AKADEMIA GÓRNICZO-HUTNICZA IM. STANISŁAWA STASZICA W KRAKOWIE

Wydział Informatyki

INSTYTUT INFORMATYKI

Środowiska Udostępniania Usług

Grupa 4 - czwartek 9:45

Operator Framework - studium przypadku technologii

Operator Framework - a case study in technology

Dominika Bocheńczyk

Mateusz Łopaciński

Piotr Magiera

Michał Wójcik

Kraków, 2024

Spis treści

Rozdział 1

Wprowadzenie

Kubernetes to niekwestionowany lider w segmencie automatyzacji i orkiestracji aplikacji kontenerowych, pełniąc kluczową rolę w skalowalnym wdrażaniu oprogramowania na infrastrukturę sieciową. Doskonale wpisuje się w trend zastępowania architektur monolitycznych przez mikroserwisy, które znacząco zwiększają reaktywność i elastyczność systemów informatycznych. Istnieje jednakże pewien podzbiór aplikacji, dla którego pojawiają się wyzwania związane z automatyzacją ich obsługi, zwłaszcza w sytuacji awarii lub dynamicznego zwiększania obciążenia. Opisane aplikacje, to oprogramowanie typu *stateful*, takie jak bazy danych (Postgres, MySQL, Redis), middleware (RabbitMQ) czy systemy monitorowania (Prometheus), których stan jest krytyczny dla ciągłości działania i nie może zostać utracony w przypadku awarii.

Rysunek 1.1: Przykładowe aplikacje typu stateful

Zaproponowane przez twórców Kubernetes rozwiązania, takie jak Stateful Sets połączone z Persistent Volumes, umożliwiają utrzymanie danych na dysku i relacji master-slave między replikami baz danych, jednakże ich konfiguracja i zarządzanie mogą być złożone i czasochłonne, co utrudnia w pełni automatyczne zarządzanie cyklem życia tych aplikacji. Ponadto, standardowe narzędzia Kubernetes nie zawsze dostarczają wystarczających możliwości zarządzania stanem aplikacji, co może skutkować koniecznością utrzymywania systemów bazodanowych poza klastrem Kubernetes, co jest niezgodne z ideą Infrastructure as Code.

Rozdział 2

Podstawy teoretyczne i stos technologiczny

2.1. Podstawy teoretyczne

Operator Framework rozszerza możliwości Kubernetes, dostarczając narzędzia umożliwiające tworzenie operatorów – specjalistycznych programów, które zarządzają innymi aplikacjami wewnątrz klastra Kubernetes. Operatorzy są projektowani tak, aby w sposób ciągły monitorować stan aplikacji, automatycznie podejmując decyzje o koniecznych działaniach naprawczych, skalowaniu, aktualizacji lub konfiguracji w odpowiedzi na zmieniające się warunki operacyjne.

Istotą Operator Framework jest umożliwienie automatyzacji operacji, które tradycyjnie wymagałyby ręcznego przeprowadzenia przez zespoły operacyjne lub administratorów systemów. Przykładowo, operator bazy danych nie tylko zarządza replikacją danych, ale również może automatycznie zarządzać schematami bazy danych, przeprowadzać rotację certyfikatów, czy realizować procedury backupu i przywracania danych.

Jako że każda aplikacja typu stateful może posiadać specyficzny sposób zarządzania, potrzebuje ona swojego własnego Operatora. Z tego względu istnieje nawet publiczne repozytorium, z którego można pobrać konfiguracje opracowane pod konkretne oprogramowanie (znajduje się ono pod linkiem https://operatorhub.io/).

