Politechnika Śląska

Dokumentacja projektowa

2023/2024

Zarządzanie systemami informatycznymi

Konteneryzacja i implementacja kubernetes: utowrzenie klastra w usłudze Microsoft Azure

Kierunek: Informatyka

Członkowie zespołu:
Dawid Gala
Hubert Bojda

Gliwice, 2023/2024

Spis treści

1 Wprowadzenie

1.1 Role w projekcie

Hubert Bojda - zajęcie się komunikacją aplikacji z bazą danych, konteneryzacja i testy.
Dawid Gala - stworzenie plików, definiujących aplikacje w klastrze i utworzenie zdalnego środowiska w usłudze Microsoft Azure.

1.2 Cel projektu

Naszym celem jest stworzenie skonteneryzowanej aplikacji, którą następnie chcemy wdrożyć w środowisku chmurowym. Ponieważ wcześniej realizowaliśmy już podobne projekty, tym razem chcemy poszerzyć naszą wiedzę i umiejętności, wdrażając naszą aplikację w Kubernetesie. Chcemy nauczyć się zarówno podstawowych, jak i zaawansowanych aspektów tej technologii oraz zrozumieć, jak można efektywnie zarządzać skonteneryzowanymi aplikacjami w skalowalnym i elastycznym środowisku klastrowym.

2 Założenia projektowe

Plan działań obejmuje kilka kluczowych kroków:

- Stworzenie aplikacji: Zaczniemy od napisania aplikacji, której poszczególne komponenty zostaną skonteneryzowane za pomocą narzędzi takich jak Docker.
- Tworzenie obrazów Docker: Dla każdego komponentu aplikacji utworzymy obrazy Docker, które będą zawierały wszystkie niezbędne zależności i konfiguracje.
- Konfiguracja Kubernetes: Zainstalujemy i skonfigurujemy klaster Kubernetes, który pozwoli nam zarządzać wdrożeniem naszych kontenerów. Na początku to wszystko będzie testowane na środowisku lokalnym - za pomocą Minikube. Poznamy podstawowe elementy, takie jak Pody, Deployments, Services.
- Przygotujemy pliki YAML, które zdefiniują zasoby Kubernetes, takie jak Deployment i Service. Następnie wdrożymy naszą aplikację w klastrze Kubernetes.
- Integracja z chmurą: Gdy już wszystko lokalnie będzie działać tak jak sie spodziewamy, skonfigurujemy nasz klaster Kubernetes w środowisku chmurowym Azure Kubernetes Service (AKS), aby zyskać dodatkowe możliwości skalowania i zarządzania infrastrukturą.

2.1 Założenia techniczne i nietechniczne

Założenia techniczne :

- Miejsce w przestrzeni dyskowej na obrazy lokalne i wirtualizacje minikube.
- Subskrypcja w usłudze Microsoft Azure.
- Globalna baza danych - MongoDb udziela darmowej wersji swojej bazy w celach naukowych.

Założenia nietechniczne:

- Termin: Ukończenie do 5 czerwca 2024r, aby zaprezentować na wykładzie.
- Budżet jak najniższy, najlepiej zerowy, subskrypcja studencka usługi Microsoft Azure w tym bardzo nam pomoże.

