

Projekt z rozproszonych i obiektowych systemów baz
danych

Projekt aplikacji wykorzystującej repliki MongoDB

Autorzy:

Maciej Kiedrowski, nr indeksu: 200105

Joanna Piątek, nr indeksu: 199966

Grupa: Poniedziałek 16:10

Data oddania: 23.01.2017

Prowadzący zajęcia: Dr inż. Robert Wójcik, W4/K-9

Ocena pracy:

Spis treści

1	Wstęp	3
1.1	Cele projektu	3
1.2	Założenia projektowe	3
2	Replikacja	4
2.1	Replikacja w MongoDB	4
2.1.1	Zestaw replik	4
2.1.2	Replikacja	5
2.1.3	Wysoka dostępność	5
2.1.4	Dostępność	6
3	Implementacja bazy danych w środowisku MongoDB	7
3.1	Infrastruktura	7
3.1.1	Instancje	7
3.1.2	Sieć wewnętrzna	7
3.1.3	Sieć publiczna	7
3.2	Konfiguracja MongoDB	7
3.3	Schemat bazy danych	9
3.4	Weryfikacja	9

1 Wstęp

1.1 Cele projektu

1.2 Założenia projektowe

2 Replikacja

Proces replikacji polega na synchronizacji danych pomiędzy serwerami. Pozwala osiągnąć następujące korzyści:

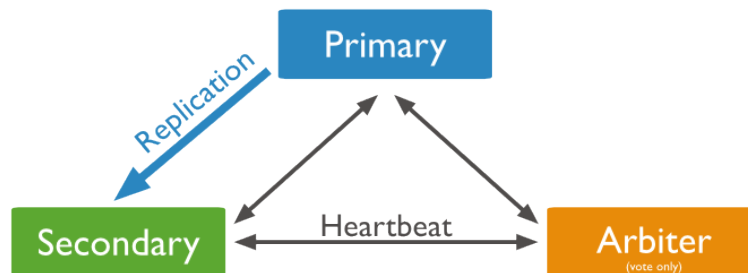
- Bezpieczeństwo danych poprzez redundancję
- Wysoką dostępność
- Skalowanie wydajności

2.1 Replikacja w MongoDB

Replikacja wbudowana w platformę *MongoDB* opiera się na *replica set* - zestawie replik.

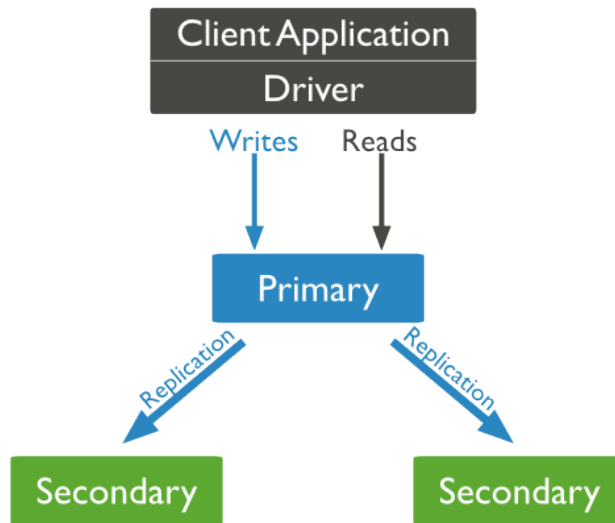
2.1.1 Zestaw replik

Zestaw replik grupa procesów *mongod* utrzymujących ten sam zestaw danych. Zestaw replik w ramach MongoDB składa się z węzłów zawierających dane oraz ewentualnego węzła wspomagającego arbitraż.



Rysunek 1: Zestaw replik

W każdym zestawie replik jeden z węzłów pełni rolę *Primary*. Węzeł ten jest jedynym, który akceptuje operacje zapisu, jest również domyślnym węzłem dla wszystkich operacji odczytu z zestawu replik. Pozostałe węzły wchodzące w skład zestawu, a niebędące węzłem arbitrażowym działają w trybie *Secondary*. Minimalna ilość węzłów zapewniająca poprawną pracę zestawu to 3, ilość węzłów powinna być nieparzysta.



