



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.....	5
ROZDZIAŁ I. Podstawowe pojęcia, definicje oraz instalacja.....	6
I.1 Definicje.....	6
I.2 Historia.....	7
I.3 Komponenty.....	9
I.4 Zastosowanie.....	11
I.5 Instalacja i konfiguracja.....	12
ROZDZIAŁ II. Praca z Kubernetes.....	14
II.1 Tworzenie i wdrożenie aplikacji w środowisku Kubernetes	14
II.2 Zarządzanie i monitorowanie aplikacji w środowisku Kubernetes	14
ROZDZIAŁ III. Wdrożenie aplikacji w Kubernetes.....	15
III.1 Aplikacja	15
III.2 Jak skonfigurować Kubernetes i Horizontal Pod Autoscaling	15
Zakończenie	16
LITERATURA	16
Spis tabel	18
Spis rysunków.....	18
Streszczenie.....	19
Summary	19

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. Zgłębiane 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 dla którego została stworzona jak również jak wyglądała praca przy rozwoju aplikacji przed komercjalizacją Kubernetes i konteneryzacji.

ROZDZIAŁ I. 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.

I.1 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 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

3. Wirtualizacja

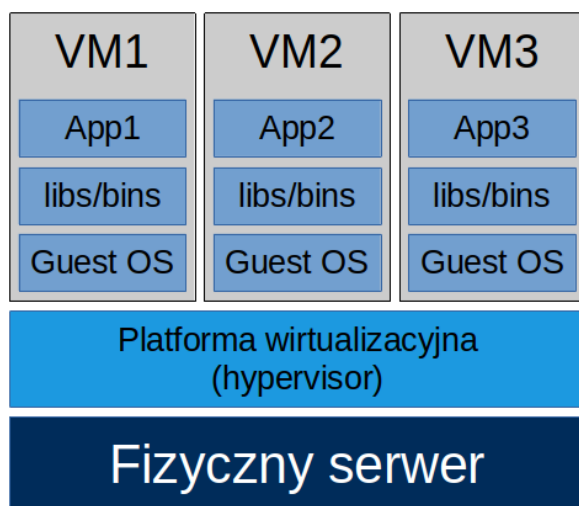
„Wirtualizacja to technologia, która pozwala na tworzenie użytecznych usług IT, wykorzystując zasoby tradycyjnie związane ze sprzętem. Umożliwia ona pełne

¹ <https://kubernetes.io/pl/docs/concepts/overview/> [06-05-2023].

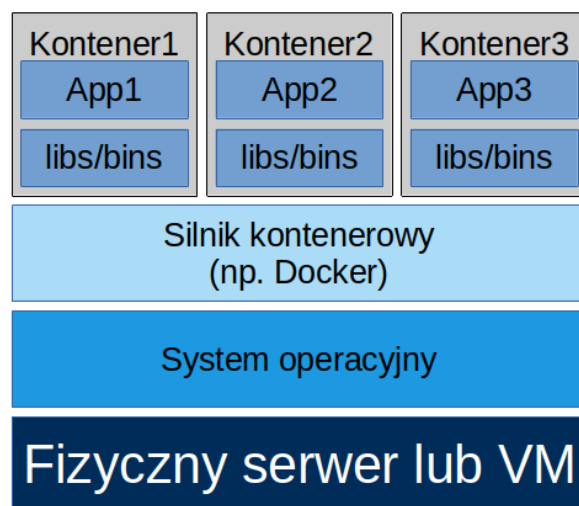
² <https://www.docker.com/resources/what-container/> [06-05-2023].

wykorzystanie możliwości fizycznej maszyny poprzez dystrybucję jej zdolności wśród wielu użytkowników lub środowisk.”³

Wirtualne maszyny



Kontenery



Źródło: <https://www.suse.com/c/pl/konteneryzacja-it-dla-laika-kubernetes-rancher-i-nie-tylko-czyli-jak-wykonac-skok/>

<https://www.suse.com/c/pl/konteneryzacja> 1

I.2 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

³ <https://www.redhat.com/en/topics/virtualization/what-is-virtualization> [06-05-2023].

