wydział elektrotechniki elektroniki informatyki i automatyki

AMG.net
A Bull Group Company

Projektowanie architektury w chmurze

Dobre praktyki

Małgorzata Urbaniak
IT Architect, AMG.net

wydział elektrotechniki elektroniki informatyki i automatyki

AMG.net
A Bull Group Company

Co czyni chmurę atrakcyjną?

Politechnika Łódzka

AMG.net
A Bull Group Company

Korzyści z chmury

Abstract resources	Skoncentrowanie się na potrzebach funkcjonalnych zamiast na specyfikacji sprzętowej. Potrzeby dotyczące infrastruktury zmieniają się razem z potrzebami funkcjonalnymi.
On-Demand Provisioning	Rozszerzanie infrastruktury wtedy gdy to jest potrzebne (na żądanie). Zwalnianie zasobów gdy nie są używane.
Scalability in minutes	Skalowanie architektury w zależności od potrzeb użytkownika w momencie zapotrzebowania.
Pay per consumption	Płatność za realne użycie. Brak długoterminowych zobowiązań.
Efficiency of Experts	Wykorzystanie umiejętności, wiedzy i zasobów ekspertów.

Skalowalność

- Skalowalność (*ang. scalability*) to zapewnienie coraz wydajniejszej pracy w miarę zwiększania liczby elementów składowych. (Źródło: http://pl.wikipedia.org/wiki/Skalowalność)
- Modele skalowania:
  - Wertykalny (bigger / smaller boxes)
  - Horyzontalny (more / less boxes)
- Chmura w założeniach została zaprojektowana by dostarczać nieskończoną wydajność.
- Niezwykle ważne jest, aby zbudować skalowalną architekturę w celu skorzystania z skalowalnej infrastruktury.

wydział elektrotechniki elektroniki informatyki i automatyki

AMG.net
A Bull Group Company

Dobre praktyki budowania aplikacji w chmurze

Politechnika Łódzka

AMG.net
A Bull Group Company

Siedem najlepszych praktyk projektowania aplikacji w chmurze:

- Design for failure and nothing fails
- Loose coupling sets you free
- Implement “Elasticity”
- Don't fear constraints
- Think Parallel
- Leverage different storage options
- Build Security in every layer

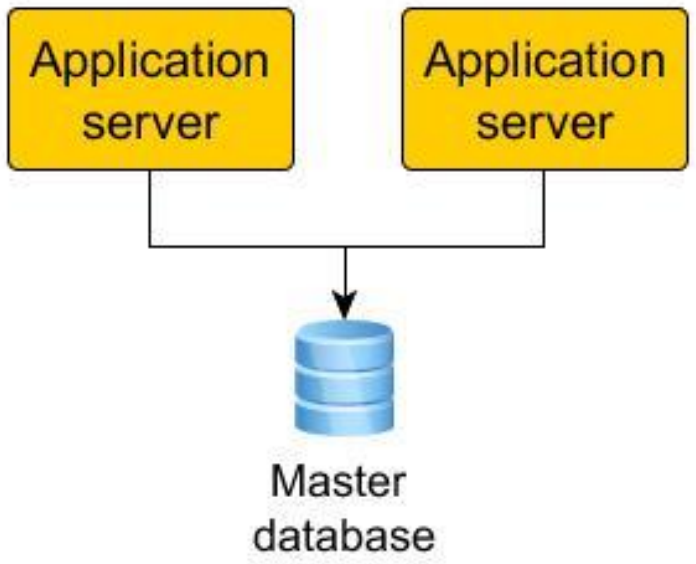
AMG.net
A Bull Group Company

> *Everything fails, all the time*
>
> Werner Vogels, CTO Amazon.com

Cel: Aplikacja powinna kontynuować swoje działanie w przypadku uszkodzenia, usunięcia lub wymiany fragmentów sprzętu.

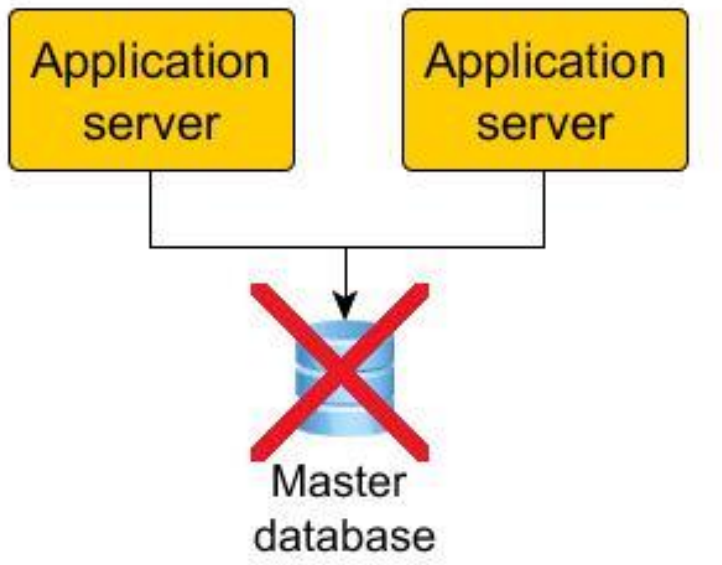
Projektuj z myślą o błędach

- Unikaj pojedynczych punktów awarii (*ang. singles point of failure*).