Rysunek 2: Schemat działania zestawu replik

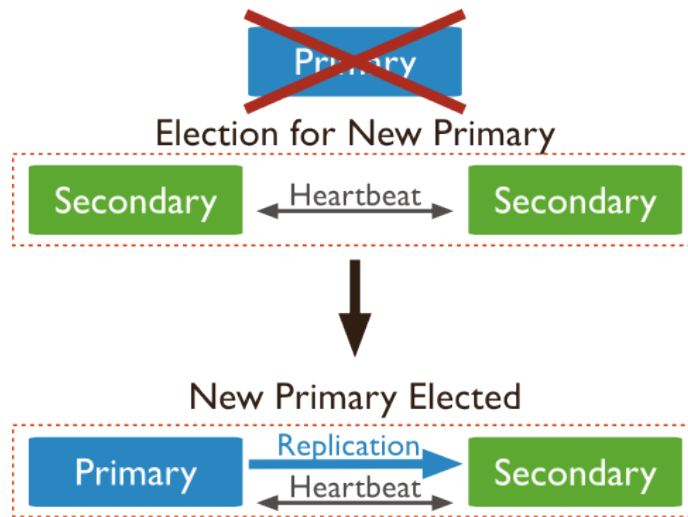
2.1.2 Replikacja

Replikacja w *MongoDB* jest replikacją asynchroniczną. Węzły *Secondary* replikują dziennik operacji (*oplog*) węzła *Primary* i wykonują operacje w nim zawarte na przechowywanym zbiorze danych.

2.1.3 Wysoka dostępność

Zestawie replik gwarantuje wysoką dostępność danych. W przypadku awarii węzła, w szczególności węzła *Primary* uruchamiany jest proces *failover*. Ma on na celu zapewnienie płynności dostępu do danych. Pozostałe działające węzły w ramach repliki przeprowadzają głosowanie nad wyborem nowego *Primary*, po zakończeniu głosowania zestaw odzyskuje pełną sprawność. Proces *failover* trwa przeważnie poniżej minuty, z czego 10-30 sekund to wykrycie awarii węzła *Primary*, następne 10-30 sekund proces głosowania.

W trakcie głosowania zestaw pracuje w trybie *read-only* - żaden z węzłów nie posiada statusu *Primary* a zatem wszystkie żądania zapisu do bazy są odrzucane.



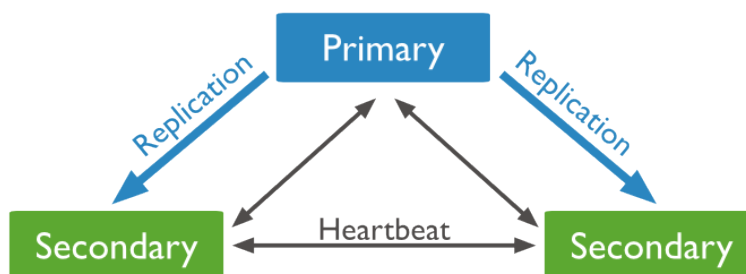
Rysunek 3: Wysoka dostępność z wykorzystaniem procesu *failover*

Warunkiem koniecznym do istnienia węzła *Primary* w zestawie, jest dostępność ponad 50% węzłów. Oznacza to, że dla zestawu składającego się z 3 instancji, jest on w pełni funkcjonalny przy awarii 1 węzła, natomiast zestaw składający się z 5 węzłów pozwala na awarię 2 węzłów.

W przypadku dostępności mniejszej liczby węzłów w zestawie, wszystkie pozostałe przechodzą w tryb *Secondary* - zestaw działa w trybie *read-only*. Mechanizm ten służy zabezpieczeniu danych przed brakiem lub niewystarczającą replikacją w systemie.

2.1.4 Dostępność

Poprawne działanie węzłów monitorowane jest za pomocą *Heartbeats*. Instancje co 2 sekundy wzajemnie informują się o poprawnym działaniu, brak informacji przez 10 sekund traktowany jest jako awaria węzła.



Rysunek 4: Monitorowanie działania węzłów

3 Implementacja bazy danych w środowisku MongoDB

Celem implementacji było stworzenie zestawu replik składającego się z 3 procesów *mongod* działających na niezależnych instancjach.

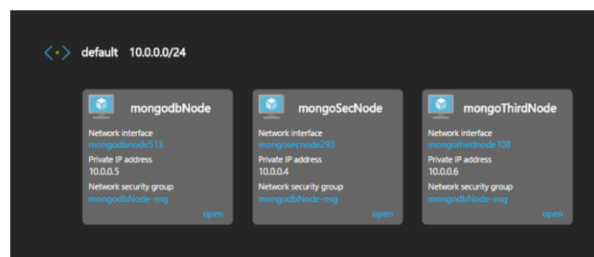
3.1 Infrastruktura

3.1.1 Instancje

Baza danych oparta została o chmurę *Azure* w modelu IaaS - *infrastruktura jako usługa*. Utworzone zostały 3 maszyny wirtualne z wykorzystaniem obrazów dostarczanych przez *Bitnami*. Systemem operacyjnym działającym na maszynach wirtualnych jest *Ubuntu 14.04*, natomiast wersja MongoDB to 3.2.9.