Kubernetes - system open-source oparty na doświadczeniach zdobytych z Borg, Omega i innych wewnętrznych systemów firmy.⁴

Przed stworzeniem i popularyzacją mikroserwisów przy wdrażaniu i utrzymywaniu aplikacji wykorzystywana była architektura monolityczna. 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.

⁴ M. Luksa, *Kubernetes in Action*, Manning, Shelter Island USA 2018, str. 16

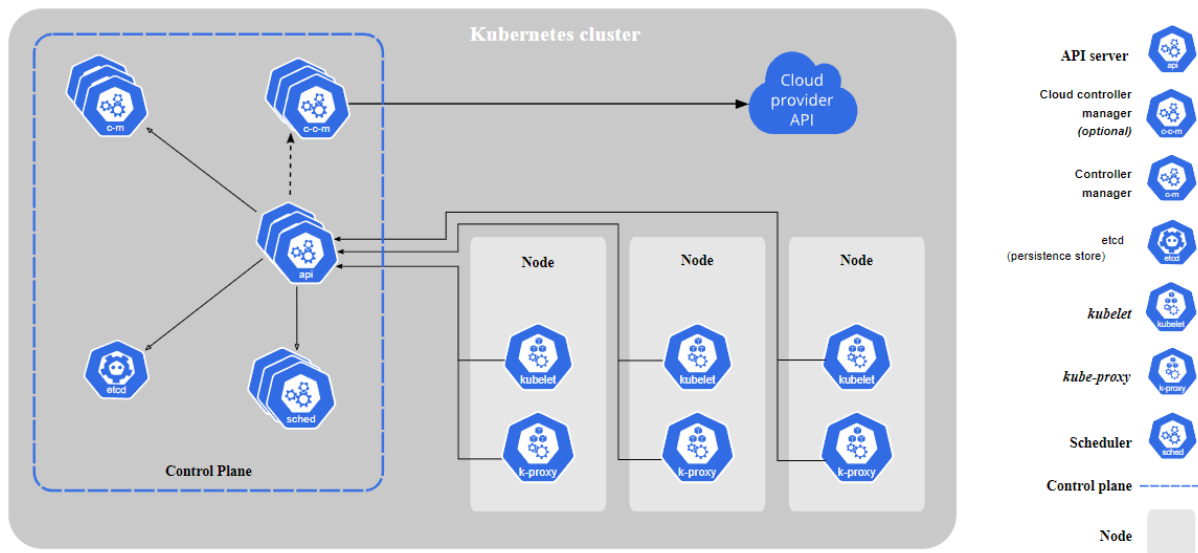
I.3 Komponenty

Każdy klaster Kubernetes składa się z węzłów, które mogą być fizycznymi maszynami lub wirtualnymi. Idea abstrakcji polega na tym, że niezależnie od rodzaju infrastruktury, na której działa aplikacja, Kubernetes jest w stanie ją wdrażać. Węzły pełnią rolę hostów dla podów, podstawowych jednostek w Kubernetes, które mogą być tworzone lub zarządzane. Każdy pod reprezentuje pojedynczą instancję aplikacji lub działający proces w Kubernetes i może zawierać jeden lub wiele kontenerów. Kubernetes zarządza cyklem życia kontenerów wewnątrz poda, włącznie z ich uruchamianiem, zatrzymywaniem i replikacją. Pody skupiają się na samej aplikacji, a nie na technicznych detalach kontenerów. Informacje dotyczące konfiguracji Kubernetes, w tym stanu kontenerów, są przechowywane w Etcd, czyli rozproszonym magazynie klucz-wartość. Tworzenie i usuwanie podów na węzłach dostosowuje się do bieżących potrzeb użytkownika. W tym procesie pomocne są kontrolery, które zarządzają kwestiami logistycznymi, takimi jak inicjowanie, skalowanie i zamykanie podów. Kontrolery różnią się w zależności od rodzaju obsługiwanej aplikacji. Przykładowo, StatefulSet jest używany dla aplikacji potrzebujących trwałego stanu, a Deployment służy do dynamicznego skalowania aplikacji w górę lub w dół, a także do aktualizacji lub cofania wersji aplikacji.

