Uniwersytet Gdański
Wydział Matematyki, Fizyki i Informatyki
Instytut Informatyki

Słownik międzysłowniańskiego języka

Maria Koren

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

Gdańsk

26 czerwca 2024

Spis treści

1 Opis projektu

Projekt reprezentujący słownik języka międzysłowiańskiego powstał z myślą o ludziach, chcących się nauczyć powyższego języka. Na moment stworzenia aplikacji znaleziony został tylko słownik języka międzysłowiańskiego [3]. Ale ten słownik jest zwykłym słownikiem i nie nadaje się do uczenia języka. Dlatego w aplikacji oprócz przeglądania słów można je zapisać sobie i wrócić do listy swoich zapisanych słów. Administrator może dodawać nowe słowa, usuwać istniejące słowa, przeglądać i usuwać słowa zapisane przez wszystkich użytkowników. Ponieważ aplikacja jest innowacją, zostawiona możliwość dodawania opinii przez każdego użytkownika Z racji tego, że czasami użytkownicy mogą dawać niewłaściwe komentarze administrator może je usuwać.

1.1 Opis architektury

Kubernetes to przenośna, rozszerzalna platforma typu open source do zarządzania kontenerowymi obciążeniami i usługami, która ułatwia zarówno deklaratywną konfigurację, jak i automatyzację. Posiada duży, szybko rozwijający się ekosystem. Usługi, wsparcie i narzędzia Kubernetes są powszechnie dostępne.

Aplikacja w kubernetesie składa się z następujących elementów:

- Kubernetes Cluster: Centralny element zarządzania kontenerami, składający się z jednego lub więcej węzłów (nodes). Każdy węzeł może być maszyną wirtualną lub fizyczną, zarządzaną przez Kubernetes.
- Pods: Najmniejsza jednostka przetwarzania w Kubernetes, która może zawierać jeden lub więcej kontenerów. Aplikacja będzie uruchomiona w wielu podach.
- Services: Abstrakcja sieciowa w Kubernetes, która definiuje logiczny zestaw podów i politykę dostępu do nich. Usługi zapewniają stały punkt dostępu do zmieniających się dynamicznie zestawów podów.
- Deployments: Służą do deklaratywnego zarządzania podami i replikami aplikacji. Umożliwiają wdrażanie nowych wersji aplikacji i automatyczne skalowanie.
- ConfigMaps i Secrets: Mechanizmy do zarządzania konfiguracją aplikacji i danymi wrażliwymi, takimi jak hasła i klucze API.

Opisywana aplikacja składa się z takich mikroserwisów jak Frontend, Backend, Keycloak, Bazy Danych.

Aplikacja działa na jednym węzle. Każdy ze wcześniej wymienionych komponentów ma zdefionowane service i deployment

1.2 Opis infrastruktury

Projekt działa w środowisku lokalnym, co oznacza, że wszystkie komponenty Kubernetes są uruchomione na wirtualnych środowiskach, takim jak Kind (Kubernetes IN Docker).

Aplikacja jest konteneryzowana za pomocą Dockera, co umożliwia jej uruchomienie w izolowanych środowiskach. Definicje wdrożeń (deployments) oraz usług (services) są zapisane w plikach YAML, co umożliwia ich łatwe zarządzanie i wersjonowanie.

1.3 Opis komponentów aplikacji

Aplikacja składa się z następujących komponentów:

- Backend
- Frontend
- Keycloak
- Bazy Danych

1.3.1 Backend

Backend został napisany w języku JavaScript. Uruchamiany jako kontener Docker. Komunikuje się z bazą danych MongoDB, frontende, aplikacji. Ma zabiespieczone endpointy, weryfikujące role: zalogowany użytkownik, zalogowany jako admin.

1.3.2 Frontend

Jest to aplikacja kliencka, napisana wykorzystując React. Uruchamiany jako kontener Docker. Komunikuję się z backendem dla dostarczania informacji. Przy logowaniu przekierowuje na Keycloak.

1.3.3 Keycloak

System zarządzania tożsamością i dostępem, korzystający z bazy danych Postgres. Uruchamiany w Kubernetes jako osobny pod

1.3.4 Bazy Danych

Zostały użyte 2 bazy danych: MongoDB oraz Postgres. Baza MongoDB służy do przechowywania danych, dotyczących aplikacji: słowa, opinii, słowa zapisane dla użytkownika. Baza Postgres używana dla przechowywania danych z Keycloaka.