3.1.2 Sieć wewnętrzna

Maszyny wirtualne zostały połączone przez prywatną sieć wirtualną, zilustrowaną poniżej.



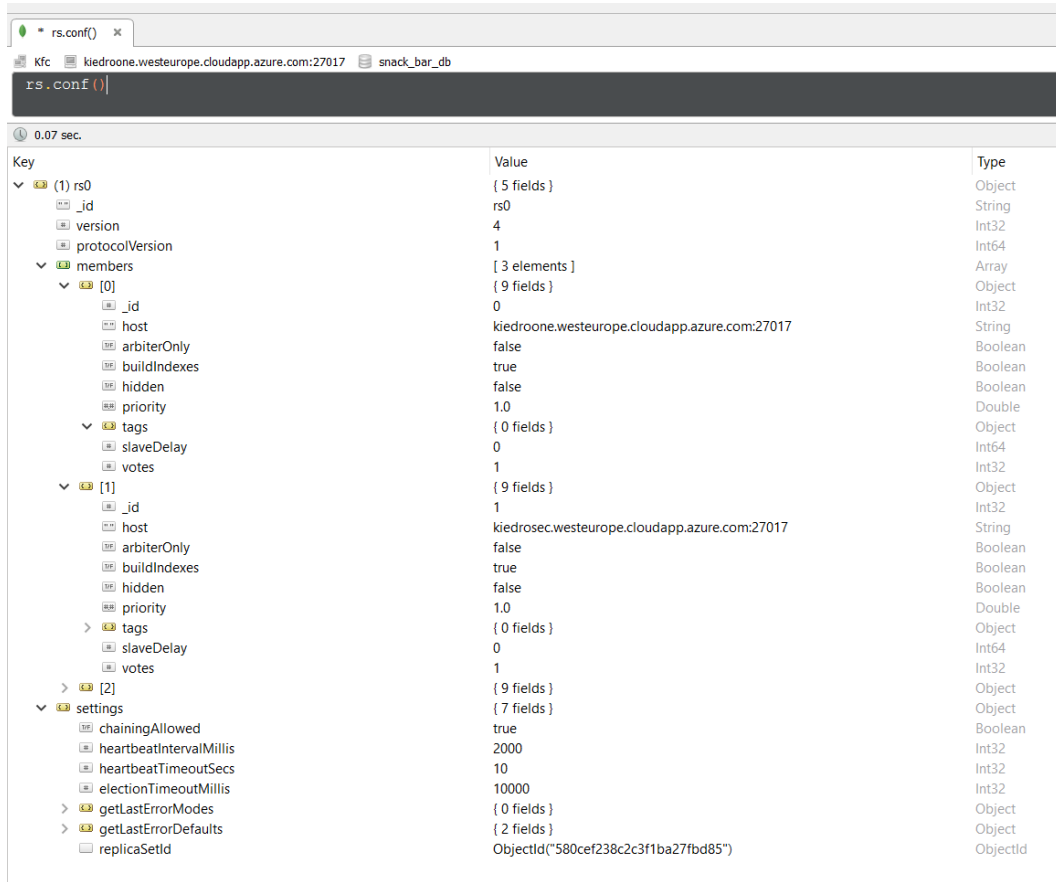
Rysunek 5: Sieć wirtualna

3.1.3 Sieć publiczna

Aby umożliwić połączenie z zestawem replik od dowolnego klienta, bez wymogu jego wdrożenia w chmurze *Azure*, każdej z instancji maszyn wirtualnych został przydzielony publiczny adres IP, oraz nazwa DNS umożliwiająca dostęp do maszyn.

3.2 Konfiguracja MongoDB

Korzystając z *mongo shell* utworzony został zestaw replik oraz jego konfiguracja.



Rysunek 6: Konfiguracja zestawu replik

Korzystając z pliku konfiguracyjnego uaktywiono prosty interfejs REST, pozwalający na łatwą diagnozę zestawu replik.

[Home](#) | [View Replset Config](#) | [replSetGetStatus](#) | [Docs](#)

Set name: rs0
Majority up: yes

Member	id	Up	cctime	Last heartbeat	Votes	Priority	State	Messages	optime
kiedroone.westeurope.cloudapp.azure.com:27017	0	1	35 mins	1 sec	1	1.000000	SECONDARY		(term: 15, timestamp: Jan 22 15:44:02:1)
kiedrosec.westeurope.cloudapp.azure.com:27017 (me)	1	1	35 mins		1	1.000000	PRIMARY		(term: 15, timestamp: Jan 22 15:44:02:1)
kiedrothird.westeurope.cloudapp.azure.com:27017	2	0	0 secs	0 secs	1	1.000000	(was DOWN)	Network is unreachable	(term: -1, timestamp: Jan 1 00:00:00:0)