W celu sprawnego zarządzania cyklem życia aplikacji, potrzebujemy innego sposobu abstrakcji. Chcemy, aby aplikacja była trwała, nawet jeśli konkretne pody, które stanowią część aplikacji, powstają i znikają w odpowiedzi na zapotrzebowanie. W tym celu Kubernetes wprowadza koncepcję usług. Usługa to abstrakcyjny element, który grupuje pody logicznie i określa zasady dostępu do nich przez sieć. W rezultacie opisuje, jak grupa podów (lub innych obiektów Kubernetes) może być dostępna w sieci. Usługi pozwalają na elastyczne łączenie podów, które zależą od siebie nawzajem. Definiujemy je w formacie YAML lub JSON, podobnie jak inne obiekty w Kubernetes. Zestaw podów obsługiwanych przez usługę jest zwykle wybierany za pomocą selektora etykiet. Dzięki usługom, zmiany w zbiorze podów obsługujących aplikację nie wpływają na dostępność dla front-endu. Istnieją także inne elementy wewnętrzne Kubernetes. Mechanizm szeregowania równoważy obciążenie pomiędzy węzłami, dostosowując się do zasobów i potrzeb aplikacji. Menedżer kontrolera zapewnia zgodność stanu systemu (takiego jak aplikacje, obciążenie itp.) z pożądanym stanem zdefiniowanym w konfiguracji. Warto

zaznaczyć, że Kubernetes nie zastępuje niskopoziomowych mechanizmów kontenerów, takich jak Docker. Zamiast tego dostarcza wyższych abstrakcji, umożliwiając korzystanie z tych mechanizmów w celu utrzymania działających aplikacji w skali.⁵

Centralnym punktem działania Kubernetes są jego komponenty, które stanowią fundament jego funkcjonalności.



Źródło: <https://kubernetes.io/docs/concepts/overview/components/>

Komponenty master:

kube-apiserver – jest to główne API (interfejs programowania aplikacji) klastra Kubernetes. Stanowi most pomiędzy użytkownikiem a klastrem, pozwalając na zarządzanie i monitorowanie zasobów klastra poprzez API REST. Zadania obejmują przyjmowanie i weryfikację żądań API, zarządzanie metadanymi i przekazywanie operacji do odpowiednich komponentów klastra.

etcd - to magazyn klucz-wartość o silnej spójności, przechowujący konfigurację klastra i stan systemu. Etcd zapewnia trwałe przechowywanie informacji i działa jako źródło prawdy dla innych komponentów, umożliwiając koordynację i zarządzanie stanem klastra.

kube-scheduler - jego głównym zadaniem jest decydowanie, na jakim węźle w klastrze powinien zostać uruchomiony nowy pod (grupa kontenerów). W oparciu o różnorodne

⁵ <https://www.computerworld.pl/news/Czym-jest-Kubernetes,433339.html> [06-05-2023].

kryteria, takie jak dostępność zasobów, ograniczenia oraz wymagania aplikacji, kube-scheduler dokonuje optymalnych decyzji dotyczących dystrybucji zadań.

kube-controller-manager - ten komponent zarządza różnymi kontrolerami, które monitorują stan klastra i naprawiają ewentualne odchylenia od pożądanej konfiguracji. Przykładowe kontrolery obejmują kontroler replik, który zapewnia, że liczba replik podów pozostaje zgodna z oczekiwaną wartością.

cloud-controller-manager - w środowiskach chmurowych ten komponent zarządza integracją klastra Kubernetes z danym dostawcą chmury. Odpowiada za dostarczanie specyficznych dla dostawcy zasobów i funkcji do klastra, na przykład za zarządzanie zasobami chmurowymi, takimi jak maszyny wirtualne.

Komponenty node (węzła):

Kubelet – jest to agent działający na każdym węźle klastra. Zarządza cyklem życia podów, zapewniając, że są one uruchamiane i utrzymywane zgodnie z oczekiwaniami. Kubelet przekazuje również informacje o stanie węzła do kube-apiserver.

kube-proxy - jest odpowiedzialny za zarządzanie siecią na poziomie węzła. Tworzy reguły przekierowań i bilansowania obciążenia, aby umożliwić komunikację między podami i usługami w klastrze.

