

KONTENERYZACJA - DOCKER

RÓŻNICA MIĘDZY DOCKER COMPOSE I DOCKER SWARM, DOCKER + WSL2

KONTENERYZACJA

KONTENERYZACJA TO PROCES PAKOWANIA APLIKACJI I JEJ ZALEŻNOŚCI W IZOLOWANE JEDNOSTKI OPROGRAMOWANIA, ZWANE KONTENERAMI, KTÓRE MOŻNA ŁATWO PRZENOSIĆ MIĘDZY ŚRODOWISKAMI.

KONTENERYZACJA I DOCKER

DOCKER JEST JEDNYM Z NAJPOPULARNIEJSZYCH NARZĘDZI DO KONTENERYZACJI. UMOŻLIWIA TWORZENIE, ZARZĄDZANIE I URUCHAMIANIE KONTENERÓW W SPOSÓB EFEKTYWNY I POWTARZALNY.

DOCKER COMPOSE

DOCKER COMPOSE TO NARZĘDZIE DO DEFINIOWANIA I ZARZĄDZANIA WIELOKONTENEROWYMI APLIKACJAMI. POZWALA NA URUCHAMIANIE APLIKACJI ZŁOŻONYCH Z WIELU USŁUG ZDEFINIOWANYCH W PLIKU YAML.

PRZYKŁAD DOCKER COMPOSE

```
services:
  redis:
    image: redislabs/redismod
    ports:
      - '6379:6379'
  web:
    build:
      context: .
      target: builder
    # flask requires SIGINT to stop gracefully
    # (default stop signal from Compose is SIGTERM)
    stop_signal: SIGINT
    ports:
      - '8000:8000'
    volumes:
      - ./code
    depends_on:
      - redis
```

URUCHOMIENIE APLIKACJI

GDY KONFIGURACJA APLIKACJI JEST PRAWIDŁOWO OPISANA W PLIKU YAML, ABY JĄ URUCHOMIĆ, WYSTARCZY WYKORZYSTAĆ POLECENIE DOCKER COMPOSE.

URUCHOMIENIE APLIKACJI

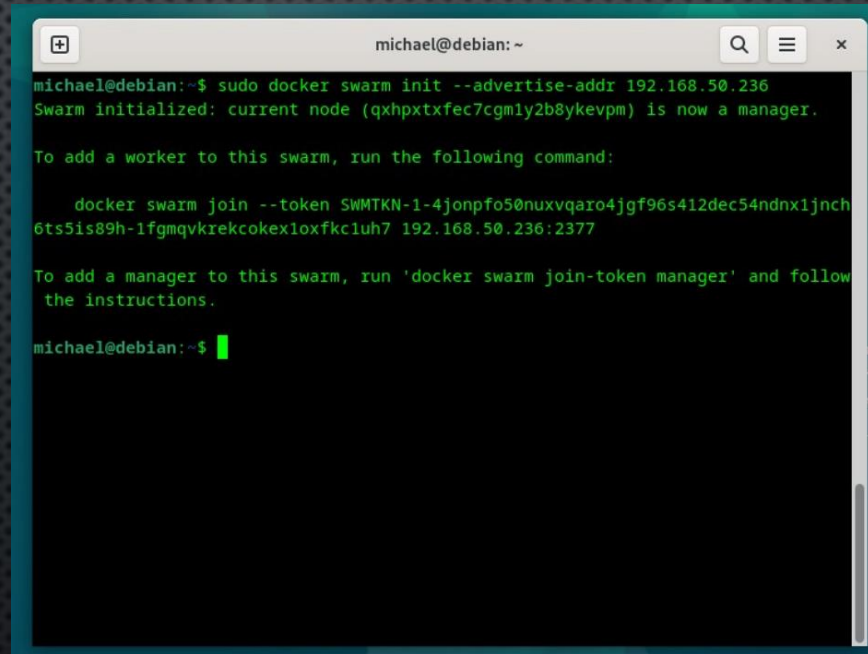
```
$ docker-compose up -d

Building web
[+] Building 0.6s (12/12) FINISHED
docker:default
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 526B
=> [internal] load .dockerignore
=> => transferring context: 2B
=> resolve image config for docker.io/docker/dockerfile:1.4
=> CACHED
docker-image://docker.io/docker/dockerfile:1.4@sha256:9ba7531bd80fb0a858632727cf7a112fbfd19b17e94c4e84ced81e24ef1a0dbc
=> [internal] load metadata for docker.io/library/python:3.10-alpine
=> [builder 1/5] FROM docker.io/library/python:3.10-alpine
=> [internal] load build context
=> => transferring context: 228B
=> CACHED [builder 2/5] WORKDIR /code
=> CACHED [builder 3/5] COPY requirements.txt /code
=> CACHED [builder 4/5] RUN --mount=type=cache,target=/root/.cache/pip pip3
install -r requirements.txt
=> CACHED [builder 5/5] COPY . /code
=> exporting to image
=> => exporting layers
=> => writing image sha256:3ed010016a47e8538f2c11b89973f7881e4ad54041f96268ac4e89927f67d883
=> => naming to docker.io/library/flask-redis-web
WARNING: Image for service web was built because it did not already exist. To rebuild
this image you must use `docker-compose build` or `docker-compose up --build`.
Creating flask-redis_redis_1 ... done
Creating flask-redis_web_1 ... done
```

DOCKER SWARM

DOCKER SWARM JEST NARZĘDZIEM DO ZARZĄDZANIA KLASTRAMI KONTENERÓW DOCKER. UMOŻLIWIA ŁATWE SKALOWANIE I ZARZĄDZANIE APLIKACJAMI KONTENEROWYMI W ŚRODOWISKU PRODUKCYJNYM.