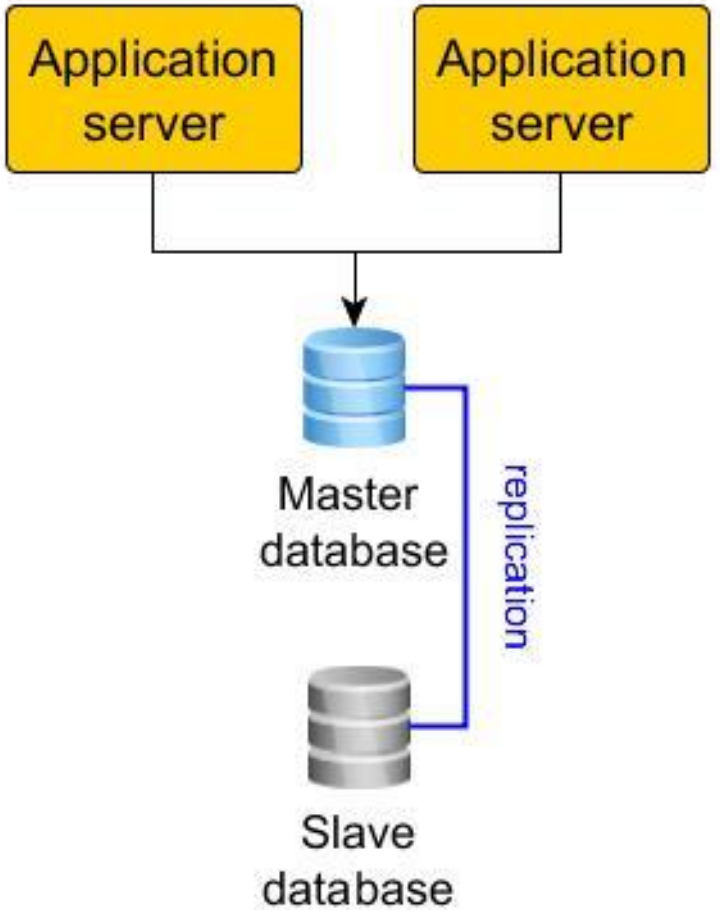
Projektuj z myślą o błędach

- Unikaj pojedynczych punktów awarii (*ang. singles point of failure*).



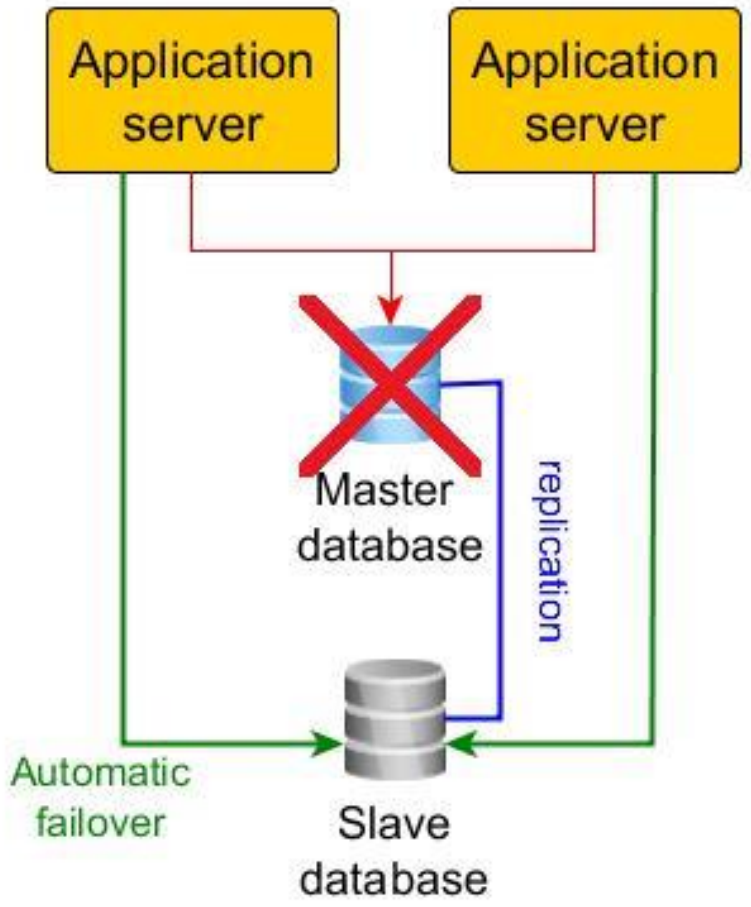
Projektuj z myślą o błędach

- Unikaj pojedynczych punktów awarii (*ang. singles point of failure*).
- Projektuj z założeniem, że każdy element może się zespuć. Przewiduj awarie i przygotuj się na nie z wyprzedzeniem.



Projektuj z myślą o błędach

- Unikaj pojedynczych punktów awarii (*ang. singles point of failure*)
- Projektuj z założeniem, że każdy element może się zespuć. Przewiduj awarie i przygotuj się na nie z wyprzedzeniem



Wskazówki od Amazon Web Services:

- Korzystaj z adresów Elactic IP do określania spójnych i rekonfigurowalnych tras.
- Korzystaj z wielu EC2 Availability Zones (co najmniej dwóch).
- Konfiguruj bazy danych w trybie MultiAZ (wiele instancji Slave w różnych Availability Zones).
- Korzystaj z monitoringu Amazon CloudWatch (real-time monitoring).
- Korzystaj z Amazon Elastic Block Store (EBS) jako systemu plików.