Wzorzec Operator pozwala na łączenie kontrolerów jednego lub więcej zasobów aby rozszerzyć zachowanie klastra bez konieczności zmiany implementacji. Operatorzy przyjmują rolę kontrolerów dla tzw. Custom Resource. Custom Resource rozszerzają / personalizują konkretne instalacje Kubernetesa, z tym zachowaniem że użytkownicy mogą z nich korzystać jak z wbudowanych już zasobów (np. Pods). [1]

2.2. Stos technologiczny

Głównym komponentem technologicznym naszego projketu jest Operator Framework, który zapewnia zestaw narzędzi, szablonów i wytycznych, ułatwiających programistom tworzenie operatorów, co przyspiesza proces tworzenia nowych aplikacji i usprawnia zarządzanie nimi w środowiskach Kubernetes.

W przypadku wyłącznie prezentacji działania Operatora na klastrze Kubernetesowym, możemy skorzystać z narzędzia jakim jest minikube. Minikube pozwala na szybkie stworzenie lokalnego klastra Kubernetesa w danym systemie operacyjnym. Dzięki temu, mając jedynie kontener Dockerowy lub środowisko maszyny wirtualnej, możemy lepiej skupić się na samej funkcjonalności Kubernetesa dla naszych potrzeb. [2]

Rozdział 3

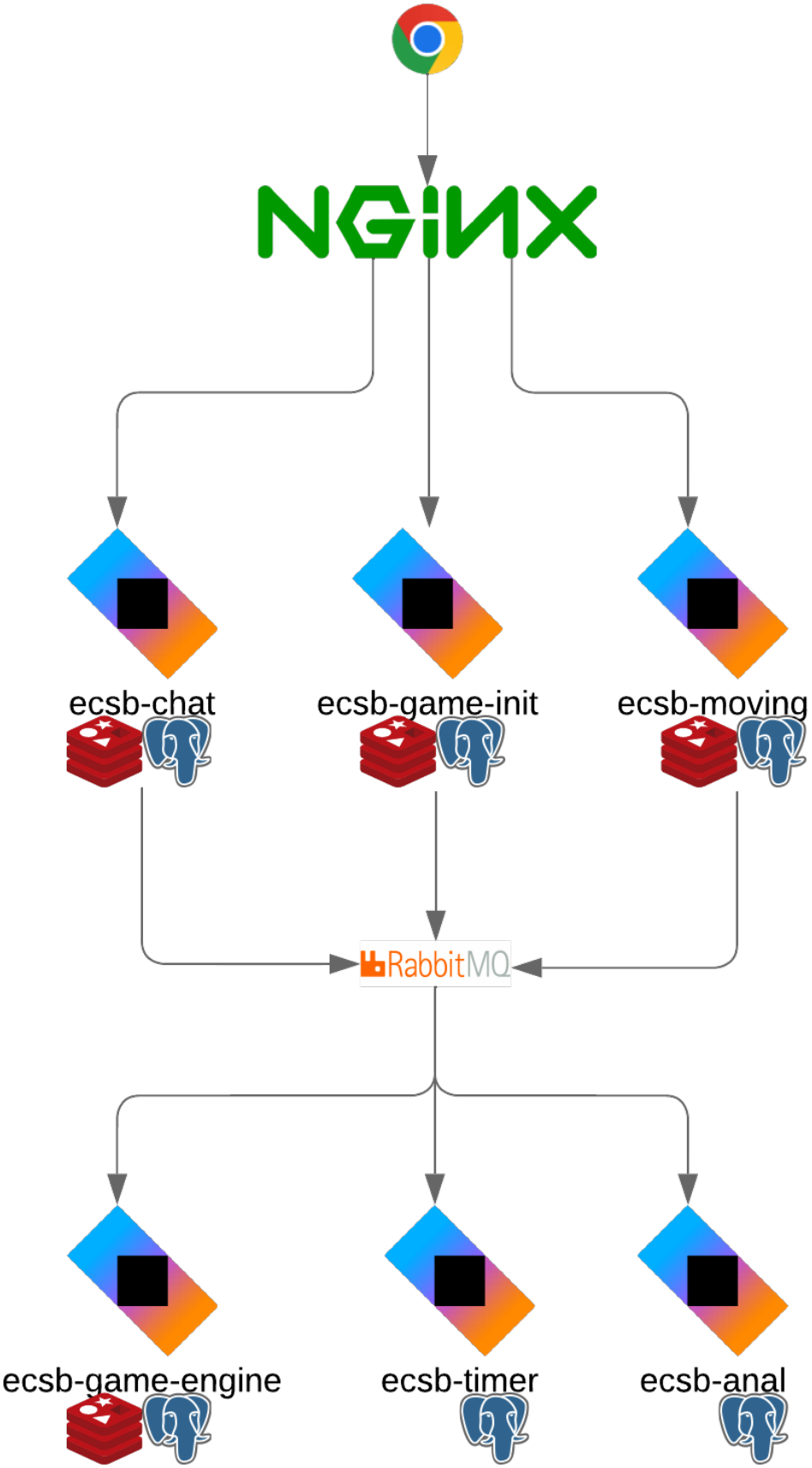
Opis studium przypadku

3.1. Przykład aplikacji typu stateful

Aplikacją, na której pokażemy działanie Operatora, będzie Elektroniczna Chłopska Szkoła Biznesu (eCSB), cyfrowa implementacja gry planszowej 'Chłopska Szkoła Biznesu' wydanej przez Małopolski Instytut Kultury. Aplikacja ta jest pracą inżynierską czworga studentów naszego Wydziału, obecnie kontynuowaną w ramach pracy magisterskiej. Gra polega na produkcji zasobów według przydzielonego zawodu, handlu towarami, zakładaniu spółek oraz odbywaniu wypraw w celu korzystniej wymiany produktów na pieniądze. W czasie kilkunastominutowej rozgrywki każdy z graczy stara się zgromadzić jak największy majątek.

