Uniwersytet Gdański
Wydział Matematyki, Fizyki i Informatyki
Instytut Informatyki

AuthMaster

Marta Kurzych

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

Gdańsk

25 czerwca 2024

Spis treści

1 Opis projektu

Pewna firma - "Innovatech Inc. zauważyła potrzebe kompleksowego rozwiazania do zarzadzania użytkownikami i ich autoryzacja w ramach swoich aplikacji webowych. Potrzebowali bazy, która integruje nowoczesne technologie, aby zapewnić niezawodność, skalowalność i bezpieczeństwo. Tak powstał projekt "AuthMaster nowatorski system autoryzacji oparty na połaczeniu React, Flask, MongoDB, Keycloak oraz Kubernetes. Dodatkowo, do przechowywania konfiguracji Keycloak używany jest PostgreSQL. Dzieki tej architekturze firma może sprawnie zarzadzać użytkownikami, bezpiecznie przechowywać dane oraz łatwo skalować aplikacje zgodnie z rosnacymi potrzebami.

1.1 Opis architektury

Architektura aplikacji "AuthMaster"jest oparta na Kubernetes[2] - klastrowym systemie zarzadzania kontenerami, który zapewnia skalowalność i wysoka dostepność.

React stanowi interfejs użytkownika, który komunikuje sie z API Flask. Flask jest serwerem backendowym obsługujacym logike aplikacji i zarzadzajacym sesjami użytkowników. MongoDB to baza danych NoSQL, która przechowuje informacje o użytkownikach. Keycloak[1] działa jako serwer tożsamości, zarzadzajac autoryzacja i uwierzytelnianiem, a jego konfiguracja jest przechowywana w bazie danych PostgreSQL. Kubernetes jest platforma do zarzadzania wdrażaniem kontenerów Docker[3], zapewniajaca skalowanie i monitoring.

Kubernetes to open-source'owa platforma do zarzadzania kontenerami, która automatyzuje wiele procesów zwiazanych z wdrażaniem, skalowaniem i operacjami aplikacji kontenerowych. Kubernetes jest wyjatkowo elastyczny i umożliwia łatwe zarzadzanie dużymi aplikacjami w różnych środowiskach.

1.2 Opis infrastruktury

Aplikacja "AuthMaster"działa w środowisku chmurowym, które zapewnia elastyczność i łatwość zarzadzania zasobami.

Kubernetes zarzadza kontenerami Docker, które uruchamiaja poszczególne komponenty aplikacji. Docker umożliwia tworzenie obrazów kontenerów dla React, Flask, MongoDB, Keycloak i PostgreSQL.

Kubernetes zapewnia orkiestracje kontenerów, automatyczne uruchamianie, zatrzymywanie i zarzadzanie kontenerami w klastrze. System ten jest samonaprawiajacy, co oznacza, że automatycznie ponownie uruchamia uszkodzone kontenery, wymienia je, jeśli wezeł w klastrze zawiedzie, oraz usuwa kontenery, które nie odpowiadaja na zdefiniowane testy.

1.3 Opis komponentów aplikacji

React tworzy dynamiczny interfejs użytkownika, który komunikuje sie z API Flask poprzez zapytania HTTP. Flask obsługuje rejestracje użytkowników, logowanie, zarzadzanie sesjami oraz komunikacje z MongoDB i Keycloak. MongoDB przechowuje dane

o użytkownikach, takie jak hasła i inne informacje profilowe. Keycloak zarzadza autoryzacja i uwierzytelnianiem, zapewniajac mechanizmy logowania i zarzadzania sesjami, a jego konfiguracja jest przechowywana w bazie danych PostgreSQL. Kubernetes orkiestruje wdrażanie i zarzadzanie kontenerami, zapewniajac wysoka dostepność i skalowalność.

1.4 Konfiguracja i zarzadzanie

Konfiguracja i zarzadzanie aplikacja odbywa sie na poziomie klastra Kubernetes. Używane sa pliki YAML do definiowania podów, usług, replikasetów i deploymentów. Kluczowe elementy to ConfigMaps i Secrets, które przechowuja konfiguracje i dane wrażliwe w klastrze.

ConfigMapy sa używane do przechowywania zmiennych środowiskowych dla backendu. ConfigMap zawiera konfiguracje, które nie sa wrażliwe i moga być łatwo zarzadzane oraz aktualizowane.

Secrets sa używane do przechowywania danych wrażliwych, takich jak hasło do bazy danych PostgresSQL. Sa one zaszyfrowane w klastrze Kubernetes i dostepne tylko dla autoryzowanych podów.

