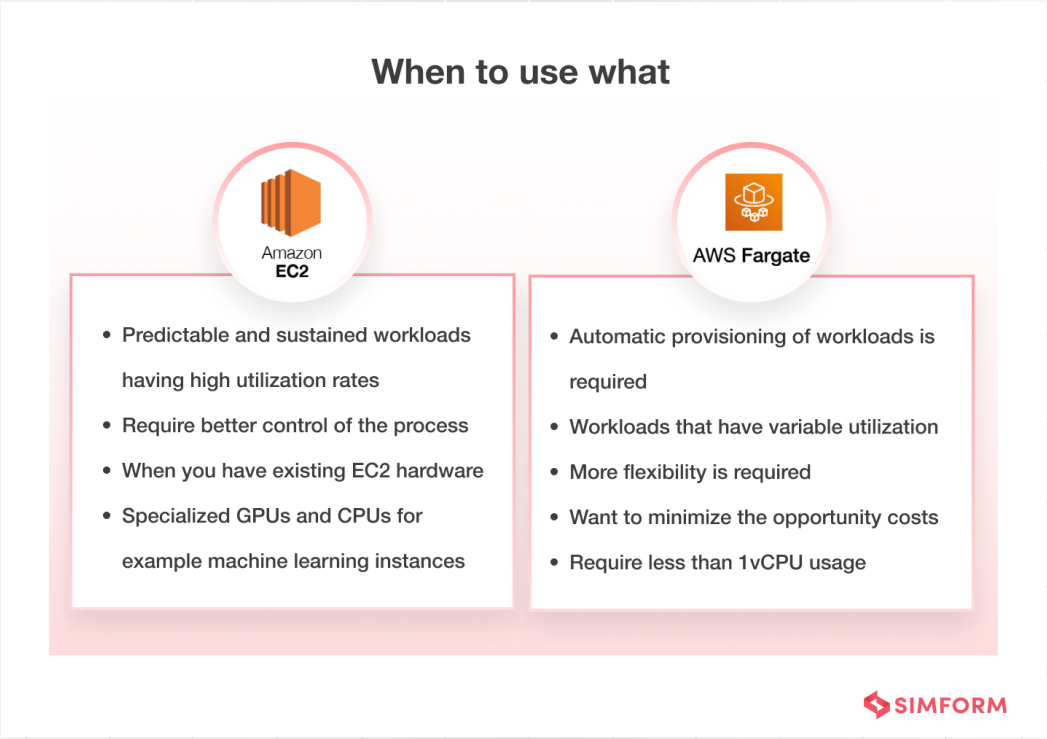
AWS Fargate - główne komponenty

- **Task Definitions** (Definicje zadań):
  - Specyfikacje kontenerów (obrazy, zmienne środowiskowe, porty).
  - Zasoby (CPU, RAM).
- **Task** (Zadanie): Instancja uruchomionego kontenera na podstawie definicji zadania.
- **Networking** (VPC, ENI):
  - Fargate działa w wirtualnej sieci VPC.
  - Każde zadanie otrzymuje dedykowany interfejs sieciowy (ENI).



AWS Fargate vs EC2

- Brak potrzeby zarządzania węzłami w klastrze (EC2 instances).
- Automatyczne skalowanie i alokacja zasobów.
- Brak możliwości dostępu do systemu operacyjnego hosta.

