Uniwersytet Gdański
Wydział Matematyki, Fizyki i Informatyki
Instytut Informatyki

Konfiguracja klastra Kubernetes dla aplikacji Games-Store

Mateusz Domino

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

Gdańsk

26 czerwca 2024

Spis treści

1 Opis projektu

Firma Games-Store Corporation, specjalizująca się w handlu grami, zleciła stworzenie nowej aplikacji webowej do sprzedaży gier różnych kategorii online. Aplikacja została stworzona przez pododział firmy. Serwis ma umożliwiać logowanie i rejestrację użytkowników, zakup oraz zwroty gier, oraz obsługiwać różne metody płatności. Również ma być dostępny panel administratora do dodawania nowych gier, dostępny tylko dla pracowników.

1.1 Opis architektury

Architektura została zaimplementowana w środowisku Kubernetes, co umożliwia w razie potrzeby łatwe skalowanie i wdrożania nowych zmian za pomocą obrazów Docker. Węzły klastra Kubernetes:

- Express-server - API do przetwarzania zapytań z frontendu oraz zwracania danych z bazy danych.
- Frontend - Aplikacja frontend napisana w frameworku Next.js.
- Mongo-db - Baza danych MongoDB.
- Nginx-proxy - Serwer Reverse-Proxy Nginx, przekazuje zapytania do aplikacji.

1.2 Opis infrastruktury

1.2.1 Środowisko uruchomieniowe

Aplikacja wykorzystuje środowisko Docker w następującej wersji silnika:

- version 24.0.6
- build ed223b

Wersja wykorzystywanego Kubernetes:

- Client Version: v1.27.2
- Kustomize Version: v5.0.1
- Server Version: v1.27.2

1.2.2 Zasoby obliczeniowe

Każda replika węzła w klastrze ma zdefiniowane minimalne i maksymalne zasoby obliczeniowe. Repliki są automatycznie tworzone i skalowane w zależności od aktualnego obciążenia, wykorzystując optymalnie zasoby.

1.2.3 Pamięć masowa

Dla bazy danych MongoDB, jest przydzielony wolumen PersistentValueClaim, o wielkości 5Gi na dane.

1.3 Opis komponentów aplikacji

Aplikacja składa się się z 4 komponentów, każdy został zdefiniowany w oddzielnym pliku manifest dla Kubernetesa jako osobne pody:

1.3.1 Baza danych MongoDB

Służy do przechowywania informacji o użytkownikach, grach, powiadomieniach, wiadomościach pomocy technicznej, transakcjach oraz zwrotów gier. Do tego użyto bazę danych MongoDB z odpowiednimi kolekcjami dla danych.

1.3.2 API Express node.js

Api zostało stworzone w node.js używając framework Express. Służy do obsługi zapytań wysyłanych przez Frontend, obsługuje przetwarzanie transakcji, użytkowników, przekazuje informacje z bazy danych oraz waliduje tokeny logowania. Również zawiera operacje dostępne tylko dla pracowników sklepu, między innymi: dodawanie nowych gier, przetwarzanie zwrotów gier, odpowiadanie na wiadomości pomocy technicznej oraz zarządzanie użytkownikami. Do tego jest wykorzystywany zmodyfikowany obraz node zawierający aplikację.

1.3.3 Frontend next.js

Frontend został zaimplemenowany w frameworku Next.js, operatym na React. Użytkownik za jego pomocą może wykonywać wszystkie operacje w aplikacji, takie jak logowanie, rejestracja, zakup gier i ich zwracanie, wysyłanie wiadomości do pomocy, sprawdzanie powiadomień, zmiana informacji profilu takich jak na przykład nazwa użytkownika i hasło. Zawiera również panel administratora. Komunikacja z API polega na wysyłaniu zapytań HTTP. Został w tym celu wykorzystany zmodyfikowany obraz node zawierający skompilowaną aplikację.

1.3.4 Serwer Reverse Proxy Nginx

Jego zadaniem jest przekierowywanie zapytań do poszczególnych komponentów, działa na obrazie nginx.

