Integracja mongo DB z mongo-express przy pomocy Docker'a oraz Docker-compose

Marian Dorosz

23.03.2022

Spis treści

Spis rysunków

1 Pobieranie obrazów z DockerHub'a

Pierwszym krokiem jest pobranie obrazów kontenerów z DockerHub'a. Aby to zrobić należy skorzystać z polecenia **docker pull nazwa obrazu**. W przypadku obu obrazów należy skorzystać z tego samego polecenia, więc poniżej pojawi się przykład tylko dla obrazu mongo DB.

```
mario@DESKTOP-2JFCR0O:/$ docker pull mongo
Using default tag: latest
latest: Pulling from library/mongo
4d32b49e2995: Pulling fs layer
4d32b49e2995: Pull complete
26a89ffa9c8e: Pull complete
c6a26a1adeb9: Pull complete
0f6c4ca429ae: Pull complete
87cd51bf7ebc: Pull complete
68750eb424ec: Pull complete
008900bad1d7: Pull complete
e33eed19868f: Pull complete
e7bc3cbfdaeb: Pull complete
358eefa21051: Pull complete
Digest: sha256:ad947856db716ddd0b9cc525e341c77208ed8dafcb4a6ad23f9b3a
ddd7a4f71c
Status: Downloaded newer image for mongo:latest
docker.io/library/mongo:latest
mario@DESKTOP-2JFCR0O:/$ docker images
REPOSITORY        TAG       IMAGE ID       CREATED        SIZE
jenkins/jenkins   latest    40a0f9c2bc25   30 hours ago   460MB
mongo             latest    798d1656acba   3 days ago     698MB
mario@DESKTOP-2JFCR0O:/$
```

Rysunek 1: Pobieranie obrazu mongo DB

Wykorzystanie ww. polecenia - **docker pull mongo** spowoduje, że na nasz komputer zostanie pobrany obraz kontenera aplikacji mongo DB.

2 Konstruowanie pliku docker-compose.yaml

2.1 Do czego służą pliki typu docker-compose.yaml?

Piki **docker-compose.yaml** usprawniają tworzenie kontenerów, gdyż powstały do tworzenia dużej ilości kontenerów małym nakładem pracy. Dodatkowo infrastruktura aplikacji utworzona w tych plikach staje się szablo-

nem, przez co taki plik możemy w zasadzie uruchomić na każdym komputerze, który posiada Docker'a i dostęp do internetu. Gdy posiadamy gotowy plik, wystarczy skorzystać z polecenia - **docker-compose -f (nazwa pliku).yaml up**. Polecenie to spowoduje, że Docker utworzy kontenery zgodnie z konfiguracją zawartą w pliku.

2.2 Analiza pliku Compose, potrzebnego do zintegrowania mongo DB z mongo-express

```
version: '3'                                # wersja pliku docker-compose
services:                                   # ponizej zostana wymienione kontenery, ktore maja zostac uruchomione
  mongodb:                                  # nazwa kontenera
    image: mongo                            # nazwa obrazu
    ports:                                  # port zewnętrzny(sieciowy):port wewnętrzny(port kontenera)
    - 6000:6000
    environment:                            # zmienne srodowiskowe
    - MONGO_INITDB_ROOT_USERNAME=admin      # login root'a do bazy
    - MONGO_INITDB_ROOT_PASSWORD=admin      # haslo do konta root'a

  mongo-express:                             # nazwa kontenera
    depends_on:
      - mongodb
    image: mongo-express                     # nazwa obrazu
    ports:                                   # port zewnętrzny(sieciowy):port wewnętrzny(port kontenera)
    - 8080:8081
    environment:                            # zmienne srodowiskowe
      - ME_CONFIG_MONGODB_ADMINUSERNAME=admin  # login do konta root
      - ME_CONFIG_MONGODB_ADMINPASSWORD=admin  # haslo do konta root
      - ME_CONFIG_MONGODB_SERVER=mongodb       # nazwa kontenera z baza mongoDB
```

Rysunek 2: Kod potrzebnego pliku

Wersja pliku docker-compose Parametr **version:'3'** jest informacją, jakiej wersji jest plik Compose, który tworzymy. Wersja 3 została stworzona aby była kompatybilna w docker-compose oraz docker swarm.