Rysunek 3.1: Ekran powitalny ECSB

eCSB jest aplikacją webową opartą o mikroserwisy oraz komunikację z wykorzystaniem wiadomości. W minimalnym wariancie składa się z 6 bezstanowych modułów napisanych w języku Kotlin, podłączonych do bazy danych Postgres, bazy klucz-wartość Redis oraz brokera wiadomości RabbitMQ, a także aplikacji webowej, komunikującej się z modułami poprzez protokoły REST oraz WebSocket. Dostęp do wszystkich elementów architektury realizowany jest dzięki serwerowi HTTP nginx, który pełni rolę reverse-proxy (zapewniając przy tym certyfikaty SSL i ruch HTTPS) oraz API gateway (przekierowując żądania do odpowiednich mikroserwisów). Pełna architektura rozwiązania przedstawiona jest poniżej 3.2.



Rysunek 3.2: Architektura projektu eCSB

Warto dodać, że niektóre z modułów zostały zaprojektowane z myślą o skalowaniu systemu i rozkładaniu obciążenia. Są to moduły chat, moving oraz game-engine. Pozostałe 3 moduły odpowiadają za usługi stosunkowo rzadkie (tworzenie sesji gry) lub w pełni scentralizowane (zbieranie logów, odświeżanie czasu gry).

3.2. Przypadek użycia Operatora

Naszym scenariuszem, poprzez który zademonstrujemy działanie Operatora dla aplikacji eCSB, będą następujące operacje:

1. uruchomienie klastra Kubernetesowego z minimalnym wariantem eCSB (6 modułów)
2. potrojenie skalowalnych modułów (moving, chat, game-engine), a następnie powrót do wariantu minimalnego
3. przywrócenie centralnego modułu po awarii (timer lub game-init)

