

Studium Magisterskie

Kierunek: Analiza danych Big Data

Patryk Rutkowski

Nr albumu 116655

**Kubernetes – opis i wykorzystanie narzędzia**

Praca magisterska

napisana w Instytucie

Statystyki i Demografii

pod kierunkiem naukowym

dr. Sebastiana Zająca

Warszawa 2023

Spis treści

[Wstęp 6](#_Toc134354862)

[Rozdział I. Podstawowe pojęcia, definicje oraz instalacja 7](#_Toc134354863)

[I.1 Definicje 7](#_Toc134354864)

[I.2 Historia 8](#_Toc134354865)

[I.3 Komponenty 8](#_Toc134354866)

[I.4 Zastosowanie 10](#_Toc134354867)

[I.5 Instalacja i konfiguracja 10](#_Toc134354868)

[Rozdział II. Praca z Kubernetes 13](#_Toc134354869)

[II.1 Tworzenie i wdrożenie aplikacji w środowisku Kubernetes 13](#_Toc134354870)

[II.2 Zarządzanie I monitorowanie aplikacji w środowisku Kubernetes 15](#_Toc134354871)

[II.3 Lorem ipsum dolor sit amet, consectetur 15](#_Toc134354872)

[II.3.1.1 Lorem ipsum dolor sit amet, consectetur 16](#_Toc134354873)

[II.3.1.2 Lorem ipsum dolor sit amet, consectetur adipisicing elit 16](#_Toc134354874)

[II.3.1.3 Onsectetur adipisicing eli 17](#_Toc134354875)

[II.3.1.4 Aliquip ex ea commod 18](#_Toc134354876)

[Rozdział III. Współpraca Kubernetes z innymi aplikacjami oraz analiza porównawcza narzędzi 19](#_Toc134354877)

[III.1 Integracja Kubernetes z innymi narzędziami i usługami 20](#_Toc134354878)

[III.2 Analiza porównawcza Kubernetes z innymi narzędziami do zarządzania aplikacjami 20](#_Toc134354879)

[III.2.1.1 Aut labore et 20](#_Toc134354880)

[III.2.1.2 Oroident, proident 21](#_Toc134354881)

[III.3 Excepteur pxcepteur 21](#_Toc134354882)

[III.3.1.1 Asint sint sint sint 21](#_Toc134354883)

[III.3.1.2 Abcc ccsscssc sdsdza 22](#_Toc134354884)

[III.3.1.3 Kdsjdjs adsdfs dfsdfh 23](#_Toc134354885)

[III.3.1.4 Loriem lori trinume trie 24](#_Toc134354886)

[III.4 Loriem loriem loriem loriem loriem loriem loriem loriem loriem 25](#_Toc134354887)

[III.4.1.1 Duis duis duis duis 25](#_Toc134354888)

[Zakończenie 26](#_Toc134354889)

[LITERATURA 28](#_Toc134354890)

[Spis tabel 31](#_Toc134354891)

[Spis rysunków 31](#_Toc134354892)

[Streszczenie 32](#_Toc134354893)

[Summary 32](#_Toc134354894)

Wstęp

Niniejsza praca ma na celu omówienie Kubernetes, czyli narzędzia do orkiestracji kontenerów służących do tworzenia, wdrażania i zarządzania aplikacjami w środowiskach rozwojowych i produkcyjnych. Zgłębione zostaną elementy Kubernetes takie jak np. węzły (nodes), kontenery (pods), kontrolery replikacji (replication controllers), serwisy (services) oraz wdrożenia (deployments). W pracy zostaną omówione przykłady zastosowania, wzorce projektowe oraz antywzorce, co stanowić będzie solidny wstęp do pracy z omawianym narzędziem. Porównane zostaną inne konkurencyjne do Kubernetes narzędzia wraz z omówieniem wad i zalet każdego. Przybliżone zostaną również narzędzia oraz obszary IT związane z tematem Kubernetes, jak na przykład Docker, kontenery, wirtualizacja, mikroserwisy, technologie chmurowe. Wspomniana zostanie historia tworzenia tych narzędzi celem zobrazowania powodu dlaczego została stworzona jak również jak wyglądała praca przy rozwoju aplikacji przed komercjalizacją Kubernetes i konteneryzacji.

# Podstawowe pojęcia, definicje oraz instalacja

Współczesne technologie informatyczne rozwijają się w niezwykle szybkim tempie, a konteneryzacja i orkiestracja kontenerów stały się kluczowymi elementami architektury nowoczesnych systemów komputerowych. Niniejszy rozdział ma na celu wprowadzenie podstawowych pojęć związanych z Kubernetes, zarysowanie jego historii oraz nawiązanie do poprzednich rozwiązań spełniających podobne role, omówienie komponentów z których się składa, ich zastosowania oraz instalacji i konfiguracji.