Nazwa kontenera Pierwszym parametrem kontenera, który ustawia się w pliku Compose jest jego nazwa. Aby utworzyć kontener o zadanej przez nas nazwie wystarczy wpisać w pliku **nazwa_kontenera:**. Po uzupełnieniu nazwy można konfigurować dodatkowe parametry kontenera.

Nazwa obrazu Parametrem **image: nazwa obrazu** ustalamy jaki obraz będzie wykorzystany do utworzenia kontenera.

Port zewnętrzny(sieciowy):port wewnętrzny(port kontenera) Parametrem **Ports: port zewnętrzny:port wewnętrzny** ustawiamy porty zewnętrzne (TCP) oraz wewnętrzne (porty na których nasłuchuje kontener).

Mapowanie portów powstało, aby umożliwić łączenie się z aplikacją uruchomioną w kontenerze z poziomu komputera.

Zmienne środowiskowe Parametr **environment** pozwala na uruchomienie kontenera z dodatkowymi opcjami konfiguracyjnymi, które nie są elementami samego docker'a, lecz są dodatkowymi konfigurowalnymi parametrami aplikacji. Przykładowo: w przypadku mongo DB jest to skonfigurowanie loginu i hasła root'a dla systemu bazodanowego.

Depends on Ten parametr spowoduje, że kontener z aplikacją mongo-express nie powstanie dopóki nie zostanie uruchomiony kontener mongo DB.

2.3 Uruchomienie pliku Compose

Pliki typu Compose uruchamia się wcześniej wspomnianym:
docker-compose -f (nazwa pliku).yaml up.

2.3.1 Wynik uruchomienia pliku

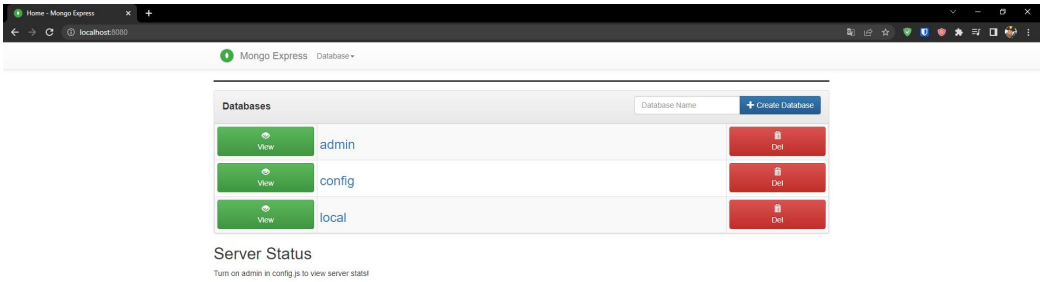
```
mario@DESKTOP-2JFCR0O:/mnt/d/Programowanie/Docker/projekt_mongodb+mongo-express$ docker-compose -f mongo-docker-compose.yaml up
Creating network "projekt_mongodbmongo-express_default" with the default driver
Creating projekt_mongodbmongo-express_mongodb_1 ... done
Creating projekt_mongodbmongo-express_mongo-express_1 ... done
Attaching to projekt_mongodbmongo-express_mongodb_1, projekt_mongodbmongo-express_mongo-express_1
mongodb_1        | about to fork child process, waiting until server is ready for connections.
mongodb_1        | forked process: 30
mongodb_1        |
mongodb_1        | {"t":{"$date":"2022-03-23T22:20:08.593+00:00"},"s":"I",  "c":"CONTROL",  "id":20698,   "ctx":"-","msg":"***** SE
RVER RESTARTED *****"}
mongodb_1        | {"t":{"$date":"2022-03-23T22:20:08.605+00:00"},"s":"I",  "c":"NETWORK",  "id":4915701, "ctx":"main","msg":"Initi
alized wire specification","attr":{"spec":{"incomingExternalClient":{"minWireVersion":0,"maxWireVersion":13},"incomingInternalClien
t":{"minWireVersion":0,"maxWireVersion":13},"outgoing":{"minWireVersion":0,"maxWireVersion":13},"isInternalClient":true}}}
mongodb_1        | {"t":{"$date":"2022-03-23T22:20:08.605+00:00"},"s":"I",  "c":"CONTROL",  "id":23285,   "ctx":"main","msg":"Autom
atically disabling TLS 1.0, to force-enable TLS 1.0 specify --sslDisabledProtocols 'none'"}
mongodb_1        | {"t":{"$date":"2022-03-23T22:20:08.607+00:00"},"s":"W",  "c":"ASIO",     "id":22601,   "ctx":"main","msg":"No Tr
ansportLayer configured during NetworkInterface startup"}
mongodb_1        | {"t":{"$date":"2022-03-23T22:20:08.607+00:00"},"s":"I",  "c":"NETWORK",  "id":4648601, "ctx":"main","msg":"Impli
cit TCP FastOpen unavailable. If TCP FastOpen is required, set tcpFastOpenServer, tcpFastOpenClient, and tcpFastOpenQueueSize."}
mongodb_1        | {"t":{"$date":"2022-03-23T22:20:08.608+00:00"},"s":"W",  "c":"ASIO",     "id":22601,   "ctx":"main","msg":"No Tr
ansportLayer configured during NetworkInterface startup"}
mongodb_1        | {"t":{"$date":"2022-03-23T22:20:08.608+00:00"},"s":"I",  "c":"REPL",     "id":5123008, "ctx":"main","msg":"Succe
ssfully registered PrimaryOnlyService","attr":{"service":"TenantMigrationDonorService","ns":"config.tenantMigrationDonors"}}
mongodb_1        | {"t":{"$date":"2022-03-23T22:20:08.609+00:00"},"s":"I",  "c":"REPL",     "id":5123008, "ctx":"main","msg":"Succe
ssfully registered PrimaryOnlyService","attr":{"service":"TenantMigrationRecipientService","ns":"config.tenantMigrationRecipients"}
}
mongodb_1        | {"t":{"$date":"2022-03-23T22:20:08.609+00:00"},"s":"I",  "c":"CONTROL",  "id":5945603, "ctx":"main","msg":"Multi
 threading initialized"}
mongodb_1        | {"t":{"$date":"2022-03-23T22:20:08.609+00:00"},"s":"I",  "c":"CONTROL",  "id":4615611, "ctx":"initandlisten","ms
g":"MongoDB starting","attr":{"pid":30,"port":27017,"dbPath":"/data/db","architecture":"64-bit","host":"9958960a9d7b"}}
mongodb_1        | {"t":{"$date":"2022-03-23T22:20:08.609+00:00"},"s":"I",  "c":"CONTROL",  "id":23403,   "ctx":"initandlisten","ms
g":"Build Info","attr":{"buildInfo":{"version":"5.0.6","gitVersion":"212a8dbb47f07427dae194a9c75baec1d81d9259","openSSLVersion":"Op
enSSL 1.1.1f  31 Mar 2020","modules":[],"allocator":"tcmalloc","environment":{"distmod":"ubuntu2004","distarch":"x86_64","target_ar
ch":"x86_64"}}}}
```