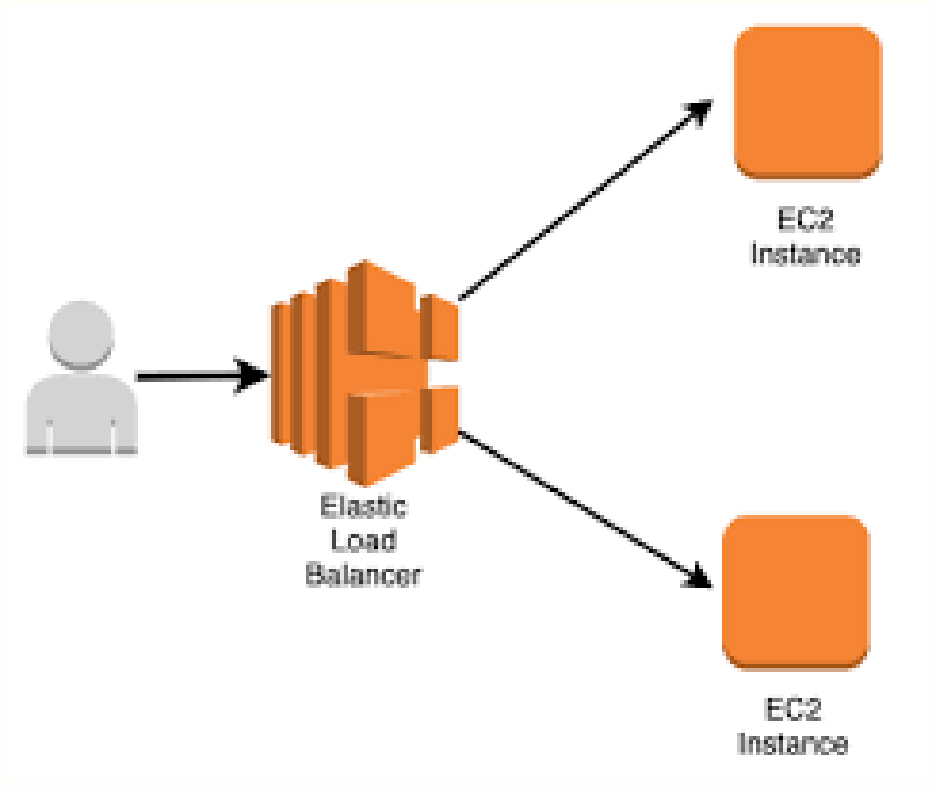


Wrocław
University
of Science
and Technology

Load Balancing

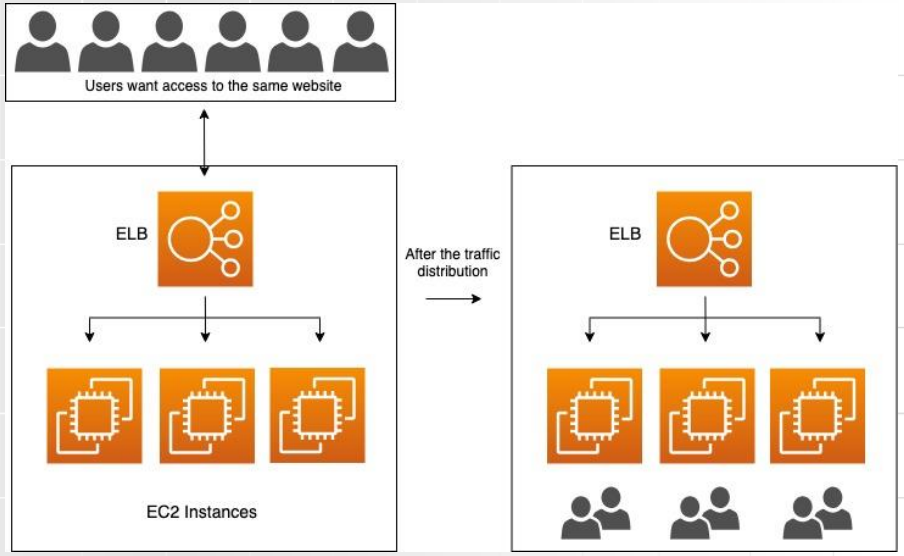
Elastic Load Balancing [1]

- Automatycznie rozdziela przychodzący ruch aplikacji na wiele instancji (np. EC2)