## Definicje

1. Kubernetes

“Kubernetes to przenośna, rozszerzalna platforma oprogramowania open-source służąca do zarządzania zadaniami i serwisami uruchamianymi w kontenerach, która umożliwia deklaratywną konfigurację i automatyzację. Ekosystem Kubernetesa jest duży i dynamicznie się rozwija. Usługi dla Kubernetesa, wsparcie i narzędzia są szeroko dostępne.”[[1]](#footnote-1) Skrót "K8s" został utworzony przez zastąpienie ośmiu liter znajdujących się między "K" a "s" cyfrą 8.

2. Kontener

„Kontener to standardowa jednostka oprogramowania, która pakuje kod i wszystkie jego zależności, aby aplikacja działała szybko i niezawodnie w różnych środowiskach komputerowych.”[[2]](#footnote-2)

3. Wirtualizacja

„Wirtualizacja to technologia, która pozwala na tworzenie użytecznych usług IT, wykorzystując zasoby tradycyjnie związane ze sprzętem. Umożliwia ona pełne wykorzystanie możliwości fizycznej maszyny poprzez dystrybucję jej zdolności wśród wielu użytkowników lub środowisk.”[[3]](#footnote-3)

## Historia

Kubernetes został stworzony w Google w 2014 roku, jako efekt pracy nad wcześniejszymi Borg I Omega, a aktualnie jest utrzymywany przez Cloud Native Computing Foundation.

Przez lata Google opracowywało wewnętrzne systemy, takie jak Borg oraz późniejszy system Omega, które wspierały deweloperów i administratorów systemów w zarządzaniu tysiącami aplikacji i usług. Systemy te nie tylko ułatwiały rozwój i zarządzanie, ale także pozwalały na osiągnięcie wyższego stopnia wykorzystania infrastruktury - kluczowego aspektu dla tak rozległej organizacji. Dysponując setkami tysięcy maszyn, nawet drobne ulepszenia w wykorzystaniu infrastruktury przekładają się na oszczędności rzędu milionów dolarów, co stanowi znaczącą motywację do opracowania takich rozwiązań. Po utrzymaniu Borga i Omegi w tajemnicy przez dekadę, w 2014 roku Google wprowadziło Kubernetes - system open-source oparty na doświadczeniach zdobytych z Borg, Omega i innych wewnętrznych systemów firmy.[[4]](#footnote-4)

Przed stworzeniem I popularyzacją mikroserwisów przy wdrażaniu I utrzymywaniu aplikacji wykorzystywana była architektura monilityczna. Aplikacje monolityczne ze ściśle powiązanymi komponentami wymagają wspólnego tworzenia, wdrażania i zarządzania jako jednolity proces OS. Zmiany w części aplikacji powodują konieczność ponownego wdrożenia całego systemu, a brak wyraźnych granic zwiększa złożoność oraz pogarsza jakość. Aplikacje monolityczne działają na niewielu wydajnych serwerach, skalowane pionowo (dodawanie procesorów, pamięci) lub poziomo (dodatkowe serwery, replikowanie aplikacji). Skalowanie pionowe jest kosztowne, a skalowanie poziome może być trudne (np. bazy danych relacyjne).

Rozwiązaniem są mikrousługi - mniejsze, niezależnie wdrażane komponenty komunikujące się przez proste API. Skalowanie mikrousług odbywa się per-service, umożliwiając skalowanie tylko potrzebujących więcej zasobów. Wady mikrousług obejmują trudności w zarządzaniu, konfiguracji, debugowaniu i śledzeniu wywołań w rozbudowanych systemach, choć niektóre z tych problemów są rozwiązane przez rozproszone systemy śledzenia, takie jak Zipkin.

W architekturze mikroserwisów komponenty są rozwijane i wdrażane niezależnie. Z uwagi na autonomię i częste rozwijanie przez odrębne zespoły, możliwe jest korzystanie z różnych bibliotek i ich wymiana w razie potrzeby. Różnice w zależnościach między komponentami prowadzą do sytuacji, gdzie aplikacje wymagają różnych wersji tych samych bibliotek.

Wdrażanie dynamicznie łączonych aplikacji z różnymi wersjami bibliotek współdzielonych lub specyficznymi warunkami środowiskowymi może stanowić wyzwanie dla zespołu operacyjnego zarządzającego serwerami produkcyjnymi. Zwiększająca się liczba komponentów wdrożonych na tym samym hoście sprawia, że zarządzanie zależnościami i spełnienie wszystkich wymagań staje się bardziej złożone.