UTWORZENIE KLASTRA MANAGERA

A terminal window titled 'michael@debian: ~' with search, menu, and close icons. It shows the command 'sudo docker swarm init --advertise-addr 192.168.50.236' being executed. The output indicates the swarm is initialized and provides a token for adding workers. It also includes instructions for adding more workers and managers.

```
michael@debian:~$ sudo docker swarm init --advertise-addr 192.168.50.236
Swarm initialized: current node (qxhpctxfec7cgmly2b8ykevpw) is now a manager.

To add a worker to this swarm, run the following command:

    docker swarm join --token SWMTKN-1-4jonpfo50nuxvqaro4jgf96s412dec54ndnx1jncb
6ts5is89h-1fgmqvkrckcokex1oxfkcluh7 192.168.50.236:2377

To add a manager to this swarm, run 'docker swarm join-token manager' and follow
the instructions.

michael@debian:~$
```

DOŁĄCZENIE WEZŁÓW PRACOWNIKÓW DO KLAstra SWARM

```
Plik  Edycja  Widok  Wyszukiwanie  Terminal  Pomoc
michael@linuxmint:~$ docker swarm join --token SWMTKN-1-4jonpfo50nuxvqaro4j
This node joined a swarm as a worker.
michael@linuxmint:~$
```


SPRAWDZENIE STATUSU WĘZŁÓW W ROJU

```
michael@debian: ~  
michael@debian:~$ sudo docker node ls  
ID                HOSTNAME          STATUS    AVAILABILITY  MANAGER  
STATUS  ENGINE VERSION  
7zg0c1y2cw7nqm5p164zgk99r  DESKTOP-0CDREBL  Ready    Active  
26.1.1  
qxhpctxfec7cgm1y2b8ykevp *  debian           Ready    Active        Leader  
20.10.24+dfsg1  
smye2khfu687l3rzhutvfknkj  linuxmint        Ready    Active  
24.0.5  
michael@debian:~$
```

URUCHOMIENIE USŁUG ZA POŚREDNICTWEM MANAGERA

PO SKONFIGUROWANIU KLASTRA MOŻEMY DEPLOYOWAĆ NASZE USŁUGI DO KLASTRA SWARM, ZAPEWNIAJĄC SKALOWALNOŚĆ I NIEZAWODNOŚĆ APLIKACJI W ŚRODOWISKU PRODUKCYJNYM.

URUCHOMIENIE USŁUG ZA POŚREDNICTWEM MANAGERA

```
$ sudo docker service ls
```

ID	NAME	MODE	REPLICAS	IMAGE	PORTS
o274e9du03b7	my_stack_redis	replicated	1/1	redislabs/redismod:latest	*:6379->6379/tcp
ww9vrz7yh0d0	nginx	replicated	3/3	nginx:latest	*:80->80/tcp

```
$ sudo docker service ps o274e9du03b7
```

ID	NAME	IMAGE	NODE	DESIRED STATE	CURRENT STATE
4d12zwwbudn5	my_stack_redis.1	redislabs/redismod:latest	debian	Running	Running 6 minutes ago

```
$ sudo docker service ps ww9vrz7yh0d0
```

ID	NAME	IMAGE	NODE	DESIRED STATE	CURRENT STATE	ERROR	PORTS
7tj2z9ovyw4q	nginx.1	nginx:latest	debian	Running	Running 3 minutes ago		
yy1tpbrs4ftm	nginx.2	nginx:latest	linuxmint	Running	Running 3 minutes ago		
o8dx5fp1uzc7	nginx.3	nginx:latest	linuxmint	Running	Running 3 minutes ago		

CZYM JEST WSL2?

WINDOWS SUBSYSTEM FOR LINUX 2 (WSL2) TO NARZĘDZIE, KTÓRE UMOŻLIWIA URUCHAMIANIE ŚRODOWISKA LINUKSA NA SYSTEMACH WINDOWS.

TEST DZIAŁANIA DOCKERA WEWNĄTRZ WSL2

PO INSTALACJI DOCKER NA WSL2 MOŻEMY PRZETESTOWAĆ JEGO DZIAŁANIE POPRZECZ URUCHOMIENIE PROSTEGO KONTENERA.