Recent replset log activity:

```

2017-01-22T15:43:50.729+0000 I REPL [ReplicationExecutor] New replica set config in use: { _id: "rs0", version: 4, protocolVersion: 1, members: [ { _id: 0, host: "kiedroone.westeurope.cl
15:43:50.729+0000 I REPL [ReplicationExecutor] transition to STARTUP2
15:43:50.767+0000 I REPL [ReplicationExecutor] transition to RECOVERING
15:43:50.768+0000 I REPL [ReplicationExecutor] transition to SECONDARY
15:43:50.823+0000 I REPL [ReplicationExecutor] Member kiedroone.westeurope.cloudapp.azure.com:27017 is now in state SECONDARY
15:44:01.910+0000 I REPL [ReplicationExecutor] transition to PRIMARY
15:44:02.772+0000 I REPL [rsSync] transition to primary complete; database writes are now permitted

```

Rysunek 7: Interfejs REST

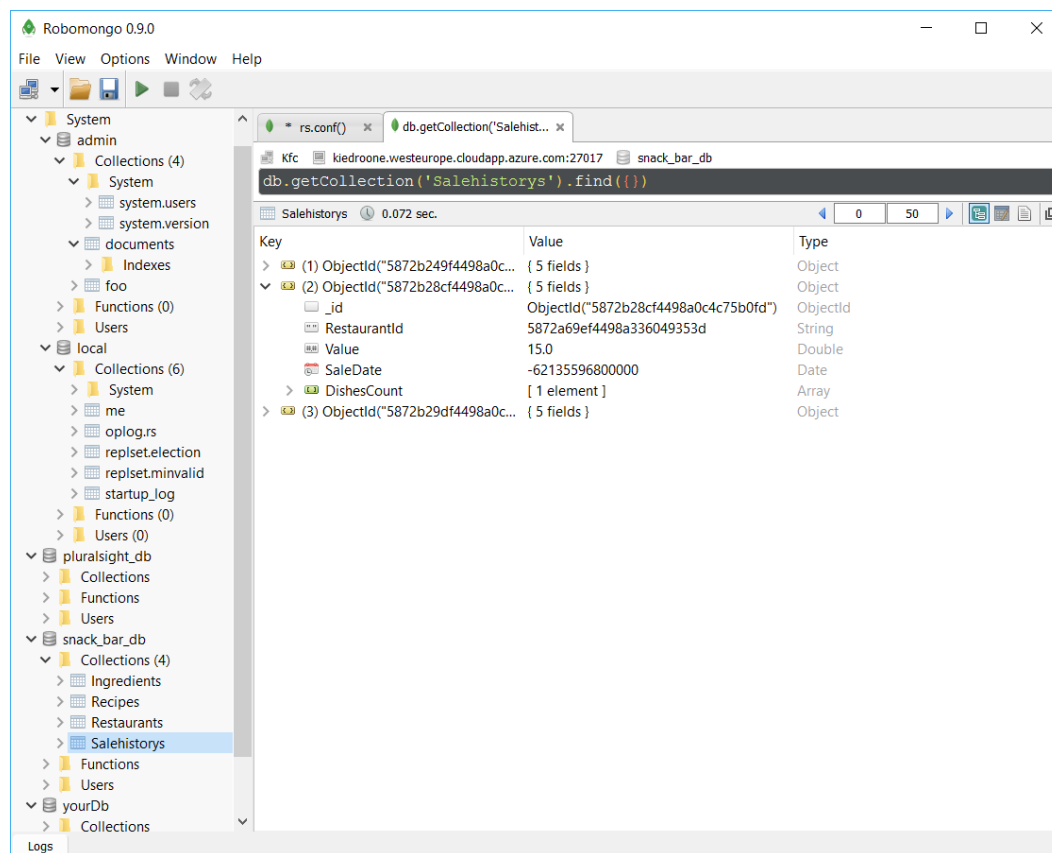
3.3 Schemat bazy danych

MongoDB należy do baz NoSQL typu dokumentowego, jedną z cech charakterystycznych baz tego typu jest brak ustalonego schematu bazy danych. Kolekcje tworzone dynamicznie w trakcie działania aplikacji pozwalają na przechowywanie dowolnych danych - jedynym warunkiem jest możliwość ich serializacji w formacie JSON.

Ta cecha baz NoSQL upraszcza wstępną konfigurację bazy danych.

3.4 Weryfikacja

Do weryfikacji poprawnego działania bazy danych - możliwości połączenia ze wszystkimi węzłami, odczytu, zapisu, działania mechanizmu replikacji posłużyło narzędzie *Robomongo* w wersji 0.9.



Rysunek 8: Robomongo