Container Runtime - odpowiada za uruchamianie kontenerów wewnątrz podów. Często używanymi kontenerowymi czasami są Docker, containerd lub rkt. To komponent, który wykonuje rzeczywiste wdrażanie kontenerów na węźle.

I.4 Zastosowanie

Podczas gdy Kubernetes został stworzony w celu efektywnej pracy z kontenerami w systemach Google, posiada on szerszy zakres funkcji i może być używany praktycznie przez każdego, niezależnie od tego, czy korzystają z Google Compute Engine na urządzeniach z systemem Android. Oferuje on wiele zalet, z których jedną z nich jest połączenie różnych narzędzi do wdrażania kontenerów, takich jak orkiestracja,

odkrywanie usług i równoważenie obciążenia. Kubernetes promuje interakcję między deweloperami, tworząc platformę do wymiany pomysłów na rozwijanie lepszych wersji. Dodatkowo, Kubernetes umożliwia łatwe wykrywanie błędów w kontenerach ze względu na swoją wersję beta. Oprócz orkiestracji kontenerów służyć on może również do zapewniania mechanizmów do tworzenia klastrów o wysokiej dostępności, co minimalizuje ryzyko awarii i przestojów. Jeśli jeden węzeł ulegnie awarii, system przeniesie obciążenie na inne węzły, utrzymując ciągłość działania aplikacji. Dzięki funkcjom takim jak Deploymenty, Kubernetes ułatwia wprowadzanie aktualizacji aplikacji oraz cofanie się do poprzednich wersji w razie problemów. To pozwala na płynne zarządzanie cyklem życia aplikacji. K8s umożliwia również przenoszenie aplikacji pomiędzy różnymi środowiskami, takimi jak prywatne centra danych i chmura publiczna. To idealne rozwiązanie dla firm, które chcą wykorzystać elastyczność chmury, jednocześnie zachowując kontrolę nad danymi.⁶

I.5 Instalacja i konfiguracja

Kubernetes zainstalować można na wiele sposobów, na przykład za pomocą chmur publicznych (Elastic Kubernetes Service w Amazonie 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.⁷

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

⁶ S. Fleming, *Kubernetes Handbook*, 2018 str. 17, 35

⁷ B. Burns, J. Beda, K. Hightower, *Kubernetes: Up and Running*, O'Reilly, Sebastopol USA 2019, str. 28

klaster Kubernetes włączamy za pomocą komendy “minikube start”. Żeby komunikować się z Kubernetes potrzebny jest również Kubernetes Client (KUBECTL). Aby sprawdzić czy klaster jest włączony należy użyć komendy “kubectl cluster-info”.⁸