1.4 Konfiguracja i zarządzanie

Konfiguracja aplikacji jest zdefiniowana w plikach manifest Kubernetes, które określają serwisy i deploymenty klastra. Zawarte są w nich poprzez zmienne środowiskowe, między innymi: secret dla szyfrowania/odszyfrowania hasła oraz adres bazy danych. Zawiera deployment 4 głównych komponentów, poprzez obrazy Docker. Do komunikacji służy serwer Nginx ustawiony jako LoadBalancer, do przekazywania zapytań zarówno do Api, jak i do frontendu.

1.5 Zarządzanie błędami

Api jest przystosowane do wysłania odpowiednich komunikatów w razie, błędnych zapytań wraz z kodami błędów.

W przypadku wystąpienia błądu w jednym z podów, Kubernetes automatycznie go usuwa, po czym tworzy nowy, identyczny. W ten sposób w środowisku produkcyjnym, zapewnione jest ciągłe działanie aplikacji.

1.6 Skalowalność

Skalowanie aplikacji jest realizowane przez Horizontal Pod Autoscaler (HPA). Przy obecnej konfiguracji monitorowane jest Api, frontend oraz serwer Nginx. W razie zwiększonego obciązania któregoś z tych elementów, HPA automatycznie zwiększy ilość jego replik, a w razie spadku proporcjonalnie zmniejszy. Aktualnie HPA skonfigurowane jest na minimalnie po 3 repliki dla każdego z wymienionego elementu, a maksymalnie na 10. Średnie użycie HPA jest ustawione na 50 dla CPU i 70 dla pamięci RAM.

1.7 Wymagania dotyczące zasobów

Wymagania zasobów dla każdego komponenetu aplikacji, z przyjętym oczekiwanym czasem odpowiedzi poniżej 200 ms.

1.7.1 Baza danych

- Wymagania minimalne
  - CPU: 1
  - RAM: 2Gi
- Wymagania maksymalne
  - CPU: 3
  - RAM: 4Gi

1.7.2 Api

- Wymagania minimalne
  - CPU: 0.5
  - RAM: 256Mi
- Wymagania maksymalne
  - CPU: 1
  - RAM: 512Mi

1.7.3 Frontend

- Wymagania minimalne
  - CPU: 0.1
  - RAM: 128Mi
- Wymagania maksymalne
  - CPU: 0.5
  - RAM: 512Mi

1.7.4 Serwer Reverse Proxy

- Wymagania minimalne
  - CPU: 0.1
  - RAM: 64Mi
- Wymagania maksymalne
  - CPU: 0.5
  - RAM: 256Mi

1.8 Architektura sieciowa

1.8.1 Baza danych

Serwis typu NodePort, nasłuchujący na porcie 30332:27017. Ten serwis pozwala do dostęp do bazy z zewnątrz klastra na porcie 30332 na każdym węźle.

1.8.2 Api

Serwis typu ClusterIP, zapewnia dostęp do Api Express. Nasłuchuje na porcie 8000:8000. Nie można się dostać do niego z zewnątrz klastra.

1.8.3 Frontend

Serwis typu ClusterIP, zapewnia dostęp do aplikacji frontend. Nasłuchuje na porcie 3000:3000. Nie można się dostać do niego z zewnątrz klastra.

1.8.4 Serwer Reverse Proxy

Serwer Nginx to jedyny serwis typu LoadBalancer, przez co jako jedyny ma przydzielony zewnętrzny adres IP. Serwer nasłuchuje na porcie 80 i w zaleznosci od ścieżki URL, przekazuje zapytania do frontendu bądź Api.

Literatura

[1] Express Docs, *Express documentation https://expressjs.com/en/api.html.*

[2] Kubernetes Docs, *Kubernetes documentation https://kubernetes.io/docs/home.*

[3] MongoDB Docs, *Mongodb documentation https://www.mongodb.com/docs.*

[4] Next.js Docs, *Next.js documentation https://nextjs.org/docs.*

[5] JWT, *Introduction to json web tokens https://jwt.io/introduction.*

[6] Wikipedia, *bcrypt hashing function https://en.wikipedia.org/wiki/bcrypt.*

[7] _____, *Cross-origin resource sharing mechanism https://en.wikipedia.org/wiki/cross-origin_resource_sharing.*