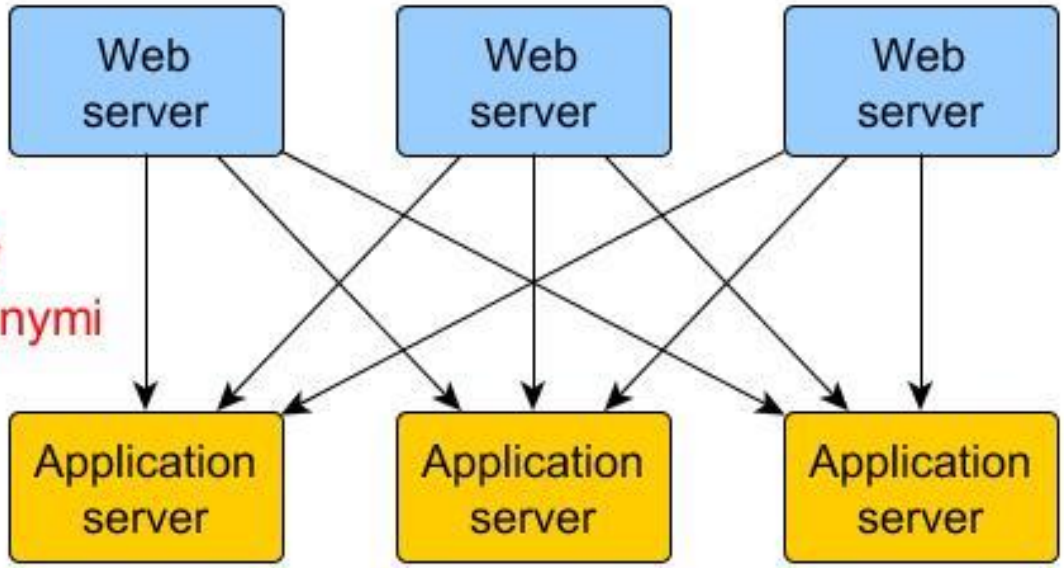
Luźne powiązania

- Luźne powiązania (*ang. loose coupling*) to termin określający sposób w jaki elementy architektury współpracują ze sobą jako samodzielne komponenty.
- Współpraca komponentów polega na wymianie komunikatów w określonej, standardowej formie.

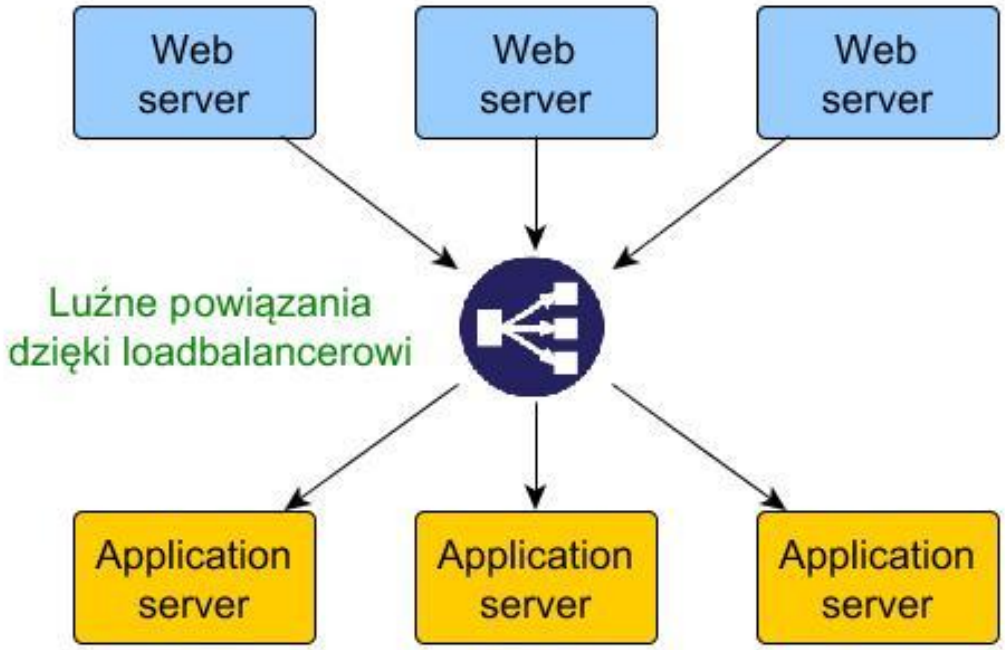
- Luźne powiązania są jedną z podstawowych cech architektury SOA, intensywnie wykorzystywaną w architekturach chmurowych.
- Im luźniejsze powiązania pomiędzy elementami architektury tym większe i lepsze skalowanie aplikacji.

