DOCKER COMPOSE

Hello

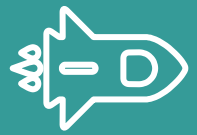
Łukasz Pieczonka

DevOps @ LU-MEDIA

Agenda

- Docker - uzupełnienie
- Czym jest Docker Compose?
- Budowanie, Uruchamianie oraz Zarządzanie.
- Ćwiczymy

DOCKER UZUPEŁNIENIE



DOCKER

Docker jest obecnie jednym z kilku narzędzi, które uruchamiają kontenery, zawiera narzędzia do budowania uruchamiania i zarządzania.

Inne narzędzia o podobnej funkcjonalności to np. Containerd lub CRI-O

Główne zalety - kontenery działają niezależnie od siebie, budowa zależności np. Aplikacja + Baza danych

DOCKER

- Do tej pory łączyliśmy się do działającego kontenera głównie przy pomocy flagi -p (--publish 8080:80), mapowanie portu hosta na port uruchomionego kontenera.

- W celu uzyskania dostępu do plików użyliśmy flagi -v (--volume host:kontener)

DOCKER NETWORK

Sieć Docker`a umożliwia komunikację ze swoim hostem oraz innymi kontenerami na hoście lub dowolnymi maszynami w sieci hosta lub poza nią

Przy instalacji Dockera otrzymujemy trzy typy sieci, aby je zobaczyć użyjemy polecenia: docker network ls

DOCKER Network

Podstawowe polecenia

- docker network connect
- docker network create
- docker network disconnect
- docker network inspect
- docker network ls
- docker network prune
- docker network rm

DOCKER Network

- Do tej pory łączyliśmy się do działającego kontenera głównie przy pomocy flagi -p (--publish 8080:80), mapowanie portu hosta na port uruchomionego kontenera.

- W celu uzyskania dostępu do plików użyliśmy flagi -v (--volume host:kontener)

DOCKER-COMPOSE

Docker Compose automatyzuje tworzenie i uruchamianie wielu kontenerów. Znacznie upraszcza to pracę oraz eliminuje błędy, najczęściej gdy musimy uruchomić wiele kontenerów zależnych od siebie.

Całość definiujemy w pliku YAML docker-compose.yml, w którym umieszczamy jakie kontenery mają zostać uruchomione, konfigurację oraz zależności.

DOCKER-COMPOSE

Instalacja docker-compose:

- Vagrant plugin:

  vagrant plugin install vagrant-docker-compose

- Linux Debian/Ubuntu:

  apt install docker-composer

DOCKER_COMPOSE

Konstrukcja pliku konfiguracyjnego docker-compose.yml

(format YAML https://pl.wikipedia.org/wiki/YAML)

Pierwszą linijką jest wersja formatu pliku konfiguracyjnego która powinna się zgadzać z wersją Docker`a, nie chodzi o wersję docker-compose.

DOCKER_COMPOSE

Sprawdzamy wersję:

```
vagrant@manager:/$ docker-compose -v
docker-compose version 1.25.0, build unknown
vagrant@manager:/$ docker -v
Docker version 20.10.12, build e91ed57
vagrant@manager:/$
```

Dla wersji 20.10.12 nie została jeszcze poprawiona dokumentacja https://docs.docker.com/compose/compose-file/, zachowana jest kompatybilność, formatować będziemy dla specyfikacji: “3.8” 19.03.0+ znaczenie to ma przy stosowaniu zaawansowanej konfiguracji.

DOCKER_COMPOSE

Przykładowy plik YML:

Docker-compose.yml - phpmyadmin + mysql

```
version: '3.1'

services:
  db:
    image: mariadb:10.3
    restart: always
    environment:
      MYSQL_ROOT_PASSWORD: infoshare123

  phpmyadmin:
    image: phpmyadmin
    restart: always
    ports:
      - 8080:80
    environment:
      - PMA_ARBITRARY=1
```

DOCKER_COMPOSE

polecenia

- docker-compose build - buduje aplikację i jej zależności
- docker-compose up - uruchamia aplikację i jej zależności
- docker-compose status - pokazuje status kontenerów i jej zależności
- docker exec - wykonuje polecenie na uruchomionym kontenerze
- docker-compose run - wykonuje polecenie uruchamiając nowy kontener
- docker attach - podłącza się do działającego kontenera
- docker-compose logs oraz docker logs - podgląd logów z kontenerów
- docker-compose stop - wyłącza kontenery
- docker-compose down - wyłącza i usuwa kontenery
- docker-compose scale - ustawia liczbę kontenerów dla usługi

DOCKER_COMPOSE

Budowanie

docker-compose build

```
version: "3.8"
services:
  app:
    build: .dir
```

DOCKER SWARM

Docker SWARM jest stosunkowo nowym kompleksowym rozwiązaniem klastrowym działającym na podobnej zasadzie co Kubernetes.