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

⁸ M. Luksa, *Kubernetes in Action*, Manning, Shelter Island USA 2018, str. 36-38

ROZDZIAŁ II. Praca z Kubernetes

II.1 Tworzenie i wdrożenie aplikacji w środowisku Kubernetes

II.2 Zarządzanie i monitorowanie aplikacji w środowisku Kubernetes

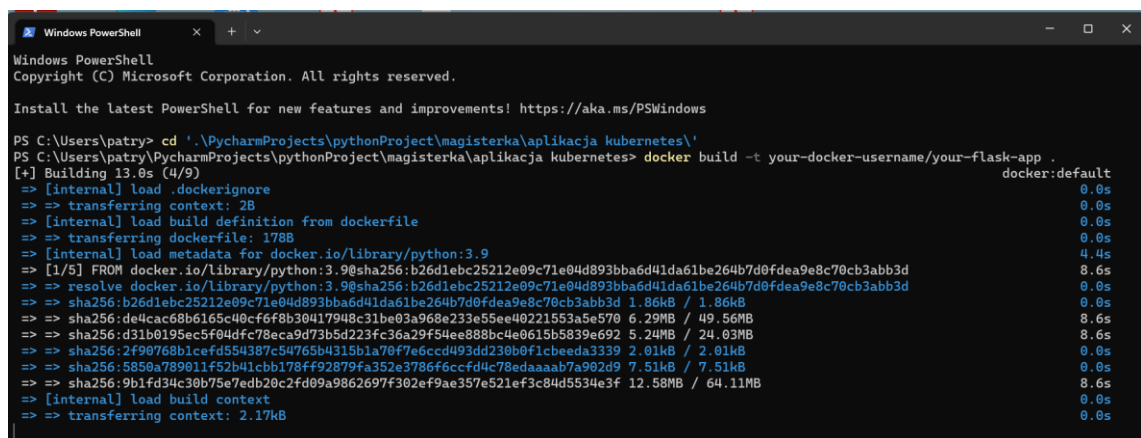
Podsumowanie

ROZDZIAŁ III. Wdrożenie aplikacji w Kubernetes

W tym rozdziale przybliżone zostanie wdrożenie Kubernetes do zarządzania kontenerami prostej aplikacji webowej z funkcją pokazywania adresu IP oraz możliwość odświeżania. Zasyta pod spodem funkcja sleep

III.1 Aplikacja

Aplikacja jest napisana w Python w bibliotece flask. Należało ją zdockeryzować czyli obudować w obraz docker, za pomocą `'docker build -t your-docker-username/your-flask-app .'`, co przedstawia poniższy zrzut ekranu



```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\patry> cd '.\PycharmProjects\pythonProject\magisterka\aplikacja kubernetes\'
PS C:\Users\patry\PycharmProjects\pythonProject\magisterka\aplikacja kubernetes> docker build -t your-docker-username/your-flask-app .
[+] Building 13.0s (4/9)
=> [internal] load .dockerignore                                docker:default 0.0s
=> => transferring context: 2B                                   0.0s
=> [internal] load build definition from dockerfile             0.0s
=> => transferring dockerfile: 178B                               0.0s
=> [internal] load metadata for docker.io/library/python:3.9    4.4s
=> [1/5] FROM docker.io/library/python:3.9@sha256:b26d1ebc25212e09c71e04d893bba6d41da61be264b7d0fdea9e8c70cb3abb3d 8.6s
=> => resolve docker.io/library/python:3.9@sha256:b26d1ebc25212e09c71e04d893bba6d41da61be264b7d0fdea9e8c70cb3abb3d 0.0s
=> => sha256:b26d1ebc25212e09c71e04d893bba6d41da61be264b7d0fdea9e8c70cb3abb3d 1.86kB / 1.86kB 0.0s
=> => sha256:de4cac68b6165c48cf6f8b30417948c31be03a968e233e55ee40221553a5e570 6.29MB / 49.56MB 8.6s
=> => sha256:d31b0195ec5f04dfc78eca9d73b5d223fc36a29f54ee888bc4e0615b5839e692 5.24MB / 24.03MB 8.6s
=> => sha256:2f90768b1cef554387c54765b4315b1a70f7e6ccd493dd230b0f1cbeeda3339 2.01kB / 2.01kB 0.0s
=> => sha256:5859a789011f52b41cbb178ff92879fa352e3786f6ccfd4c78edaaaab7a902d9 7.51kB / 7.51kB 0.0s
=> => sha256:9b1fd34c30b75e7edb20c2fd09a9862697f302ef9ae357e521ef3c84d5534e3f 12.58MB / 64.11MB 8.6s
=> [internal] load build context                                0.0s
=> => transferring context: 2.17kB                                0.0s
```

III.2 Jak skonfigurować Kubernetes i Horizontal Pod Autoscaling

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.
3. S. Fleming, *Kubernetes Handbook*, 2018

Artykuły i studia

1. Dębowski L., *ZASTOSOWANIE KOMPUTERÓW W NAUCE I TECHNICIE* 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]
8. <https://www.computerworld.pl/news/Czym-jest-Kubernetes,433339.html> [06-05-2023].

SPIS TABEL

Tabela 1. Lorem ipsum dolor sit amet..... **Błąd! Nie zdefiniowano zakładki.**

Tabela 2. Lorem ipsum dolor sit amet, consectetur adipisicing elit**Błąd!** **Nie zdefiniowano zakładki.**

SPIS RYSUNKÓW

Rysunek 1. Lorem ipsum dolor sit amet Lorem ipsum dolor sit amet**Błąd!** **Nie zdefiniowano zakładki.**

Rysunek 2. Lorem ipsum dolor sit amet, consectetur**Błąd! Nie zdefiniowano zakładki.**

STRESZCZENIE

SUMMARY