Rysunek 3: Wynik uruchomienia stworzonego pliku docker-compose

Uruchomienie utworzonego wcześniej pliku docker-compose spowoduje utworzenie dwóch kontenerów, przechowujących aplikacje mongo DB oraz

mongo-express. Z obiema aplikacjami można połączyć się z poziomu komputera, na którym je uruchomiono.

2.4 Test działania



Rysunek 4: Działająca aplikacja mongo-express

Jak widać na powyższym zrzucie, z poziomu komputera można zalogować się do aplikacji mongo-express, która łączy się z mongo DB, a więc integracja obu systemów powiodła się.

2.5 Cofanie efektów pliku docker-compose

```
mario@DESKTOP-2JFCR0O:/mnt/d/Programowanie/Docker/projekt_mongodb+mongo-express$ docker-compose -f mongo-docker-compose.yaml down
Stopping projekt_mongodbmongo-express_mongo-express_1 ... done
Stopping projekt_mongodbmongo-express_mongodb_1       ... done
Removing projekt_mongodbmongo-express_mongo-express_1 ... done
Removing projekt_mongodbmongo-express_mongodb_1       ... done
Removing network projekt_mongodbmongo-express_default
mario@DESKTOP-2JFCR0O:/mnt/d/Programowanie/Docker/projekt_mongodb+mongo-express$
```

Rysunek 5: Cofanie efektów działania pliku docker-compose.yaml

Jak widać na powyższym zrzucie cofnięcie efektów działania pliku docker-compose jest bardzo proste. Wystarczy skorzystać z polecenia **docker-compose -f (nazwa pliku).yaml down**. Po wpisaniu polecenia docker automatycznie usunie wszystkie kontenery oraz sieć docker net, którą utworzył wcześniej.