Uniwersytet Gdański
Wydział Matematyki, Fizyki i Informatyki
Instytut Informatyki

Implementacja sklepu internetowego opartego na Kubernetes

Dominika Jelińska

Projekt z przedmiotu technologie chmurowe na kierunku informatyka profil praktyczny na Uniwersytecie Gdańskim.

Gdańsk

26 czerwca 2024

Spis treści

1 Opis projektu

Projekt jest aplikacja internetowa, która umożliwia użytkownikom przegladanie, wyszukiwanie i zakup produktów online. Aplikacja obsługuje zarówno klientów, którzy moga przegladać produkty i dodawać je do koszyka, jak i zalogowanych użytkowników, którzy moga dokonywać zakupów, zarzadzać swoimi zamówieniami oraz zarzadzać swoim profilem.

Projekt wykorzystuje kontenery Dockerowe do izolacji aplikacji oraz zarzadzania jej środowiskiem uruchomieniowym. Kubernetes zapewnia automatyzacje procesów wdrażania, aktualizacji oraz skalowania aplikacji.

1.1 Opis architektury

Architektura aplikacji opartej na Kubernetes składa sie z mikroserwisów działajacych w klastrze. Główne komponenty architektury obejmuja:

1. **Frontend**: Serwis odpowiedzialny za interakcje z użytkownikami poprzez przegladarke internetowa. Serwis korzysta z Keycloak do autoryzacji użytkowników.
2. **Backend**: API oraz logika biznesowa aplikacji, obsługujaca żadania frontendu i komunikujaca sie z baza danych.
3. **Baza danych**: Baza MongoDB przechowujaca informacje o produktach, zamówieniach oraz autoryzowanych użytkownikach.
4. **Keycloak**: Usługa do zarzadzania tożsamościami i dostepem do funkcjonalności aplikacji. Keycloak zapewnia bezpieczne uwierzytelnianie i autoryzacje użytkowników.

Architektura opiera sie na mikroserwisach, co umożliwia niezależne skalowanie poszczególnych komponentów oraz elastyczność w zarzadzaniu zasobami aplikacji.

Każdy z komponentów jest uruchamiany w osobnych kontenerach Dockerowych, co zapewnia izolacje, niezależność i elastyczność w zarzadzaniu zasobami. Kubernetes zarzadza klastrami tych kontenerów, umożliwiajac automatyczne skalowanie za pomoca replik.

1.2 Opis infrastruktury

Aplikacja działa w środowisku Kubernetes, uruchomionym lokalnie za pomoca Minikube. Minikube umożliwia łatwe emulowanie klastra Kubernetes na pojedynczej maszynie, co jest idealnym rozwiazaniem do testowania i rozwoju aplikacji przed jej wdrożeniem na środowisko produkcyjne.

W klastrze Kubernetes zaimplementowane sa trzy główne komponenty: frontend obsługiwany przez serwis Frontend-Service, backend przez Backend-Service, oraz baza danych MongoDB zarzadzana przez serwis Mongo-Service. Frontend-Service udostepnia interfejs użytkownika, komunikujacy sie z backendem przez Backend-Service, który obsługuje logike biznesowa aplikacji. MongoDB zapewnia przechowywanie danych o użytkownikach, produktach i zamówieniach.

Konfiguracja sieci w klastrze Kubernetes oraz odpowiednie przypisanie zasobów, takich jak limit wykorzystywanej pamieci RAM sa dostosowane do potrzeb każdego z komponentów aplikacji.

1.3 Opis komponentów aplikacji

Poniżej przedstawiono szczegółowy opis poszczególnych komponentów:

1. **Frontend** aplikacji jest uruchamiany jako aplikacja internetowa w kontenerze Dockerowym. Komunikuje sie z backendem za pomoca interfejsu API. Dzieki skonfigurowanemu deploymentowi i serwisowi w Kubernetes, frontend jest łatwo skalowalny i dostepny dla użytkowników. Frontend integruje sie również z usługa Keycloak dla autoryzacji użytkowników. Konfiguracja frontendu, taka jak adresy URL backendu i Keycloak sa przekazywane przez ConfigMap w Kubernetes.