Ścisłe powiązania

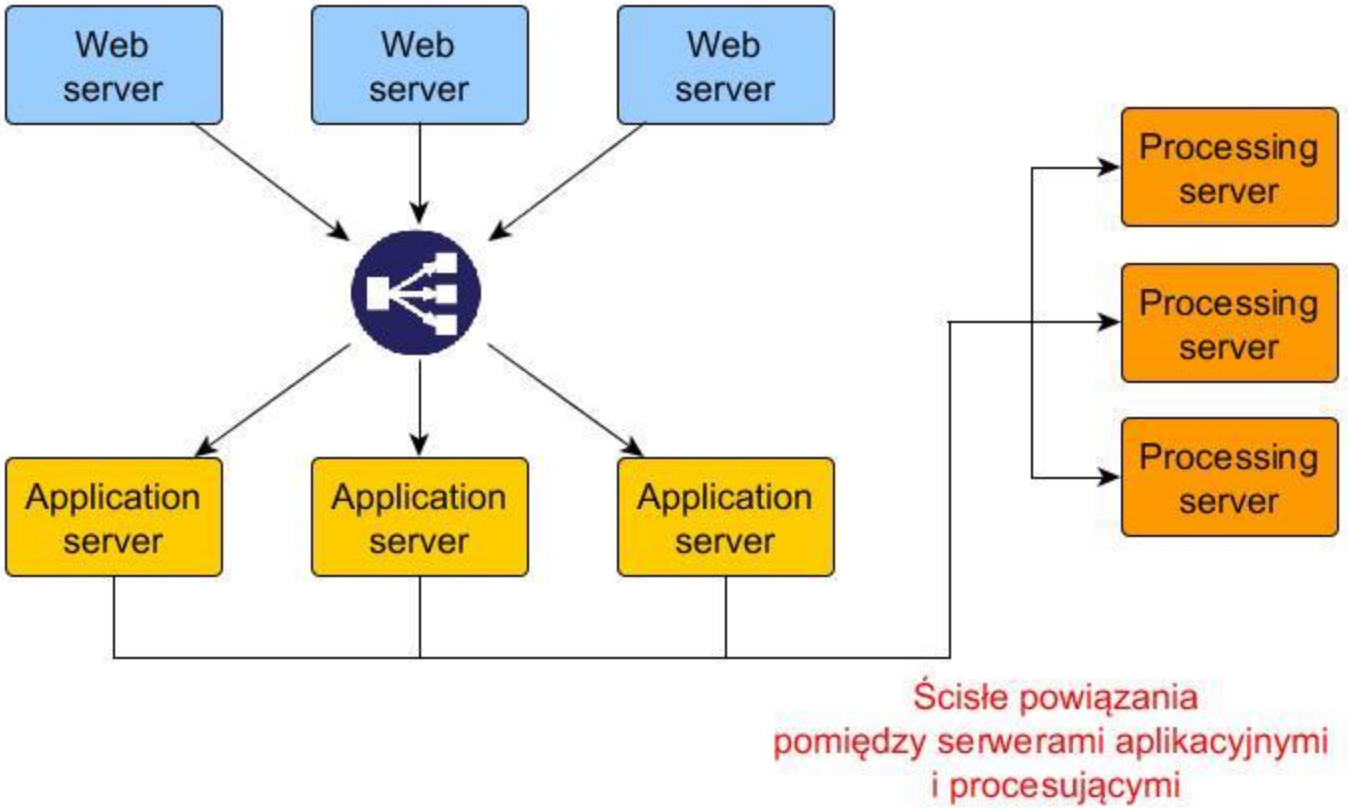
Ścisłe powiązania pomiędzy serwerami webowymi i aplikacyjnymi



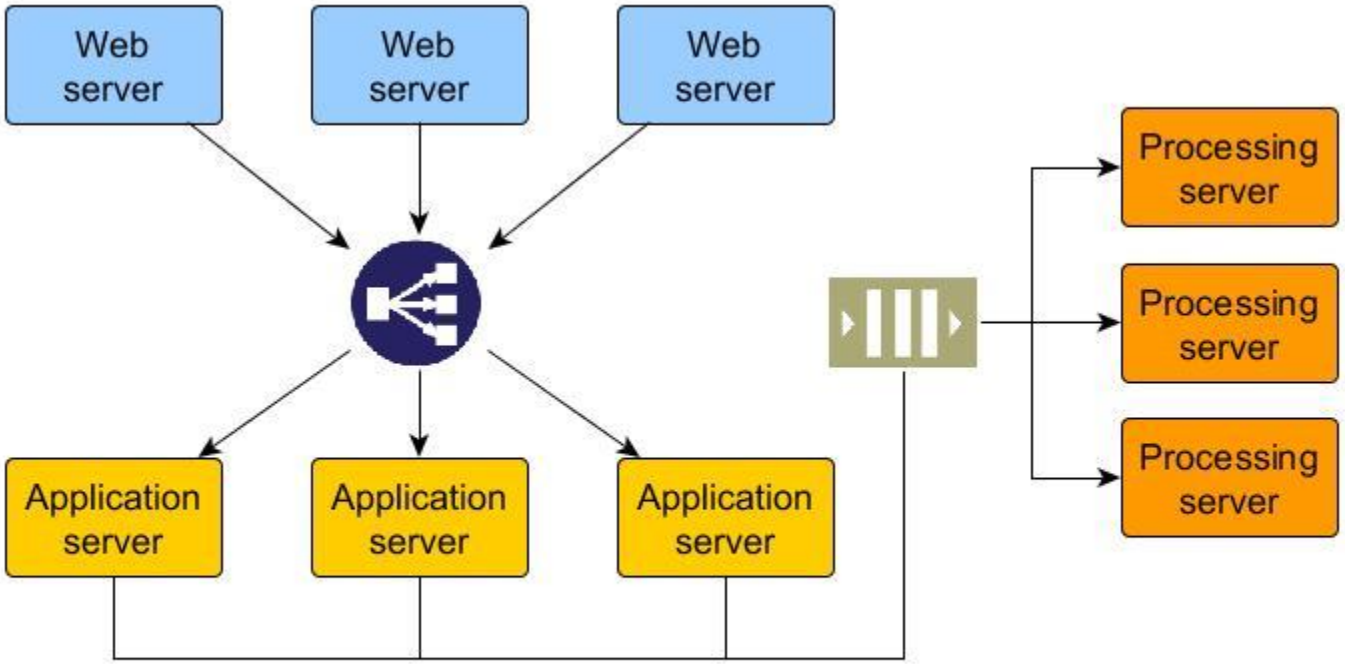
Równoważenie obciążenia (*ang. load balancing*)



Ścisłe powiązania



Kolejki komunikatów (*ang. queues*)



Luźne powiązanie
z wykorzystaniem kolejki

Luźne powiązania

- Projektuj architektury składające się z samodzielnych komponentów.
- Projektuj każdy komponent jako „czarne pudełko” – z dobrze zdefiniowanym interfejsem.
- Używaj load balancerów do rozdzielania ruchu pomiędzy instancje.
- Wykorzystuj kolejki do przekazywania komunikatów pomiędzy komponentami.