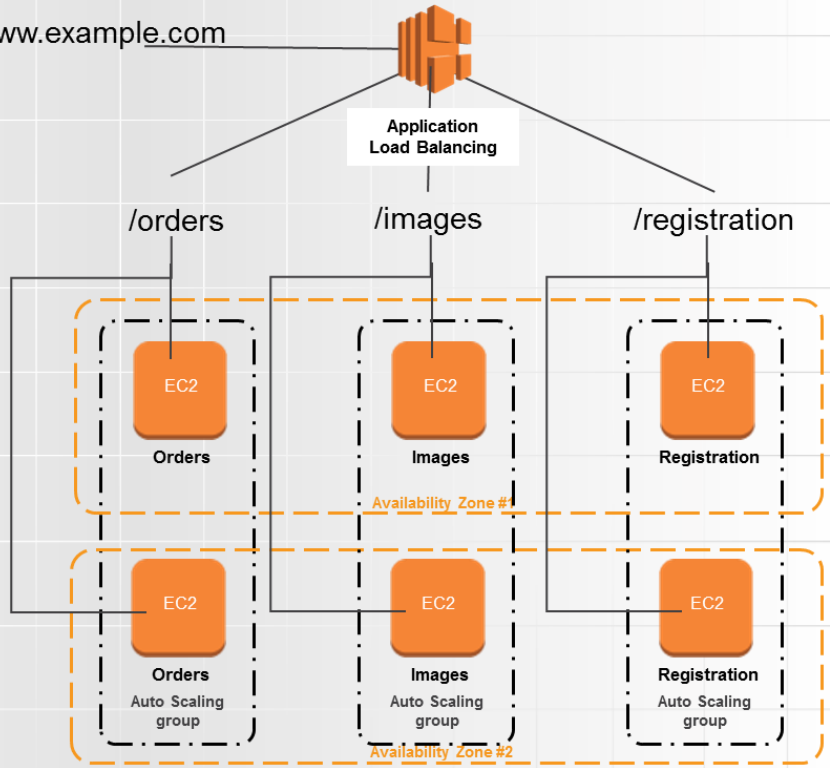
Elastic Load Balancing [2]

- Jeśli ruch na stronie internetowej nagle wzrośnie, ruch ten może zostać przekierowany do innych instancji EC2 (lub innych typów instancji, takich jak instancje Lambda), które zostały wcześniej ustanowione w tym celu.
- Duży ruch może spowodować zamknięcie aplikacji i stron internetowych, jeśli serwer nie
- Równoważenie obciążenia pozwala uniknąć przeciążenia pojedynczego serwera z powodu zwiększonego kierowanego do niego ruchu.



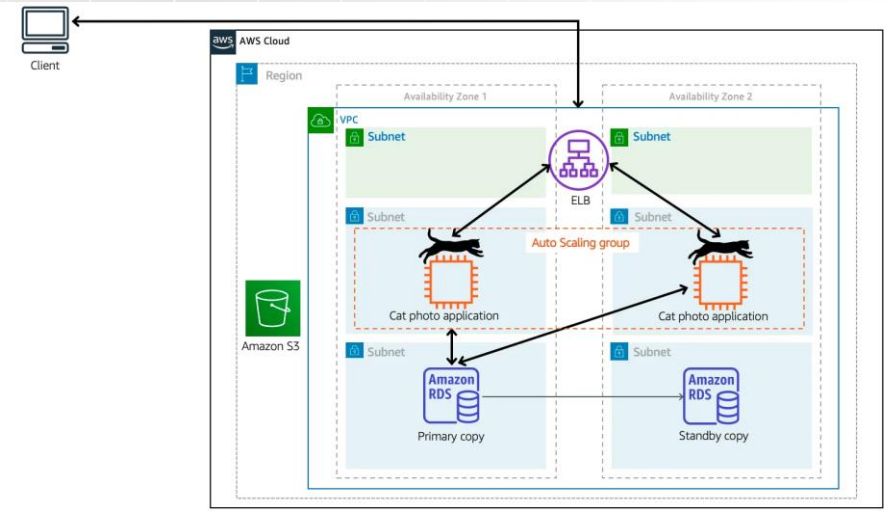
Application Load Balancer [1]