2.2 Stos technologiczny

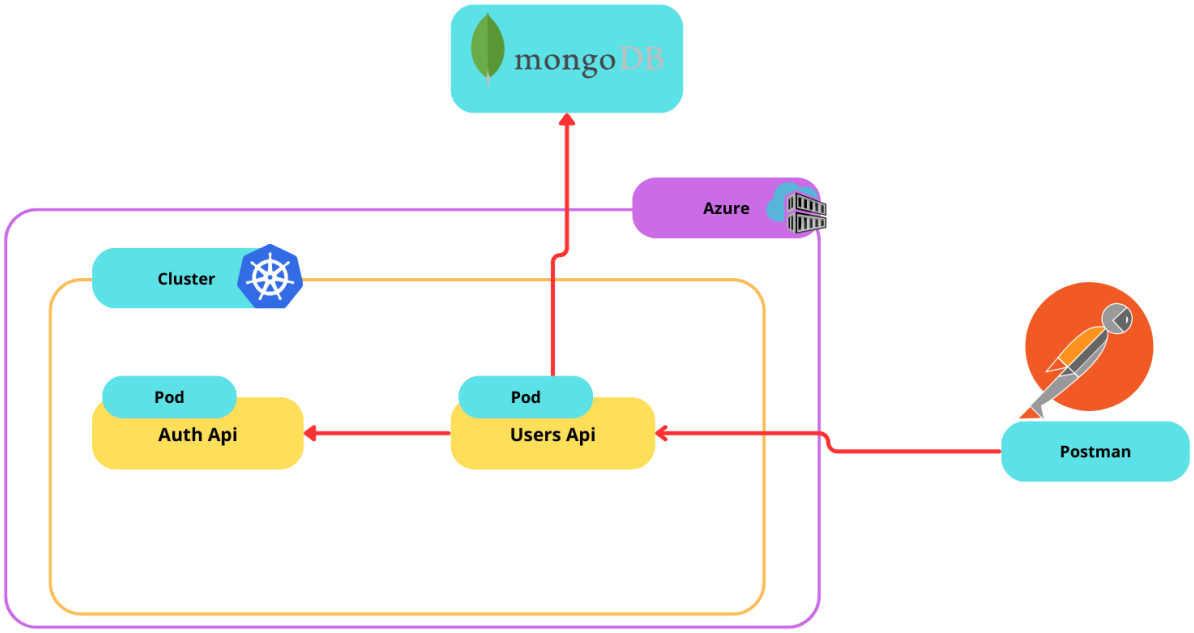
Narzędzia i systemy informatyczne związne z projektem:

- **Docker**:
  - Docker to platforma do tworzenia, wdrażania i uruchamiania aplikacji w kontenerach. Umożliwia izolację aplikacji oraz jej zależności w środowisku, co zapewnia bezpieczeństwo środowiska.
- **Docker Hub**:
  - Docker Hub to publiczna rejestracja obrazów kontenerów, gdzie użytkownicy mogą przechowywać, udostępniać i pobierać obrazy Dockera. Umożliwia łatwe publikowanie i dystrybucję kontenerów zarówno w środowiskach deweloperskich, jak i produkcyjnych. W naszym przypadku pozwoli kubernetesowi pobierać najnowsze obrazy, na podstawie których utworzy kontenery w klastrach.
- **Kubernetes**:
  - Kubernetes to system typu Open Source, stworzony przez Google do orkiestracji kontenerów, który automatyzuje wdrażanie, skalowanie i zarządzanie aplikacjami kontenerowymi. Umożliwia zarządzanie klastrami kontenerów i zapewnia ich niezawodność oraz skalowalność.
- **Kubectl**:
  - kubectl to narzędzie wiersza poleceń do interakcji z klastrami Kubernetes. Umożliwia zarządzanie aplikacjami, inspekcję zasobów, wdrażanie i rozwiązywanie problemów w środowisku Kubernetes.
- **Minikube**:
  - Minikube to narzędzie, które pozwala na lokalne uruchamianie klastra Kubernetes na komputerze deweloperskim. Umożliwia testowanie i rozwijanie aplikacji w Kubernetes bez potrzeby używania pełnowymiarowego klastra.
- **Azure**:
  - Microsoft Azure to chmurowa platforma obliczeniowa oferująca różne usługi, w tym wirtualne maszyny, bazy danych, usługi kontenerowe (Azure Kubernetes Service - AKS) oraz narzędzia do zarządzania i monitorowania aplikacji.

