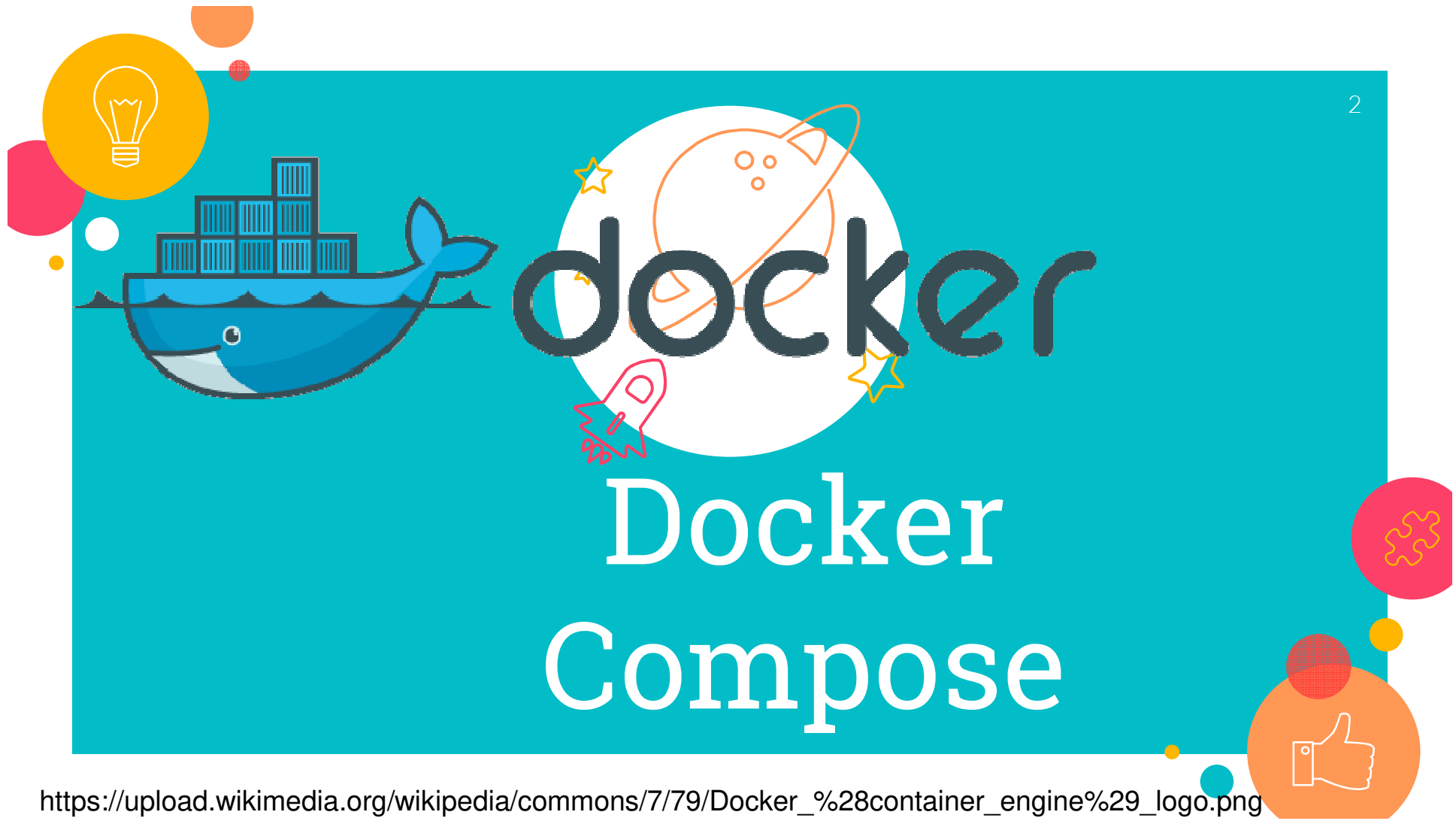


Konteneryzacja aplikacji – Docker, część 2

Marcin Szelaąg, IIn, PP



https://upload.wikimedia.org/wikipedia/commons/7/79/Docker_%28container_engine%29_logo.png



Docker Compose*

- Narzędzie do definiowania i uruchamiania **aplikacji złożonych z wielu komunikujących się ze sobą usług** (services) 3
- Do działania wymaga **Docker Engine**
- **Część** aplikacji Docker Desktop for Mac/Windows (np. c:/Program Files/Docker/Docker/resources/bin)
- **Niezależna aplikacja** na Linuxie

*<https://docs.docker.com/compose/overview/>



Docker Compose

- **Konfiguracja** aplikacji opisywana jest w pliku YAML (`docker-compose.yml`)
- Aplikacja uruchamiana jest jednym poleceniem (`docker-compose up`); uwaga – Docker Compose nie przebudowuje automatycznie, po zmianie pliku Dockerfile, obrazów usług dla których w pliku konfiguracyjnym podano atrybut `build`, jeżeli obrazy te są już dostępne w lokalnym rejestrze – można to wymusić poleceniem `docker-compose up --build`
- Komendy do zarządzania cyklem życia aplikacji

Docker Compose

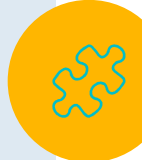
- Możliwość **sekwencyjnego wczytywania** konfiguracji:

```
docker-compose -f docker-compose.yml  
-f production.yml
```
- Możliwość **podmiany składowej usługi** działającej aplikacji (z zachowaniem `data volumes`):

```
$ docker-compose build web  
$ docker-compose up --no-deps -d web
```
- Uruchomienie tylko na pojedynczym komputerze z Docker Engine
- Specyfikacja formatu pliku konfiguracyjnego:
<https://docs.docker.com/compose/compose-file/>