URUCHOMIENIE PRZYKŁADU DOCKER COMPOSE

```
$ git clone https://github.com/DevxMike/konteneryzacja_projekt
$ cd konteneryzacja_projekt/docker_compose_example/flask-redis
$ docker-compose -d up

✓ redis Pulled
.
.
.
=> => naming to docker.io/library/flask-redis-web
[+] Running 3/3
✓ Network flask-redis_default Created
✓ Container flask-redis-redis-1 Started
✓ Container flask-redis-web-1 Started

$ docker ps
```

CONTAINER ID	IMAGE	COMMAND	NAMES	CREATED	STATUS
aa53ffead627	flask-redis-web	"python3 app.py"		4 minutes ago	Up 4
minutes	0.0.0.0:8000->8000/tcp, :::8000->8000/tcp		flask-redis-web-1		
30cac682d22f	redislabs/redismod	"redis-server --load..."		4 minutes ago	Up 4
minutes	0.0.0.0:6379->6379/tcp, :::6379->6379/tcp		flask-redis-redis-1		

DOCKER COMPOSE VS DOCKER SWARM

- JEST NARZĘDZIEM DO DEFINIOWANIA I URUCHAMIANIA WIELOKONTENEROWYCH APLIKACJI NA POJEDYNCZYM KOMPUTERZE LUB W ŚRODOWISKU DEWELOPERSKIM.
- POZWALA PROGRAMISTOM ZDEFINIOWAĆ CAŁĄ INFRASTRUKTURĘ APLIKACJI W PLIKACH YAML, CO UŁATWIA ZARZĄDZANIE ZALEŻNOŚCIAMI MIĘDZY KONTENERAMI, SIECIAMI I WOLUMINAMI.
- JEST IDEALNY DO URUCHAMIANIA APLIKACJI NA LOKALNYM ŚRODOWISKU, TESTOWANIA KODU I SZYBKIEGO PROTOTYPOWANIA.
- JEST NARZĘDZIEM DO ORKIESTRACJI KONTENERÓW, KTÓRE UMOŻLIWIA ZARZĄDZANIE KONTENERAMI NA WIELU MASZYNACH FIZYCZNYCH.
- UMOŻLIWIA TWORZENIE KLASTRÓW KONTENERÓW, KTÓRE POZWALAJĄ NA SKALOWANIE APLIKACJI, RÓWNOWAŻENIE OBCIĄŻENIA I ZAPEWNIENIE WYŻSZEJ DOSTĘPNOŚCI.
- NADAJE SIĘ DO ZASTOSOWAŃ PRODUKCYJNYCH, GDZIE POTRZEBNA JEST WYSOKA DOSTĘPNOŚĆ, SKALOWALNOŚĆ I MOŻLIWOŚĆ AKTUALIZACJI APLIKACJI BEZ PRZESTOJU.

INTEGRACJA Z WSL2 – WADY I ZALETY

- **ŁATWOŚĆ KONFIGURACJI:** INTEGRACJA DOCKER Z WSL2 JEST STOSUNKOWO PROSTA DO SKONFIGUROWANIA, DZIĘKI CZEMU PROGRAMIŚCI MOGĄ SZYBKO ROZPOCZĄĆ PRACĘ Z KONTENERAMI DOCKEROWYMI NA PLATFORMIE WINDOWS.
- **WYŻSZA WYDAJNOŚĆ:** URUCHAMIANIE DOCKERA WEWNĄTRZ WSL2 ZAZWYCZAJ OFERUJE LEPSZĄ WYDAJNOŚĆ NIŻ KORZYSTANIE Z DOCKER DESKTOP NA WINDOWS, PONIEWAŻ KONTENERY DZIAŁAJĄ BEZPOŚREDNIO W ŚRODOWISKU LINUX.
- **ZGODNOŚĆ Z NARZĘDZIAMI LINUXOWYMI:** KORZYSTANIE Z DOCKERA W WSL2 UMOŻLIWIA ŁATWIEJSZE DOSTOSOWANIE SIĘ DO NARZĘDZI I SKRYPTÓW PRZEZNACZONYCH DLA ŚRODOWISK LINUX.
- **ZARZĄDZANIE ZASOBAMI:** ZARZĄDZANIE ZASOBAMI, TAKIMI JAK PRZYPISYWANIE ZASOBÓW SYSTEMOWYCH DO KONTENERÓW, MOŻE BYĆ BARDZIEJ SKOMPLIKOWANE W ŚRODOWISKU WSL2 NIŻ W TRADYCYJNYM ŚRODOWISKU WINDOWS LUB LINUX.
- **BŁĘDY KONFIGURACYJNE:** NIEPRAWIDŁOWA KONFIGURACJA DOCKERA Z WSL2 MOŻE PROWADZIĆ DO RÓŻNYCH PROBLEMÓW, TAKICH JAK NIEPOPRAWNE MAPOWANIE ZASOBÓW, CO MOŻE UTRUDNIĆ KORZYSTANIE Z KONTENERÓW DOCKEROWYCH.
- **ZAŁĘŻNOŚCI SYSTEMOWE:** NIEKTÓRE APLIKACJE LUB NARZĘDZIA MOGĄ WYMAGAĆ SPECYFICZNYCH ZAŁĘŻNOŚCI SYSTEMOWYCH, KTÓRE MOGĄ BYĆ TRUDNE DO ZAINSTALOWANIA LUB SKONFIGUROWANIA W ŚRODOWISKU WSL2.