Luźne powiązania

Wskazówki od Amazon Web Services:

- Korzystaj z Amazon SQS (Simple Queue Services) do izolowania komponentów.
- Korzystaj z Amazon SQS (Simple Queue Services) jako bufer pomiędzy komponentami.
- Twórz aplikacje tak bezstanowe jak to możliwe. Zapisuj stan sesji poza komponentem, np. w Amazon SimpleDB, Amazon ElastiCache.

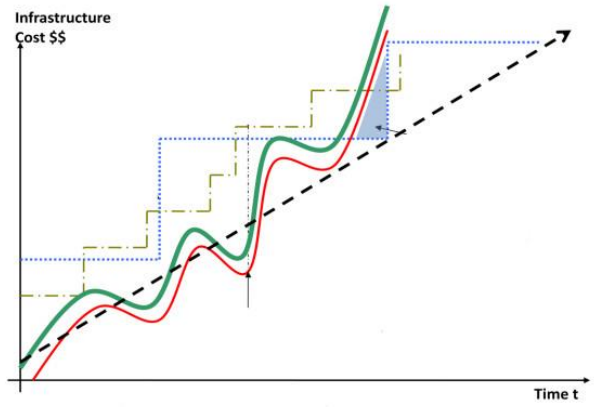
Luźne powiązania

Projektuj każdy komponent tak by:

- dostarczał spójny interfejs bez udostępniania szczegółów implementacyjnych,
- samodzielnie realizował swoją skalowalność we wszystkich odpowiednich wymiarach,
- współdziałał z innymi komponentami asynchronicznie z wykorzystaniem kolejek komunikatów (jeśli to możliwe).

Elastyczność

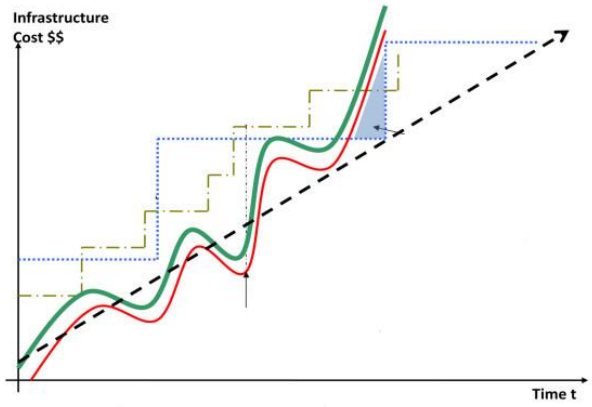
- Elastyczność to podstawowa cecha chmury.
- Projektując aplikację nie zakładaj ciągłej dostępności i stałej lokalizacji współpracujących komponentów.



- Używaj wzorców odpornych na ponowne uruchamianie lub czasowe niedostępności systemów zewnętrznych (np. odnawialna pula połączeń).
- Promuj dynamiczną konfigurację.

Elastyczność

- Zautomatyzuj proces wdrażania (infrastruktury, aplikacji):
  - Stwórz zbiór instrukcji – krótkich, często używanych skryptów instalacyjnych i konfiguracyjnych.
  - Zarządzaj konfiguracją i procesem wdrażania poprzez agentów zagnieżdżonyc w obrazie serwera (AMI).



- Konfiguruj instancje przy uruchamianiu:
  - Każda instancja powinna wiedzieć (przy starcie) jaką pełni rolę, (np. Application Server, DB Server, Web Server, środowisko testowe / produkcyjne).
  - Przy uruchamianiu instancja pobierze właściwe zasoby (kod, konfigurację, skrypty) w zależności od roli jaką pełni.

Wskazówki od Amazon Web Services:

- Definiuj grupy autoscalingowe dla klastra instancji z wykorzystaniem opcji EC2 Autoscaling.
- Monitoruj metryki systemu (CPU, memory, disk I/O, network I/O) z wykorzystaniem CloudWatch. Definiuj alerty z przypisaną akcją (np. zwiększenie liczby instancji w grupie autoscalingowej, wysłanie notyfikacji).
- Przechowuj i pobieraj informacje o konfiguracji instancji automatycznie, np. z Amazon SimpleDB.
- Umieszczaj artefakty aplikacji na Amazon S3. Nowa instancja podczas rozruchu pobierze najświeższą wersję.

Nie bój się ograniczeń

- Nie bój się ograniczeń. Szukaj dla nich rozwiązań!
- Korzystanie z elastycznych, dostępnych na żądanie zasobów pozwala na stworzenie różnych modeli architektury. Na pewno uda się znaleźć taki, który rozwiąże ograniczenia.

