OPWO – Docker (2)

Podczas zajęć należy wykonać kolejno ćwiczenia opisane w poniższych instrukcjach krok-po-kroku. Ćwiczenia dotyczące narzędzia **Docker Compose** należy wykonać korzystając z zainstalowanego lokalnie **Dockera**. **Ćwiczenia dotyczące Docker Swarm/Stack** należy wykonać z **interfejsu terminala(i)** udostępnianego na platformie **Play with Docker** (kopiowanie z/do terminala można zrealizować kombinacją `Ctrl+Insert` i `Shift+Insert`). Następnie, należy **zweryfikować z listą kontrolną** zrozumienie podanych na niej zagadnień (i w razie potrzeby wrócić do powiązanych ćwiczeń).

Ćwiczenie 1 (Docker Compose)

1. Stwórz plik `docker-compose.yml (w wersji '3')` uruchamiający aplikację złożoną z 3 usług:
   - `db` (obraz `postgres`, udostępnianie dla pozostałych usług domyślnego portu `5432`; używane zmienne środowiskowe: `POSTGRES_DB` – wartość test, `POSTGRES_USER` – wartość `student`, `POSTGRES_PASSWORD` – wartość `student`)
   - `rg` (obraz `registry.gitlab.com/protective-h2020-eu/meta-alert-prioritisation/ranking-generator:1.8.0`; mapowanie portów – port `8008` na hoście Dockera odpowiada portowi `8008` w kontenerze; używane zmienne środowiskowe: `WAIT_FOR_HOST_AND_PORT` – wartość `db:5432`, `SPRING_DATASOURCE_URL` – wartość `jdbc:postgresql://db:5432/test`, `SPRING_DATASOURCE_USERNAME` – wartość `student`, `SPRING_DATASOURCE_PASSWORD` – wartość `student`; zależy od usług `db` i `ri`)
   - `ri` (obraz `registry.gitlab.com/protective-h2020-eu/meta-alert-prioritisation/rule-inducer:1.6.0`; mapowanie portów – port `8009` na hoście Dockera odpowiada portowi `8009` w kontenerze; używane zmienne środowiskowe: `WAIT_FOR_HOST_AND_PORT` – wartość `db:5432`, `SPRING_DATASOURCE_URL` – wartość `jdbc:postgresql://db:5432/test`, `SPRING_DATASOURCE_USERNAME` – wartość `student`, `SPRING_DATASOURCE_PASSWORD` – wartość `student`; zależy od usługi `db`)
2. Uruchom aplikację odpowiednią wersją polecenia `docker-compose` i poczekaj aż uruchomią się wszystkie usługi; w czasie uruchamiania obserwuj komunikaty wypisywane przez poszczególne usługi
3. Sprawdź poprawność działania aplikacji, w szczególności dostęp do bazy danych, otwierając w przeglądarce następujące adresy url:
   - http://localhost:8008/meta-data?user_id=protective
   - http://localhost:8009/get-metadata?id=1
4. Zatrzymaj aplikację poleceniem Ctrl+C i poczekaj aż zatrzymają się wszystkie usługi składowe.

Ćwiczenie 2 (**Docker Swarm + Stack**):

- Docker Orchestration Hands-on Lab: https://training.play-with-docker.com/orchestration-hol/ (konfiguracja swarma, dołączenie managera i 2 workerów, ręczne uruchomienie usługi na swarmie, ręczne skalowanie uruchomionej usługi, drenowanie node'a z obciążenia celem eleganckiego przeniesienia zadań na inny węzeł swarm'a, usuwanie usługi, wyjście z trybu swarm)
- Swarm Mode Introduction for IT Pros: https://training.play-with-docker.com/ops-s1-swarm-intro/ (stworzenie swarma, dołączenie managera i 1 workera, uruchomienie aplikacji (stack usług) na podstawie gotowego pliku w formacie Docker Compose, skalowanie liczby zadań (replik) w uruchomionej usłudze, polecenia do obsługi stacka, wizualizacja swarma)

Lista kontrolna zagadnień:

1. składnia pliku `docker-compose.yml` dla Docker Compose
2. tworzenie swarm'a (powstaje tzw. `manager node`)

3. dołączanie do swarm'a (powstaje tzw. `worker node`)
4. ręczne uruchamianie usług na swarm'ie
5. tworzenie pliku `docker-compose.yml` dla Docker Stack
6. uruchamianie stacka (zbioru usług) na swarm'ie z użyciem pliku `docker-compose.yml`
7. skalowanie usług (zmiana liczby replik), wyłączanie węzła swarm'a z eksploatacji (drenowanie węzła)
8. podgląd struktury swarm'a, działających usług oraz ich zadań (replik)
9. zakończenie stacka
10. wyjście z trybu swarm