1.5 Zarzadzanie błedami

Zarzadzanie błedami obejmuje kilka kluczowych aspektów, które sa realizowane przy użyciu narzedzi oferowanych przez Kubernetes.

W systemie Kubernetes, zarzadzanie błedami rozpoczyna sie od zastosowania mechanizmów Liveness i Readiness Probes, które sa wykorzystywane do monitorowania stanu aplikacji.

W przypadku Keycloak, Liveness Probe sprawdza, czy aplikacja działa prawidłowo i nie zawiesiła sie. Dzieki temu, jeśli Keycloak przestanie odpowiadać, Kubernetes automatycznie ponownie uruchomi kontener, co pomaga utrzymać aplikacje w stanie operacyjnym.

Readiness Probe sprawdza, czy aplikacja jest gotowa do obsługi żadań. Jeśli aplikacja nie przejdzie tego testu, Kubernetes przestanie kierować do niej ruch. Dzieki Readiness Probe, Kubernetes może dynamicznie zarzadzać ruchem do podów, zapewniajac, że tylko zdrowe instancje sa dostepne dla użytkowników.

W razie poważnych awarii, Kubernetes automatycznie restartuje kontenery, które przestały działać prawidłowo. Ponadto, można ustawić odpowiednie zasady retry (powtórzenia) w kodzie aplikacji oraz na poziomie warstwy sieciowej, aby zapewnić odporność na chwilowe problemy.

1.6 Skalowalność

Skalowalność aplikacji jest zapewniona dzieki Kubernetes, który umożliwia reczne skalowanie przez ustawienie liczby replik podów na podstawie obserwacji obciażenia. Kubernetes zapewnia elastyczność, umożliwiajac dostosowywanie zasobów w zależności od potrzeb firmy. Chociaż nie jest używany w obecnej konfiguracji, Kubernetes oferuje również możliwość automatycznego skalowania liczby podów w zależności od obciażenia za pomoca Horizontal Pod Autoscaler (HPA).

1.7 Wymagania dotyczace zasobów

Każdy komponent aplikacji ma określone wymagania dotyczace zasobów. Dla React i Flask ustawiono limity zasobów na 200Mi pamieci RAM i 500m CPU. Oznacza to, że interfejs użytkownika React oraz API Flask sa zoptymalizowane pod katem niskiego zużycia zasobów, co pomaga w efektywnym zarzadzaniu obciażeniem w klastrze Kubernetes.

MongoDB, jako baza danych, ma wyższe wymagania dotyczace zasobów, a dla trwałego przechowywania danych używane sa Persistent Volumes (PV) o pojemności 2Gi oraz Persistent Volume Claims (PVC) o pojemności 1Gi. PostgreSQL, używany do przechowywania konfiguracji Keycloak, ma podobne wymagania dotyczace zasobów: PV o pojemności 2Gi i PVC o pojemności 1Gi. Limity zasobów dla Keycloak i PostgreSQL sa ustawione odpowiednio, aby zapewnić płynne działanie systemu.

1.8 Architektura sieciowa

Architektura sieciowa aplikacji "AuthMaster"opiera sie na usługach Kubernetes, które zarzadzaja ruchem wewnetrznym i zewnetrznym. Wszystkie pody aplikacji maja zdefiniowane serwisy (service.yml), które umożliwiaja komunikacje miedzy nimi oraz z zewnetrznymi systemami. Service działa jako stabilny punkt dostepu do aplikacji działajacych w podach, niezależnie od tego, na którym weźle w klastrze działaja te pody.

Dla komponentów takich jak Keycloak, frontend (React) i backend (Flask), dodatkowo zdefiniowane sa Ingressy, które zarzadzaja dostepem z zewnatrz do aplikacji. Ingress Controller umożliwia zarzadzanie ruchem HTTP/HTTPS, zapewniajac równoważenie obciażenia oraz terminacje SSL.

Ingress zapewnia, że ruch HTTP/HTTPS jest przekierowywany do odpowiednich usług w klastrze na podstawie zdefiniowanych reguł. W tym przypadku, wszystkie żadania skierowane do keycloak sa przekierowywane do usługi Keycloak działajacej na porcie 8080.

Taka konfiguracja umożliwia efektywne zarzadzanie ruchem, zapewnia bezpieczeństwo dzieki TLS oraz ułatwia skalowanie aplikacji poprzez dodawanie kolejnych replik podów bez potrzeby zmiany konfiguracji sieciowej.

Literatura

[1] The Keycloak Authors, *Dokumentacja keycloaka*, 2024.

[2] The Kubernetes Authors, *Dokumentacja kubernetesa*, 2024.

[3] Docker Inc., *Dokumentacja dockera*, 2024.

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