Nie bój się ograniczeń

Instancje w chmurze mają za mało RAM

- Rozłóż obciążenie pomiędzy instancje
- Użyj rozproszonych współdzielonych cache (memcache)

Potrzebuję więcej IOPS na bazie danych

- Dodaj instancje read only i korzystaj z nich dla operacji odczytu
- Stwórz klaster bazodanowy

Mój serwer fizyczny ma lepszą specyfikację sprzętową

Uruchom więcej instancji, ale tylko wtedy gdy tego potrzebujesz

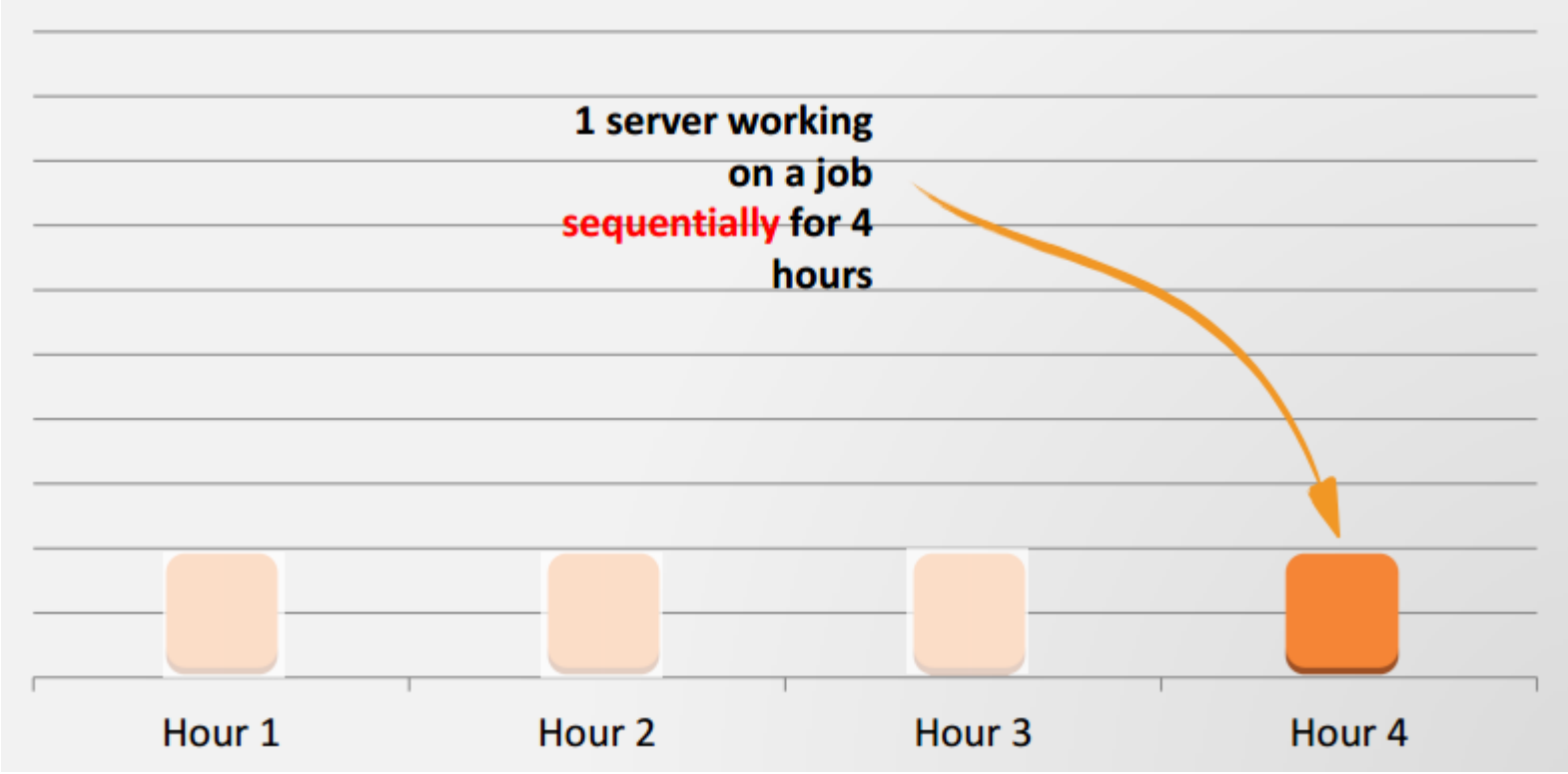
Potrzebuję statycznych IP dla instancji serwerów

Użyj skryptów startowych pobierających adresy IP z konfiguracji (np. w bazie danych)