1.4 Konfiguracja i zarządzanie

W celu konfiguracji i zarzącania zostały napisane pliki backend.yaml, frontend.yaml, mongo.yaml, postgres.yaml - dla tych komponentów. Dla keycloaka to są pliki keycloak-claim0-persistentvolumeclaim.yaml, keycloak-deployment.yaml, keycloak-service.yaml.

Bazą plików dla keycloka służył plik docker-compose.yaml, z którego za pomocą narzędzia kompose zrobiony został plik dla kubernetes.

Kompose to narzędzie open-source, które umożliwia łatwe przekształcanie definicji aplikacji w Docker Compose do konfiguracji dla Kubernetes. [1]

Dla keycloka pomimo service i deployment został zdefiniowany PersistentVolumeClaim. PersistentVolumeClaim (PVC) to obiekt w Kubernetes, który reprezentuje żądanie zasobów pamięci masowej przez użytkownika. W systemie Kubernetes PersistentVolume (PV) to zasób pamięci masowej zarządzany przez klaster, natomiast PersistentVolumeClaim to sposób, w jaki użytkownicy lub aplikacje mogą żądać tych zasobów. [2]

Oprócz Keycloaka persistance volume claim został zdefiniowany dla obu baz danych - MongoDB oraz Postgres. Aby zabiespieczyć trwałość danych.

1.5 Zarządzanie błędami

Zarządzanie błędami zapewnia stabilność i niprzerwaną pracę systemu, W Kubernetesie, jako zaawansowanej platformie zarządzania kontenerami, są pewne wpudowane mechanizmy automatycznego restartowania podów, które mogą przestać działać z różnych powodów.

Automatyczne restartowanie podów w Kubernetes jest kluczowe aby utrzymać wysoką dostępność aplikacji. Gdy pod przestaje działać, na przykład z powodu błędu aplikacyjnego lub awarii węzła, Kubernetes wykrywa ten stan i podejmuje odpowiednie działania. Proces restartowania może obejmować przenoszenie poda na inny węzeł lub próbę uruchomienia go ponownie na tym samym węźle.

1.6 Skalowalność

Jedną z ważniejszych korzyści, jakie zapewnia Kubernetes, jest możliwość łatwego skalowania aplikacji. Skalowanie aplikacji pozwala uruchomić ją na dowolnej liczbie serwerów, aby obsłużyć rosnącą liczbę użytkowników.[2]

W powyższym projekcie użyłam skalowania automatycznego, polegającego na wykorzystaniu mechanizmu nazywanego HPA (Horizontal Pod Autoscaler), który monitoruje obciążenie podów i automatycznie dostosowuje ich liczbę do aktualnego zapotrzebowania.

Podany mechanizm opiera się na monitorowaniu obciążenia cpu - jak jest powyżej 50% zwiększ ilość replik. Repliki są skalowane od 1 do 10.

1.7 Wymagania dotyczące zasobów

Ograniczenie zasobów w Kubernetes pozwala kontrolować, ile CPU i pamięci jest dostępnych dla poda. Ograniczenia te mogą być ustawione na poziomie kontenera. [4]. Limity tę powodują, że komponent aplikacji nie będzie mógł zużywać więcej ustawionych limitów pamięci lub cpu.

W podanej aplikacji limity ustamione nastepująco:

- Backend: 1Gi pamięci, 1 jednostka CPU
- Frontend: 1Gi pamięci, 0.5 jednostki CPU

- MongoDB: 1Gi pamięci, 1 jednostka CPU
- Postgres: 1Gi pamięci, 1 jednostka CPU

1.8 Architektura sieciowa

Jednym z głównych komponentów w Kubernetes jest sieć. Sieć w kubernetes pomaga w komunikacji między komponentami, aby się widziały nawzajem. W opisanej aplikacji trzema głównymi komponentami, które muszą być dostępne na zewnątrz, są frontend, backend oraz keycloak. Dla każdego z tych komponentów w klastrze Kubernetes konfiguruje się osobny LoadBalancer. LoadBalancer jest mechanizmem, który umożliwia równoważenie obciążenia i udostępnianie usług na zewnątrz klastra, zapewniając im publiczny dostęp.

Dzięki temu każdy z wymienionych komponentów (frontend, backend, keycloak) może być dostępny i komunikować się z użytkownikami lub innymi systemami spoza klastra Kubernetes, co jest kluczowe w architekturze mikroserwisowej i rozproszonych systemach aplikacyjnych.

Literatura

[1] *Kompose documentation.*

[2] *Oktawave.*

[3] *Słownik języka międzysłowniańskiego.*

[4] Mateusz Miotk, *Techologie chmurowe.*