u **Uniwersytet Gdański**
Wydział Matematyki, Fizyki i Informatyki
Instytut Informatyki

Klaster kubernetes na podstawie aplikacji ToDo

Mateusz Karnicki

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

Gdańsk

25 czerwca 2024

Spis treści

1 Opis projektu

Celem projektu było zbudowanie aplikacji pozwalajacej użytkownikowi ustalać i śledzić zadania do wykonania w przejrzysty sposób z poziomu przegladarki internetowej. W celu uzyskania jak najwiekszej ilości użytkowników aplikacja jest szybka i zawiera tylko najważniejsze funkcje, sprawiajac że można ja uruchomić niezależnie od tego jak dobry komputer posiada użytkownik.

Każda osoba, która chce korzystać z serwisu musi założyć własne konto, którego zawartość dostepna jest niezależnie od tego, z którego urzadzenia korzysta.

1.1 Opis architektury

Klaster Kubernetes składa sie z wielu współpracujacych elementów zapewniajacych poprawne działanie aplikacji. Wszystkie mikroserwisy z wyjatkiem bazy danych posiadaja własny zoptymalizowany plik Dockerfile. Na podstawie tych plików zbudowano obrazy, które zostały umieszczona w serwisie DockerHub, pozwalajac na uruchomienie wiekszości podserwisów w klastrze Kubernetes na dowolnym urzadzeniu. Każda cześć aplikacji posiada pliki Service oraz Deployment, pierwszy z nich ustalaja parametry usług, drugi natomiast zarzadza tworzeniem i skalowaniem podów oraz aktualizacjami aplikacji. *Pod* w Kubernetes to najmniejsza jednostka zarzadzania. Składa sie z jednego lub wiecej kontenerów, które współdziela pamieć, sieć i dysk.

Baza danych została zaimplementowa przy użyciu *PersistentVolume*. W Kubernetes, *PersistentVolume* (PV) reprezentuje trwała przestrzeń dyskowa do wykorzystania przez klaster. Jest to sposób na przechowywanie danych, które przetrwaja nawet po zakończeniu działania poda. [1]

1.2 Opis infrastruktury

W tym projekcie Kubernetes samodzielnie zajmuje sie ustalaniem komunikacji pomiedzy konkretnymi usługami oraz ich wieloma instancjami. Pozwala to na bezkonfliktowe rozszerzanie i skalowanie aplikacji.

W klastrze typ usług został ustawiony jako LoadBalancer, którego głównym zadaniem jest równomierne rozkładanie ruchu sieciowego miedzy *podami*, zapewniajac, że mikroserwisy pozostaja responsywne i dostepne, nawet przy wielu aktywnych instancjach. Dodatkowo wystawia je na świat zewnetrzny pozwalajac na podłaczenie sie do nich np. z przegladarki.

1.3 Opis komponentów aplikacji

- Frontend - Interfejs graficzny użytkownika do obsługi naszej aplikacji, zawiera panel logowania oraz strone główna wyświetlajaca liste zadań użytkownika oraz prosty formularz do dodawania kolejnych. Napisany został w jezyku JavaScript przy użyciu biblioteki React.
- Backend - serwer aplikacji odpowiedzialny za obsługe żadań wysyłanych przez frontend oraz zwracanie odpowiednich danych dla użytkowników. Ta cześć aplikacji została napisana w jezyku JavaScript przy użyciu frameworku Express, który umożliwia tworzenie i obsługe usług serwerowych.
- PostgreSQL - jeden z najpopularniejszych otwartych systemów zarzadzania relacyjnymi bazami danych. Dane przechowywane sa w postaci tabel składajacych sie z rekordów posiadajacych pewne atrybuty. Do obsługi zapytań wykorzystujemy kwerendy napisane w SQL. Baza przechowuje zarówno dane użytkowników jak i pliki konfiguracyjne.
- Keycloak - Usługa uwierzytelniajaca użytkowników wewnatrz naszej aplikacji. Weryfikuje oraz rozsyła tokeny, za pomoca których weryfikuje czy użytkownik jest zalogowany oraz czy posiada odpowiednie uprawnienia do zasobu. Keycloak przechowuje wszystkie konta użytkowników oraz odpowiedzialny jest za logowanie i rejestracje do aplikacji.

1.4 Konfiguracja i zarzadzanie

Pliki konfiguracyjne klastra sa przechowywane w oddzielnym folderze w formacie YAML. Każda usługa posiada osobne pliki, co przyczynia sie do uporzadkowania projektu i ułatwia szybkie odnalezienie odpowiednich opcji konfiguracyjnych. Na ich podstawie tworzone sa pody z określona konfiguracja, zawierajaca informacje takie jak obraz kontenera, zmienne środowiskowe specyficzne dla danej usługi oraz liczba instancji do utworzenia. Taki sposób konfiguracji umożliwia szybkie wdrażanie zmian oraz dostosowywanie parametrów bez konieczności ingerencji w cała strukture aplikacji.

1.5 Zarzadzanie błedami

Zarzadzanie błedami odbywa sie za pomoca wbudowanych mechanizmów Kubernetesa. Mikroserwisy maja ustalona polityke reagowania na awarie ustawiona na wartość *Always*, co oznacza, że dany pod bedzie ciagle ponownie uruchamiany aż do momentu poprawnego uruchomienia.

1.6 Skalowalność

Skalowalność odnosi sie do zdolności zwiekszania zasobów przypisanych do danego komponentu w odpowiedzi na wzrost obciażenia. W projekcie wykorzystano Horizontal Pod Autoscaler, mechanizm Kubernetesa monitorujacy zużycie zasobów. Gdy zużycie pamieci osiaga ustalona wartość - w tym projekcie ustalona na 80% - HPA automatycznie tworzy nowe instancje danego *podu*.

Dla zapewnienia efektywności i wysokiej dostepności usług, ustalono limit 5 instancji dla frontendu, backendu oraz keycloaka.

1.7 Wymagania dotyczace zasobów

W celu zabezpieczenia mikroserwisów przed zachłannościa innych na pamieć oraz użycie procesora, wszystkie pody maja ustalony poziom zasobów, którego nie wolno im przekroczyć. Granice te zostały ustalone poprzez podwojenie zaobserwowanych wartości zużycia zasobów przez *pody*.

Usługa\Zasoby	Maksymalne zużycie procesora	Maksymalne zużycie pamieci
Frontend	200 mili-rdzeni	256MiB
Backend	100 mili-rdzeni	256MiB
PostgreSQL	100 mili-rdzeni	256MiB
Keycloak	500 mili-rdzeni	2GiB

1.8 Architektura sieciowa

Architektura sieciowa została zdefiniowana w plikach konfiguracyjnych poprzez ustalenie portów, na których działaja usługi oraz przez ustalenie które usługi łacza sie z soba. Usługi takie jak frontend czy keycloak maja ustalony typ LoadBalancer, który nie tylko zapewnia równomierne rozłożenie zużycia zasobów, ale również wystawia porty na zewnatrz klastra. Użytkownicy sa w stanie połaczyć sie z aplikacja w celu zalogowania, rejestracji, badź przegladania własnych zadań. Wykorzystanie nazw usług w konfiguracji zamiast poszczególnych adresów daje klastrowi Kubernetes możliwość samodzielnego ustalania adresów wewnetrznych oraz przekierowywania ruchu sieciowego pomiedzy kilkoma instancjami tych samych usług.

Literatura

[1] mgr. Mateusz Miotk, *Technologie chmurowe*, 2024.