- **Visual Studio Code (VS Code)**:
  - Visual Studio Code to lekki edytor kodu źródłowego, który wspiera wiele języków programowania i narzędzi developerskich. Oferuje funkcje takie jak debugowanie, integracja z Git, oraz liczne rozszerzenia, które ułatwią nam pracę.
- **Postman**:
  - Postman to narzędzie do testowania API, które umożliwia wysyłanie zapytań HTTP i analizowanie odpowiedzi. Jest używane do tworzenia i testowania interfejsów API, automatyzacji testów oraz współpracy w zespole przy dokumentowaniu API.
- **MongoDB**:
  - MongoDB to nierelacyjna baza danych typu NoSQL, która przechowuje dane w formacie dokumentów JSON. Jest skalowalna, elastyczna i przede wszystkim umożliwia nam darmową opcje do testowania rozwiązania.

2.3 Oczekiwane rezultaty projektu

Naszymi oczekiwania obejmują skuteczne skonteneryzowanie aplikacji, wdrożenie jej w klastrze Kubernetes i jej dostępnośc w chmurze Azure. Jednocześnie zależy nam, aby zdolność zapisu do bazy danych MongoDb nie uległa zmianie. Poprzez konteneryzację aplikacji, ta będzie spakowana w kontener Dockerowy i wypchnięta do rejestru Docker Hub i gotowa do wdrożenia w klastrze Kubernetes. Po pomyślnym wdrożeniu, aplikacja będzie dostępna w chmurze Azure, co zapewni elastyczność i skalowalność jej działania. Współpraca z bazą danych MongoDB umożliwi aplikacji przechowywanie i odczytywanie danych logowania użykownika.



Rysunek 1: Prosty schemat, ukazujący nasze założenia

Aby spełnić wymagania, należy odpowiednio zabezpieczyć obie aplikacje oraz zapewnić komunikację tylko między nimi. Aplikacja "Users Api"będzie wystawiona publicznie, co pozwoli użytkownikom na rejestrację i logowanie. Po poprawnym uwierzytelnieniu, "Users Api"będzie komunikować się z "Auth Api", który będzie odpowiedzialny za generowanie tokenów uwierzytelniających. W ten sposób dane uwierzytelniające będą przechowywane w ukrytej części architektury, niedostępnej dla świata zewnętrznego. W celu sprawdzenia poprawności działania aplikacji, możemy użyć narzędzia takiego jak Postman, które pozwoli na wykonywanie żądań HTTP i sprawdzanie odpowiedzi serwera. Dzięki temu będziemy mogli przetestować poprawność działania naszej aplikacji, jednocześnie zachowując jej bezpieczeństwo. Dzięki Kubernetes, nawet w przypadku awarii naszej aplikacji - którą celowo uwzględniliśmy w kodzie w celach demonstracyjnych - nasza usługa pozostanie nadal dostępna. Jest to efekt orkiestracji i zarządzania kontenerami przez Kubernetes.

Dzięki elastycznemu skalowaniu i automatycznemu przywracaniu usług, Kubernetes zapewnia nieprzerwane działanie naszej aplikacji, nawet w obliczu incydentów. To sprawia, że nasza aplikacja jest bardziej niezawodna i odporna na potencjalne zakłócenia, co przekłada się na lepsze doświadczenie użytkownika i większą pewność siebie w zakresie działania naszej usługi.