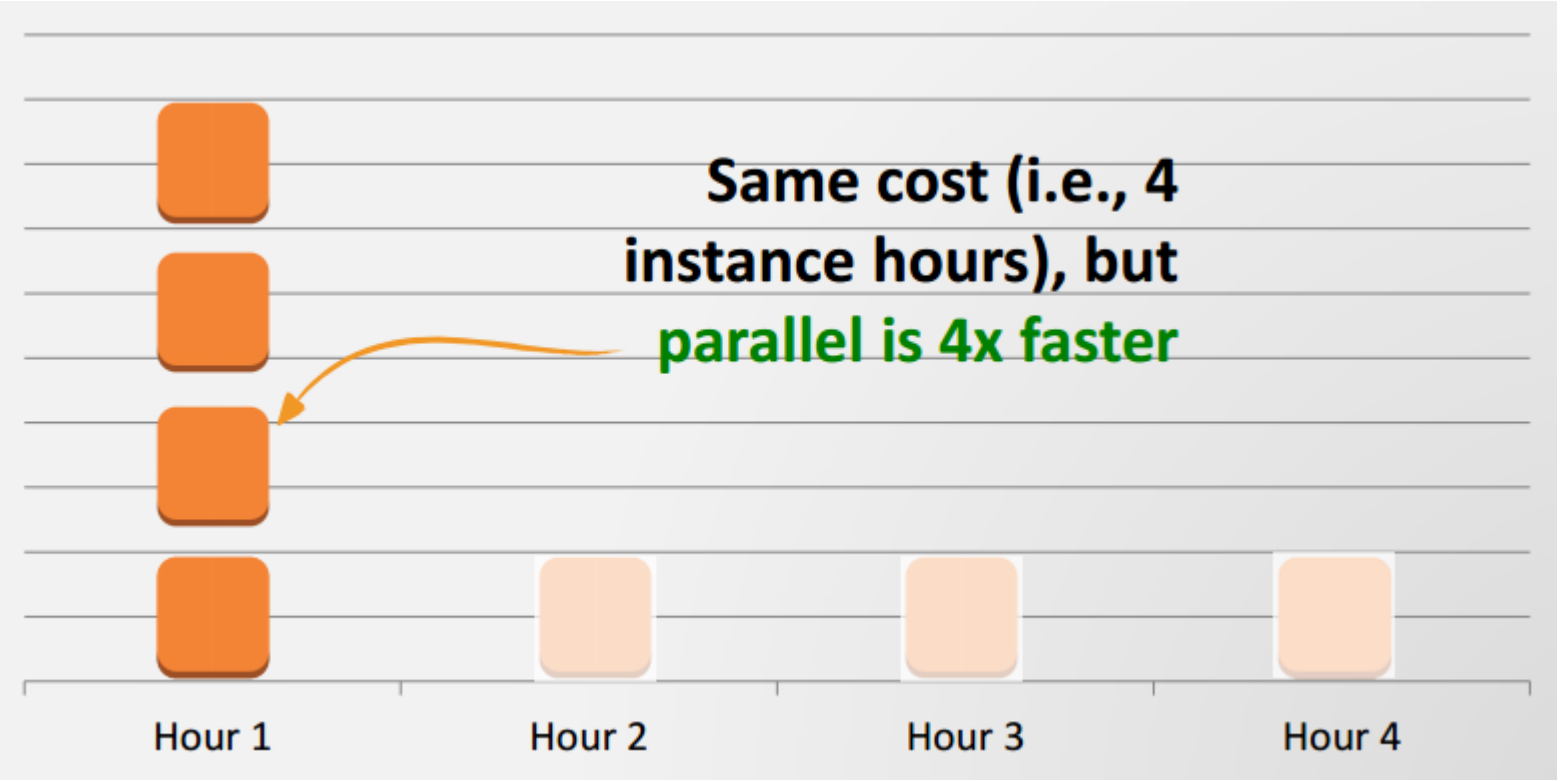
AMG.net
A Bull Group Company

> *The beauty of the cloud shines when you combine elasticity and parallelization*

Eksperymentuj z mechanizmami równoległymi, wykonującymi jednocześnie (równolegle) obliczenia na wielu instancjach serwerów.



Eksperymentuj z mechanizmami równoległymi, wykonującymi jednocześnie (równolegle) obliczenia na wielu instancjach serwerów.



Przetwarzanie równoległe

- Preferuj rozwiązania wielowątkowe zamiast sekwencyjnych
- Uruchamiaj równoległe zadania wykorzystując mechanizmy MapReduce
- Używaj load balancerów do równoważenia obciążenia pomiędzy instancjami serwerów
- Twórz proste, samodzielne zadania. Kieruj się zasadą „shared nothing”

AMG.net
A Bull Group Company

> *One size DOES NOT fit all*
>
> *Keep dynamic data closer to the compute and static data closer to the end-user*

- Rozpatruj różne opcje przechowywania danych i umieszczaj w nich odpowiednie typy danych.
- Dane dynamiczne przechowuj blisko chmury obliczeniowej.
- Treści statyczne przechowuj blisko użytkownika.

Opcje przechowywania danych w AWS:

Amazon S3	Duże statyczne obiekty (np. kopie zapasowe, logi, wersje wykonywalne aplikacji)
Amazon Cloudfront	Treść statyczna rozlokowana blisko użytkownika (np. obrazki, video, audio, PDFs, JS, CSS)
Amazon SimpleDB	Proste dane z możliwością przeszukiwania i indeksowania (np. metadane, mapowania, konfiguracja)
Amazon EC2 local disc drive	Tymczasowe dane (operacyjne)
Amazon EBS	Zasób dyskowy, trwałe przechowywanie (np. boot data,)
Amazon RDS	Usługa relacyjnej bazy danych (MySQL RDS)

AMG.net
A Bull Group Company

> *Security should be implemented in every layer of the cloud application architecture*

- Chroń dane podczas transportu
- Chroń przechowywane dane
- Zarządzaj rolami i poziomami dostępu użytkowników
- Dbaj o bezpieczeństwo kluczy dostępowych
- Zabezpieczaj swoją aplikację na każdym poziomie

Bezpieczeństwo

- Chroń dane podczas transportu
  - Używaj certyfikatów SSL podczas przesyłania danych wrażliwych/poufnych pomiędzy komponentami architektury, np. przeglądarką i serwerem webowym
  - Zasoby, do których dostęp powinien być ograniczony, umieszczaj w wirtualnej chmurze prywatnej. Będą do nich miały dostęp jedynie wybrane komponenty korzystające z protokołu IPSec.
- Chroń dane przechowywane w chmurze
  - Szyfruj dane poufne/wrażliwe jeśli chcesz przechowywać je w chmurze (np. w S3). Szyfruj dane przed wysłaniem (np. GnuPG, OpenPGP)
  - Rozważ opcje szyfrowanych systemów plików dla przechowywania danych wrażliwych
  - Pamiętaj by zabezpieczyć swoje dane przez awarią. Wykonuj regularne kopie zapasowe danych

Zarządzaj rolami i poziomami dostępu użytkowników do usług chmury:

- Korzystaj z narzędzi klasy IAM (Identity and Access Management).
- Definiuj konto per użytkownik z przypisanymi unikalnymi danymi uwierzytelniającymi (*ang. credentials*).
- Przypisuj uprawnienia do konta. Nadawaj dostęp tylko do tych zasobów, których użytkownik potrzebuje.
- W szczególności nowy użytkownik nie ma dostępu do któregokolwiek z zasobu chmury.