- Najlepiej nadaje się do równoważenia obciążenia ruchem Hypertext Transfer Protocol (HTTP) i Secure HTTP (HTTPS)
- Zapewnia zaawansowane kierowanie żądań ukierunkowane na dostarczanie nowoczesnych architektur aplikacji, w tym mikrousług i kontenerów
- Działa na poziomie indywidualnego żądania (warstwa 7)



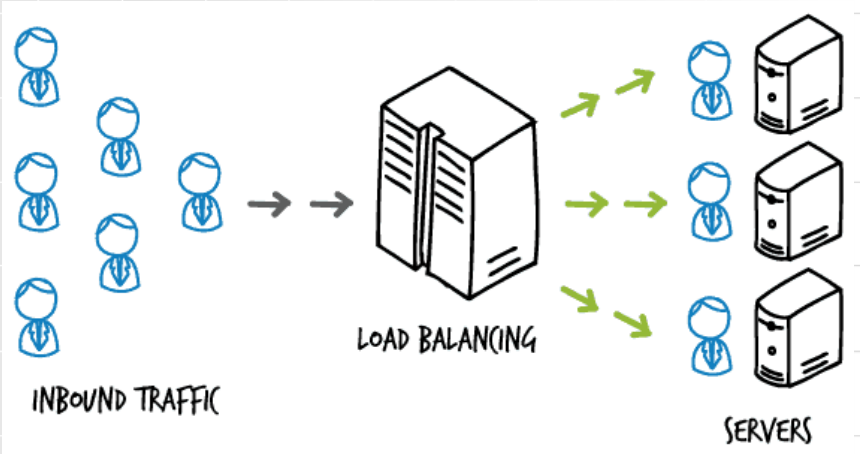
Application Load Balancer [2]

- Application Load Balancer kieruje ruch do celów w Amazon Virtual Private Cloud (Amazon VPC) na podstawie treści żądania.
- Równoważenie Load Balancer aplikacji odbywa się na podstawie zawartości jednolitego lokalizatora zasobów (URL). Na przykład, jeśli adres URL kończy się na /main, żądanie zostanie skierowane do jednej instancji; jeśli adres URL kończy się ciągiem blog/, zostanie przekierowany do innej instancji, o ile wcześniej została wykonana definicja modułu równoważenia obciążenia aplikacji, aby tak się stało.



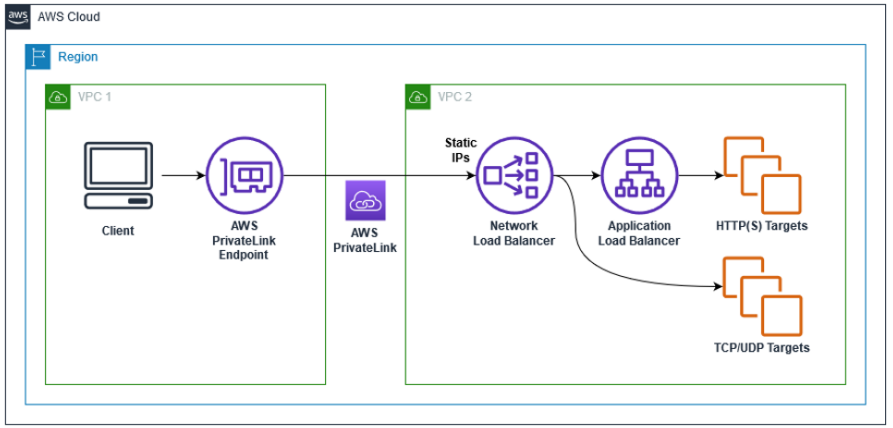
Network Load Balancer [1]

- Najlepiej nadaje się do równoważenia obciążenia ruchu protokołu kontroli transmisji (TCP), protokołu datagramów użytkownika (UDP) i protokołu TLS (Transmission Layer Security), gdzie wymagana jest ekstremalna wydajność.
- Działa na poziomie połączenia (warstwa 4),
- Kieruje ruch do celów w Amazon VPC i jest w stanie obsłużyć miliony żądań na sekundę przy zachowaniu bardzo niskich opóźnień.