Jest to rozwiązanie wspierane przez producenta - Docker Enterprise Edition

DOCKER SWARM

Najważniejsze cechy:

- Zarządzanie klastrem zintegrowane z Docker Engine - nie potrzebujesz dodatkowego oprogramowania aby utworzyć klaster i nim zarządzać.
- Zdecentralizowana konstrukcja - oznacza to, możesz zbudować cały klaster z jednego obrazu dysku.
- Skalowanie - w każdej chwili możesz określić liczbę replik kontenera, kiedy skalujesz usługę w górę i w dół, manager klastra automatycznie doda lub usunie kontenery

DOCKER SWARM

Najważniejsze cechy:

- Monitorowanie stanu usługi - menedżer monitoruje stan klastra i uzgadnia różnice między stanem rzeczywistym a stanem pożądanym. Na przykład, jeśli skonfigurujesz usługę do uruchomienia 10 replik kontenera, a maszyna robocza obsługująca dwie z tych replik ulegnie awarii, manager tworzy dwie nowe repliki, aby zastąpić repliki które uległy awarii.
- Sieć (multi-host networking) - Możesz określić sieć (overlay network) dla swoich usług.

DOCKER SWARM

Najważniejsze cechy:

- Service discovery - menedżer klastra przypisuje każdej usłudze unikalną nazwę DNS. Możesz odpytać każdy kontener działający w klastrze poprzez nazwę z serwera DNS klastra.
- Load balancing - równoważenie obciążenia, możesz wyeksponować porty do zewnętrznego load balancera

DOCKER SWARM

Najważniejsze cechy:

- Secure by default - Każdy węzeł w roju wymusza wzajemne uwierzytelnianie i szyfrowanie TLS, aby zabezpieczyć komunikację między sobą a wszystkimi innymi węzłami. Istnieje możliwość użycia samodzielnie podpisanych certyfikatów głównych lub certyfikatów z niestandardowego głównego CA.
- Rolling updates - menedżer klastra pozwala kontrolować opóźnienie pomiędzy wdrażaniem usług do różnych zestawów węzłów. Jeśli coś pójdzie nie tak, możesz cofnąć się do poprzedniej wersji usługi.

DOCKER SWARM

docker swarm init - Inicjalizacja klastra

docker swarm join - dołączenie noda do klastra

DOCKER_COMPOSE

Uruchamianie

Zarządzanie

Ćwiczymy !!

WORKSHOP

Zadanie 1

Uruchomić serwer plików NFS

```
workshop-2$ cat docker-compose-nfs.yml
version: "3.8"
services:
  nfs:
    image: itsthenetwork/nfs-server-alpine
    container_name: nfs
    restart: unless-stopped
    privileged: true
    environment:
      - SHARED_DIRECTORY=/mnt/share
    volumes:
      - /mnt/share:/mnt/share
    ports:
      - 2049:2049
      - 2049:2049/udp
      - 111:111
      - 111:111/udp
      - 32765:32765
      - 32765:32765/udp
      - 32767:32767
      - 32767:32767/udp
    deploy:
      mode: replicated
      replicas: 1
      placement:
        constraints: [node.role == manager]
```

WORKSHOP

Zadanie 2

Uruchomić bazę danych MYSQL + PHPMYADMIN

```
volumes:
  db_data:
    driver: local
    driver_opts:
      type: nfs
      o: "addr=192.168.10.2,nfsvers=3,rw,nolock"
      device: ":/mnt/mysql"
```

MYSQL wymaga flagi montowania nolock

WORKSHOP

Zadanie 3

Uruchomić kontener z Wordpress

- Użyć bazy danych z zadania 2
- Volumes użyć z zadania 1

WORKSHOP

Zadanie 4

Backup/Recovery