## Komponenty

## Zastosowanie

## Instalacja i konfiguracja

Kubernetes zainstalować można na wiele sposobów, na przykład za pomocą chmur publicznych (Elastic Kubernetes Service w Amazone Web Services, Google Kubernetes Engine lub Azure Kubernetes Service) lub lokalnie przy pomocy narzędzia minikube. Jako przykład posłuży instalacja K8s w minikube oraz Google Kubernetes Engine.[[5]](#footnote-5)

Minikube jest narzędziem do tworzenia jednowęzłowego (single-node) klastra Kubernetes, więc nie użyjemy go z aplikacjami wielowęzłowymi (multiple-node), jednak sprawdzi się w testowaniu Kubernetes oraz tworzeniu aplikacji lokalnie. Minikube można zainstalować korzystając z instrukcji w repozytorium (<https://github.com/kubernetes/minikube>) oraz dokumentacji (<https://minikube.sigs.k8s.io/docs>). Można zainstalować go na OSX, Linux lub Windows, jednak żeby korzystać z minikube na Windows należy również mieć zainstalowany VirtualBox (<https://www.virtualbox.org/wiki/Downloads>) lub KVM (<https://www.linux-kvm.org/page/Downloads>). Po zainstalowaniu minikube lokalnie klaster Kubernetes włączamy za pomocą komendy “minikube start”. Żeby komunikować się z Kubernetes potrezebny jest również Kubernetes Client (KUBECTL). Aby sprawdzić czy klaster jest włączony należy użyć komendy “kubectl cluster-info”.[[6]](#footnote-6)

Google Kubernetes Engine jest wygodniejszym narzędziem do stawiania pierwszych kroków ponieważ nie trzeba ręcznie konfigurować wszystkich węzłów klastra i sieci. Korzystanie z GKE gwarantuje, że nie będziemy mieli do czynienia ze źle skonfigurowanym, niewłaściwie działającym lub częściowo działającym klastrem. Aby rozpocząć pracę z GKE należy

* mieć konto w Google Cloud Provider
* stworzyć nowy projekt w Google Cloud Platform Console
* skonfigurować metodę płatności
* włączyć Kubernetes Engine API
* pobrać i zainstalować Google Cloud SDK
* zainstalować kubectl (jak przy minikube)

**Podsumowanie**

# Praca z Kubernetes

## Tworzenie i wdrożenie aplikacji w środowisku Kubernetes

## Zarządzanie I monitorowanie aplikacji w środowisku Kubernetes

**Podsumowanie**

# Współpraca Kubernetes z innymi aplikacjami oraz analiza porównawcza narzędzi

## Integracja Kubernetes z innymi narzędziami i usługami

## Analiza porównawcza Kubernetes z innymi narzędziami do zarządzania aplikacjami

Zakończenie

LITERATURA

**Książki**

1. *B. Burns, J. Beda, K. Hightower, Kubernetes: Up and Running, O’Reilly, Sebastopol USA 2019*
2. *M. Luksa, Kubernetes in Action, Manning, Shelter Island USA 2018*.

**Artykuły i studia**

1. Dębowski L.,*ZASTOSOWANIE KOMPUTERÓW W NAUCE I TECHNICE’ 2006*, Zeszyty Naukowe Wydziału Elektrotechniki i Automatyki Politechniki Gdańskiej Nr 22
2. IE. 1991. Competition in manufacturing leads to MRP II. 23 (July) 10-13
3. WJ Hopp, ML Spearman Commissioned Paper To Pull or Not to Pull: What Is the Question, Manufacturing & Service Operations Management, 2004

**Strony internetowe**

1. *https://kubernetes.io/pl/docs/concepts/overview/* [06-05-2023]
2. *https://www.docker.com/resources/what-container/* [06-05-2023]
3. *https://www.redhat.com/en/topics/virtualization/what-is-virtualization* [06-05-2023]
4. *https://github.com/kubernetes/minikube* [07-05-2023]
5. *https://minikube.sigs.k8s.io/docs* [07-05-2023]
6. *https://www.virtualbox.org/wiki/Downloads* [07-05-2023]
7. *https://www.linux-kvm.org/page/Downloads* [07-05-2023]

Spis tabel

[Tabela 1. Lorem ipsum dolor sit amet 14](#_Toc355852181)

[Tabela 2. Lorem ipsum dolor sit amet, consectetur adipisicing elit 15](#_Toc355852182)

Spis rysunków

[Rysunek 1. Lorem ipsum dolor sit amet Lorem ipsum dolor sit amet 17](#_Toc355852186)

[Rysunek 2. Lorem ipsum dolor sit amet, consectetur 22](#_Toc355852187)

Streszczenie

Summary

1. *https://kubernetes.io/pl/docs/concepts/overview/* [06-05-2023]. [↑](#footnote-ref-1)
2. *https://www.docker.com/resources/what-container/* [06-05-2023]. [↑](#footnote-ref-2)
3. *https://www.redhat.com/en/topics/virtualization/what-is-virtualization* [06-05-2023]. [↑](#footnote-ref-3)
4. *M. Luksa, Kubernetes in Action, Manning, Shelter Island USA 2018, str. 16* [↑](#footnote-ref-4)
5. *B. Burns, J. Beda, K. Hightower, Kubernetes: Up and Running, O’Reilly, Sebastopol USA 2019, str. 28* [↑](#footnote-ref-5)
6. *M. Luksa, Kubernetes in Action, Manning, Shelter Island USA 2018, str. 36-38* [↑](#footnote-ref-6)