Dane uwierzytelniające w AWS:

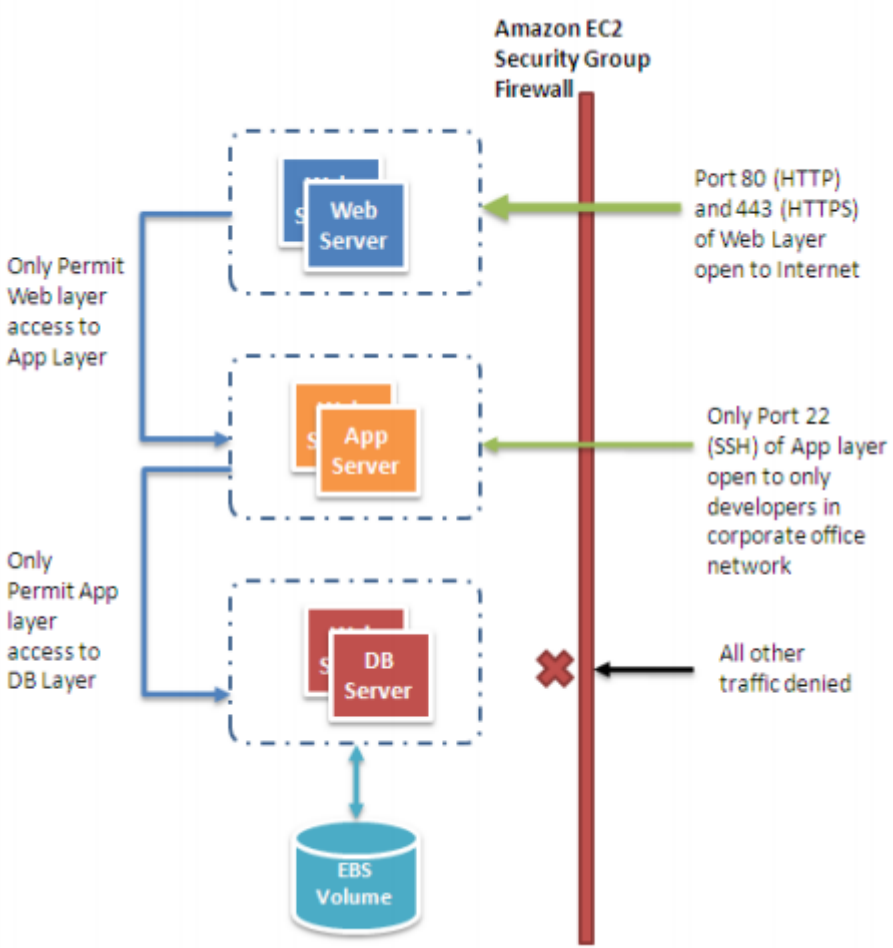
- Dwa typy danych uwierzytelniających:
  - klucz dostępowy
  - certyfikat X.509
- Klucz dostępowy (access key) składa się z dwóch części:
  - access key ID
  - secret access key

- Klucz dostępowy potrzebny jest do wyznaczenia podpisu żądania przy korzystaniu z REST lub Query API.
- Nie umieszczaj kluczy dostępowych w obrazie instancji (AMI)! Ich zmiana będzie wymagała nowego obrazu AMI. Przekaż je jako parametr wywołania.

Bezpieczeństwo

Używaj grup zabezpieczeń (*ang. security groups*), czyli nazwanych zbiorów zasad określających jaki ruch może być wpuszczony do instancji poprzez specyfikacje portów oraz adresów IP.

Grupy zabezpieczeń stanowią podstawową ochronę firewall działających instancji.



Bezpieczeństwo

- Regularnie pobieraj poprawki ze stron dostawcy oprogramowania i aktualizuj oprogramowanie obrazów instancji
- Testuj instancje po aktualizacji oprogramowania
- Przygotuj automatyczne skrypty umożliwiające okresowe wykonywanie testów bezpieczeństwa
- Upewnij się, że oprogramowanie dostarczane przez firmy trzecie jest bezpieczne i skonfigurowane zgodnie z rekomendacjami producenta
- Nie uruchamiaj procesów jako z poziomu użytkonika root/Administrator.

Literatura i materiały źródłowe

- Jinesh Varia:
  Architecting for the Cloud: Best Practices
- Rajkumar Buyya; James Broberg; Andrzej Goscinski:
  Cloud Computing: Principles and Paradigms
- Prezentacja Amazon Web Services
  Architecting in the Cloud
- Dob Todorov, Yinal Ozkan:
  AWS Security Best Practices
- Joseph Baron, Sanjay Kotecha:
  Storage Options in the AWS Cloud

wydział elektrotechniki elektroniki informatyki i automatyki

AMG.net
A Bull Group Company

Dziękujemy za uwagę