2. **Keycloak** jest usługa uruchamiana jako deployment w kontenerze Dockerowym. Dzieki skonfigurowanemu serwisowi, Keycloak obsługuje żadania uwierzytelniania od frontendu i backendu. Konfiguracja Keycloak, taka jak ustawienia klienta aplikacji, jest przekazywana przez ConfigMap.

3. **MongoDB** jest wykorzystywana jako baza danych NoSQL dla produktów, zamówień i profili użytkowników w aplikacji. Jest uruchomiona jako deployment w kontenerze Dockerowym. Poprzez skonfigurowany serwis w Kubernetes, umożliwia backendowi dostep do przechowywanych danych.

4. **Backend** aplikacji jest ruchamiany jako mikroserwis w kontenerze Dockerowym. Odpowiada za logike biznesowa i komunikacje z MongoDB oraz frontendem przez interfejs API. Dzieki deploymentowi i serwisowi w Kubernetes, backend zapewnia dostepność danych i może być skalowalny. Backend integruje sie również z Keycloak dla bezpiecznej autoryzacji użytkowników. Informacje konfiguracyjne, takie jak adresy URL MongoDB i ustawienia integracji z Keycloak, sa przekazywane przez ConfigMap.

1.4 Konfiguracja i zarzadzanie

Każdy komponent aplikacji jest uruchamiany jako kontener Dockerowy, co ułatwia ich przenoszenie miedzy środowiskami deweloperskimi a produkcyjnymi.

Ustawienia konfiguracyjne, takie jak adresy URL do serwisów backendowych, parametry połaczenia z baza danych oraz konfiguracja Keycloak, sa przekazywane do kontenerów za pomoca ConfigMap w Kubernetes. Dzieki temu zmiany w konfiguracji moga być łatwo wprowadzane i zarzadzane bez konieczności modyfikowania obrazów kontenerów.

1.5 Zarzadzanie błedami

Do monitorowania aplikacji można wykorzystać narzedzie dostepne w Kubernetes, takie jak Kubernetes Dashboard.

1.6 Skalowalność

Zarówno frontend, jak i backend aplikacji sa uruchamiane w Kubernetes przy użyciu replik, dzieki czemu można łatwo zwiekszać lub zmniejszać liczbe instancji aplikacji. Skalować można poprzez modyfikacje liczby replik bezpośrednio w pliku konfiguracyjnym lub za pomoca polecenia `kubectl`.

Obecnie MongoDB oraz Keycloak sa skonfigurowane do działania z pojedyncza instancja. To podejście minimalizuje skomplikowanie dostepu i zarzadzania danymi w systemie.

1.7 Wymagania dotyczace zasobów

Kontener frontendu aplikacji został skonfigurowany z określonymi zasobami pamieci RAM. Konfiguracja obejmuje ustawienia requests i limits.

- **requests**: Określa minimalne wymagane zasoby, które kontener potrzebuje do poprawnego działania. W przypadku frontendu wynosi to 256 MB pamieci RAM.
- **limits**: Określa maksymalna ilość pamieci RAM, jaka kontener może zużyć. Dla frontendu ustalono limit na 512 MB pamieci RAM.

1.8 Architektura sieciowa

Aplikacja wykorzystuje różne typy serwisów do zapewnienia komunikacji miedzy komponentami oraz dostepu dla użytkowników zewnetrznych.

- **Serwisy typu ClusterIP**: Wewnetrzne komponenty aplikacji komunikuja sie ze soba za pomoca serwisów typu ClusterIP. Każdy serwis posiada stały adres IP wewnatrz klastra, co umożliwia bezpieczna komunikacje miedzy pods.
- **Serwis typu NodePort dla frontendu**: Frontend aplikacji jest wystawiony na zewnatrz klastra za pomoca serwisu typu NodePort. Jest to sposób umożliwiajacy dostep do aplikacji z zewnatrz poprzez publiczny adres IP wezła klastra i wybrany port.

Literatura

[1] Dokumentacja Kubernetes `https://kubernetes.io/pl/docs/concepts/overview/`