Rozdział 4

Architektura rozwiązania

Rozdział 5

Konfiguracja środowiska

Rozdział 6

Sposób instalacji

Rozdział 7

Odtworzenie rozwiązania

Rozdział 8

Wdrożenie wersji demonstracyjnej

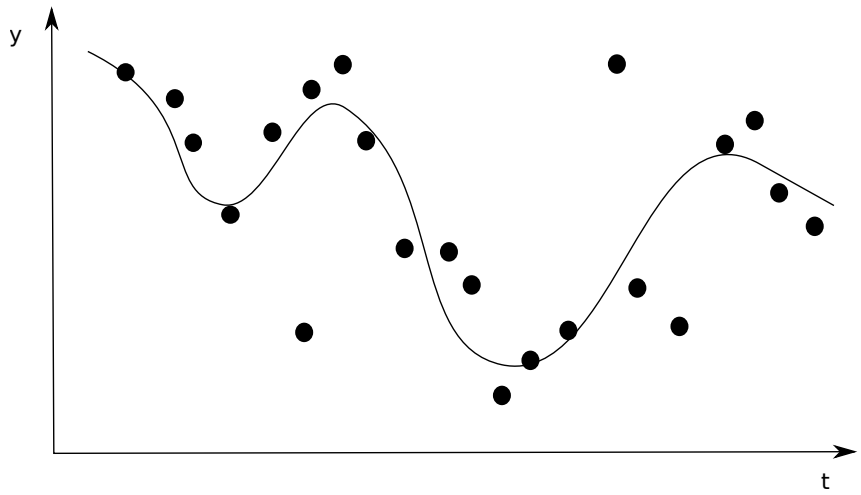
Rozdział 9

Podsumowanie

9.1. Rysunki, tabele

9.1.1. Rysunki

Przykładowy odnośnik do rysunku 9.1.



Rysunek 9.1: Przykładowy rysunek (źródło: [5])

W przypadku rysunków można odwoływać się zarówno do poszczególnych części składowych — rysunek 9.2(a) i rysunek 9.2(b)) — jak i do całego rysunku 9.2.

9.1.2. Tabele

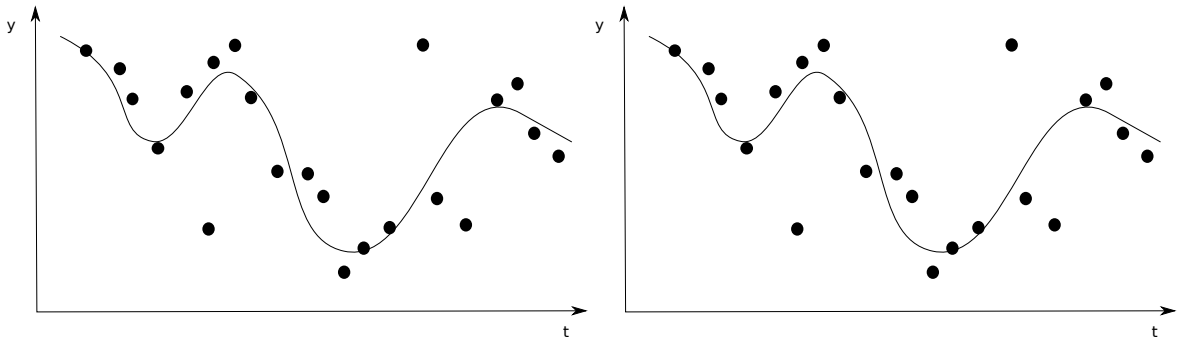
Przykładowa tabela 9.1.

9.2. Wzory matematyczne

Przykład wzoru z odnośnikiem do literatury [5]:

$$\Omega = \sum_{i=1}^{n} \gamma_i \tag{9.1}$$

Przykładowy odnośnik do wzoru (9.1).



(a) Tytuł 1 (b) Tytuł 2

Rysunek 9.2: Kolejne przykładowe rysunki (źródło: [5])

Tabela 9.1: Przykładowa tabela

	Best		**Average**		**Worst**	
No.	**AB**	**CD**	**FE**	**GH**	**IJ**	**KL**
1.	10	89	58	244	6	70
2.	15	87	57	147	4	82
3.	23	45	55	151	2	38
4.	34	90	55	246	1	82
5.	56	75	54	255	0	73

Przykładowy wzór w tekście $\lambda = \sum_{i=1}^{n} \delta_i$, bez numeracji.

9.3. Cytowania

Przykład cytowania literatury [9]. Kolejny przykład cytowania kilku pozycji bibliograficznych [3, 10, 7, 8, 4].