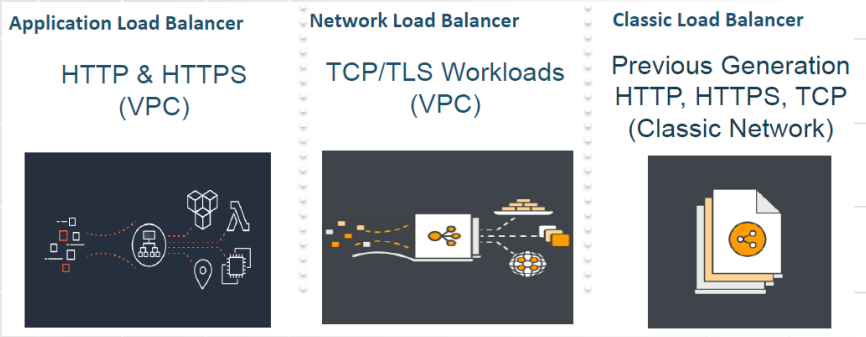
Network Load Balancer [2]

- Jest zoptymalizowany pod kątem obsługi nagłych i niestabilnych wzorców ruchu
- Ze względu na zwiększoną prędkość, którą można osiągnąć w warstwie połączenia, jest bardziej pożądany, gdy próbuje się uniknąć większego natężenia ruchu sieciowego
- Pomaga uniknąć opóźnień, gdy zainteresowanie witryną internetową stanie się viralowe



Classic Load Balancer [1]

- Zapewnia podstawowe równoważenie obciążenia w wielu instancjach EC2 i działa na poziomie żądania i połączenia.
- Działa na poziomie połączenia (warstwa 4) oraz na poziomie indywidualnego żądania (warstwa 7),
- AWS odradza korzystania z Classic Load Balancer na korzyść ALB I NLB
- Istnieje bardzo niewiele scenariuszy, w których preferowane byłoby użycie ELB



ALB vs NLB vs CLB

Comparison Table

	ALB	NLB	ELB
Basic load balancing features			
Balance load between targets	Yes	Yes	Yes
Perform health checks on targets	Yes	Yes	Yes
Highly available	Yes	Yes	Yes
Elastic	Yes	Yes	Yes
TLS Termination	Yes	Yes	Yes
Performance	Good	Very high	Good
Send logs and metrics to CloudWatch	Yes	Yes	Yes
Layer 4 (TCP)	No	Yes	Yes
Layer 7 (HTTP)	Yes	No	Yes
Running costs	Low	Low	Low

https://iamondemand.com/blog/elb-vs-alb-vs-nlb-choosing-the-best-aws-load-balancer-for-your-needs/

ALB vs NLB vs CLB

Advanced load balancing features			
Advanced routing options	Yes	N/A	No
Can send fixed response without backend	Yes	No	No
Supports user authentication	Yes	No	No
Can serve multiple domains over HTTPS	Yes	Yes	No
Preserve source IP	No	Yes	No
Can be used in EC2-Classic	No	No	Yes
Supports application-defined sticky session cookies	No	N/A	Yes
Supports Docker containers	Yes	Yes	Yes (*)
Supports targets outside AWS	Yes	Yes	No
Supports websockets	Yes	N/A	No
Can route to many ports on a given target	Yes	Yes	No

https://iamondemand.com/blog/elb-vs-alb-vs-nlb-choosing-the-best-aws-load-balancer-for-your-needs/

Wrocław
University
of Science
and Technology

Dziękuję za uwagę