3 Realizacja projektu

3.0.1 Docker

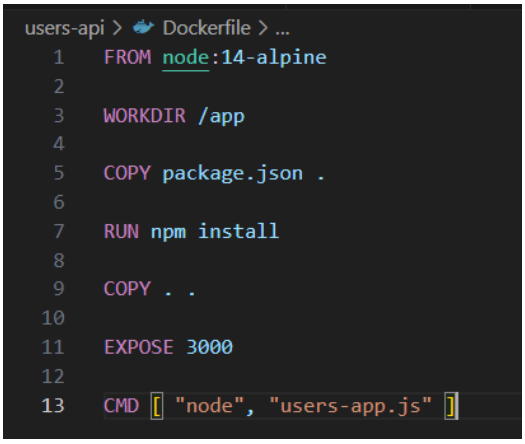
Na początku zajmijmy się konteneryzacją naszych obu aplikacji. Najpierw zajmijmy się aplikacją "Users Api". Przed konteneryzajcą warto spojrzeć w kod i zastanowić się co tak naprawde potrzebujemy. Przy konteneryzacji aplikacji warto spojrzeć na port, na którym ona nasłuchuje. Jednocześnie, dla naszej wygody i ewentualnej łatwej zmianie bazy danych. Wyodrębnimy link do zmiennej środowiskowej.

```
mongoose.connect(
  process.env.MONGODB_CONNECTION_URI,
  { useNewUrlParser: true },
  (err) => {
    if (err) {
      console.log('COULD NOT CONNECT TO MONGODB!');
    } else {
      app.listen(3000);
    }
  }
);
```

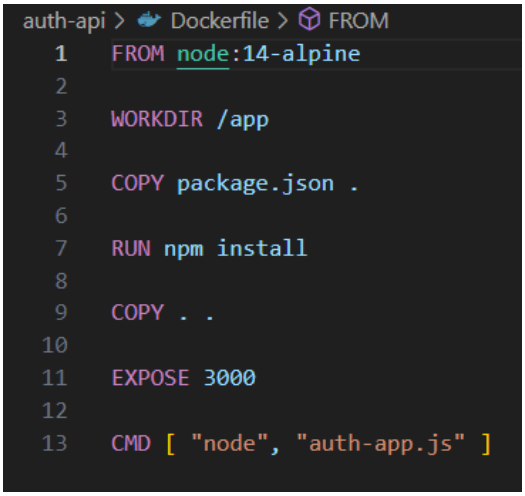
Rysunek 2: Tutaj umieść podpis pod zdjęciem

Teraz, musimy pamiętać o przypisaniu tej zmiennej, przy uruchomianiu kontenera, co pokażemy później.

Poniżej znajduje się przedstawienie pliku Dockerfile dla utworzenia obrazów obu aplikacji:



Rysunek 3: Dockerfile dla aplikacji "Users Api"



Rysunek 4: Dockerfile dla aplikacji "Auth Api"

————————T-u lekko opisac co dana komenda robi———————— Teraz, gdy już mamy nasz schemat do utworzenia obrazów naszych kontenerów, utworzymy plik "docker-compose.yaml", który nam określi schemat utworzenia kontenerów:
Jak widać na poniższym zdjęciu, do zmiennej środowiskowej:
MONGODB_CONNECTION_URI przypisaliśmy link do naszej bazy.

Rysunek 5: DockerFile, dzieki któremu docker utworzy kontenery

Uruchommy konsolę w ścieżcę, w której znajduje się nasz plik "docker-compose.yaml"i wywołamy komende:
`docker-compose up -d` Komenda ta tworzy kontenery na podstawie zawartego w pliku docker-compose, który znajduje się w naszej scieżce. Parametr d pozwala nam "odłączyć sie"od kontenera.

Po udanych testach za pomocą POSTMAN, możemy przejść do dalszej częsci.
Teraz zależy nam na umieszczeniu tych obrazów w repozytorium. Umożliwi to późniejsze pobieranie obrazów przez kubernetes, gdy będzie tworzył swoje pody. W każdym katalogu, który zawiera plik dockerfile wywołajmy komende, która zbuduje nam obraz z określoną nazwą:
`docker build . -t baitazar/kubernetes-azure-users`
Następnie:
`docker push baitazar/kubernetes-azure-users`
`docker build . -t baitazar/kubernetes-azure-auth`
Następnie:
`docker push baitazar/kubernetes-azure-auth`
Spowoduje to utworzenie obrazu o okreslonej nazwie, która musi być taka sama jak wcześniej utworzone repozytorium. Następnie za pomocą komendy `docker push` wypychamy ten obraz do repozytorium.

4 Wnioski

- *Spostrzeżenia*
- *Osiągnięcia*
- *Potencjał rozwoju*