9.4. Listy

Lista z elementami:

- pierwszym,
- drugim,
- trzecim.

Lista numerowana z dłuższymi opisami:

1. Pierwszy element listy.
2. Drugi element listy.
3. Trzeci element listy.

9.5. Algorytmy

Algorytm 1 przedstawia przykładowy algorytm zaprezentowany w [6].

Algorytm 1: Przykładowy algorytm (źródło: [6]).

input : A bitmap Im of size $w \times l$
output A partition of the bitmap
:

```
1  special treatment of the first line;
2  for i ← 2 to l do
3      special treatment of the first element of line i;
4      for j ← 2 to w do
5          left ← FindCompress(Im[i, j − 1]);
6          up ← FindCompress(Im[i − 1, ]);
7          this ← FindCompress(Im[i, j]);
8          if left compatible with this then // O(left,this)==1
9              if left < this then Union(left,this);
10             else Union(this,left);
11         end
12         if up compatible with this then                    // O(up,this)==1
13             if up < this then Union(up,this);
               // this is put under up to keep tree as flat as possible
14             else Union(this,up);
               // this linked to up
15         end
16     end
17     foreach element e of the line i do FindCompress(p);
18 end
```

9.6. Fragmenty kodu źródłowego

Listing 9.1 przedstawia przykładowy fragment kodu źródłowego.

```
# The maximum of two numbers

def maximum(x, y):

    if x >= y:
        return x
    else:
        return y

x = 2
y = 6
print(maximum(x, y),"is the largest of the numbers ", x, " and ", y)
```

Listing 9.1: Przykładowy fragment kodu (źródło: [5])

Bibliografia

[1] 2024. URL: https://kubernetes.io/docs/concepts/extend-kubernetes/operator/.

[2] 2024. URL: https://minikube.sigs.k8s.io/docs/start/.

[3] F. Allen i R. Karjalainen. „Using Genetic Algorithms to Find Technical Trading Rules". W: *Journal of Financial Economics* 51.2 (1999), s. 245–271.

[4] G. Chmaj i H. Selvaraj. „Distributed Processing Applications for UAV/Drones: A Survey". W: *Progress in Systems Engineering. Advances in Intelligent Systems and Computing*. Red. C. G. Selvaraj H. Zydek D. T. 366. Springer, Cham, 2015, s. 449–454. DOI: 10.1007/978-3-319-08422-0_66.

[5] A. Exemplary. *Exemplary title of the book*. Address of the publisher: Publisher, 2021.

[6] C. Fiorio. *algorithm2e.sty – package for algorithms*. 2017. URL: http://mirrors.ctan.org/macros/latex/contrib/algorithm2e/doc/algorithm2e.pdf.

[7] V. Pictet O. *Genetic Algorithms with Collective Sharing for Robust Optimization in Financial Applications*. Technical report. Olsen & Associates, 1995.

[8] F. Wilhelmstötter. *Jenetics*. 2021. URL: https://jenetics.io/ (term. wiz. 28. 01. 2021).

[9] G. Wilson i W. Banzhaf. „Prediction of Interday Stock Prices using Developmental and Linear Genetic Programming". W: *Applications of Evolutionary Computing. EvoWorkshops 2009: EvoCOMNET, EvoENVIRONMENT, EvoFIN, EvoGAMES, EvoHOT, EvoIASP, EvoINTERACTION, EvoMUSART, EvoNUM, EvoSTOC, EvoTRANSLOG, Tübingen, Germany, April 15-17, 2009, Proceedings*. Red. M. Giacobini i in. T. 5484. LNCS. Berlin, Heidelberg: Springer-Verlag, 2009, s. 172–181.

[10] E. Zitzler. „Evolutionary Algorithms for Multiobjective Optimization: Methods and Applications". PhD thesis. ETH Zurich, 1999.

Spis rysunków

Spis tabel

Spis algorytmów

Spis listingów