Docker Compose – przykłady

-> 07_docker-compose.yml_examples.docx
(przykłady 1-3)





https://upload.wikimedia.org/wikipedia/commons/7/79/Docker_%28container_engine%29_logo.png

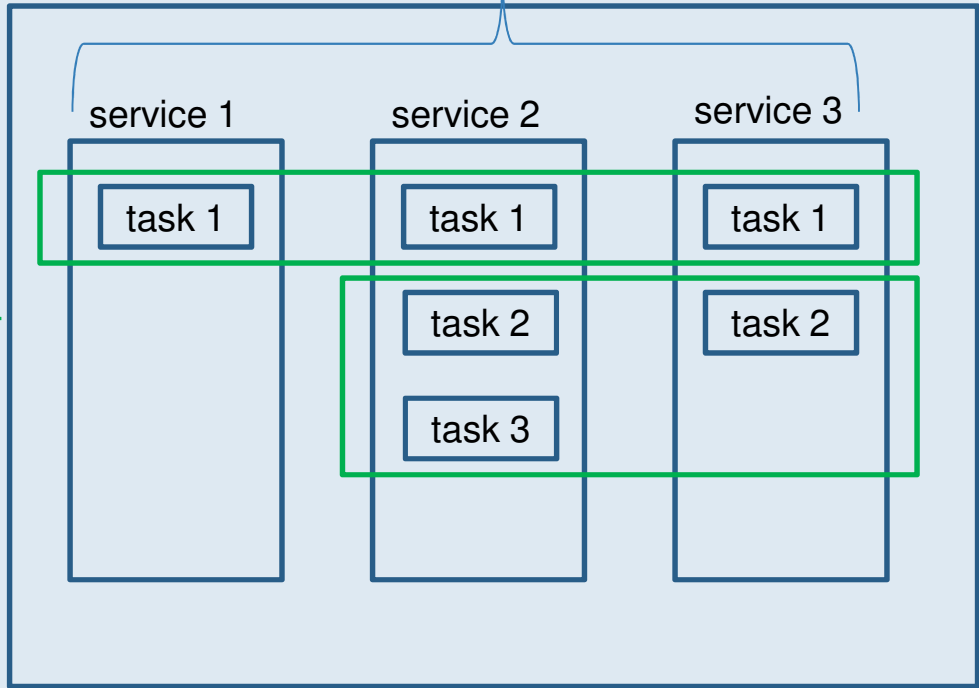
Docker Swarm

- **Swarm** – grupa hostów Dockera (engine'ów) pracujących w trybie swarm mode (klaster); każdy host (node) może pełnić rolę **managera** (manager), **pracownika** (worker), lub obie role naraz
- **Usługa** (service) – ma zdefiniowany **optymalny stan** (liczba replik kontenera, dostępne zasoby sprzętowe, udostępnione porty, itp.), który Docker stara się zachować
- Gdy worker node staje się **nieosiągalny**, jego **zadania** (kontenery składowe różnych usług) są uruchamiane na innym hoście (orkiestracja ze strony managera)
- **Zadanie** (task) = kontener + komenda do uruchomienia w kontenerze

Docker Swarm - example

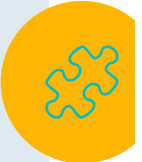


swarm (2 docker engines) stack



node 1:
worker
+
manager

node 2:
worker



Docker Swarm

- **Modyfikacja konfiguracji usługi** (np. sieci, do których usługa jest podłączona, data volumes) => Docker automatycznie uruchamia ponownie zadania (kontenery) tej usługi
- Każda usługa w swarm'ie dostaje **automatycznie adres DNS**

Docker Swarm – tworzenie

```
> docker swarm init
```

11

```
Swarm initialized: current node  
(osezla3ravy1pvrkr1cifxcrg) is now a manager.
```

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

```
docker swarm join --token SWMTKN-1-  
5jt1qysetk46vptxryketpjhjo6glj6qn3zgp1vyv05b  
4ykefy-6t53uapcov0dg6r4kt5zh9yy9  
192.168.65.3:2377
```

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

Docker Swarm – tworzenie



`docker node ls` – listowanie węzłów w₁₂
swarmie

ID	HOSTNAME	STATUS	AVAILABILITY	MANAGER STATUS
03g1y59jwfg7cf99w41t0f662	worker2	Ready	Active	
9j68exjopxe7wf16yuxml7a7j	worker1	Ready	Active	
dxn1zf6l61qsb1josjja83ngz *	manager1	Ready	Active	Leader



Docker Swarm – (ręczne) uruchomienie usługi

> **docker service create** --replicas 1 --
name helloworld alpine ping docker.com
> **docker service ls** – listuje usługi

```
C:\Users\Marcin>docker service ls
ID                NAME          MODE          REPLICAS  IMAGE
xh8v059pir9o     helloworld    replicated    1/1        alpine:latest
```

> **docker service inspect --pretty**
<SERVICE-ID>
> **docker service ps** <SERVICE-ID> – listuje
taski usługi
> **docker service scale** <SERVICE-
ID>=<NUMBER-OF-TASKS> – zmiana liczby replik
> **docker service rm** helloworld – usunięcie
usługi

Docker Swarm + docker stack deploy



Usługi można również uruchamiać z użyciem polecenia **docker stack deploy** i pliku **docker-compose.yml**

-> 07_docker-compose.yml_examples.docx
(przykład 4)



Docker Swarm + docker stack deploy

Uruchomienie aplikacji:

15

`docker swarm init` - uruchomienie swarm mode

`docker stack deploy` -c `docker-compose.yml`
<nazwa-aplikacji> => **uruchomienie/update aplikacji**

(polecenie idempotentne)

`docker stack services` <nazwa-aplikacji> =>
lista usług składowych aplikacji

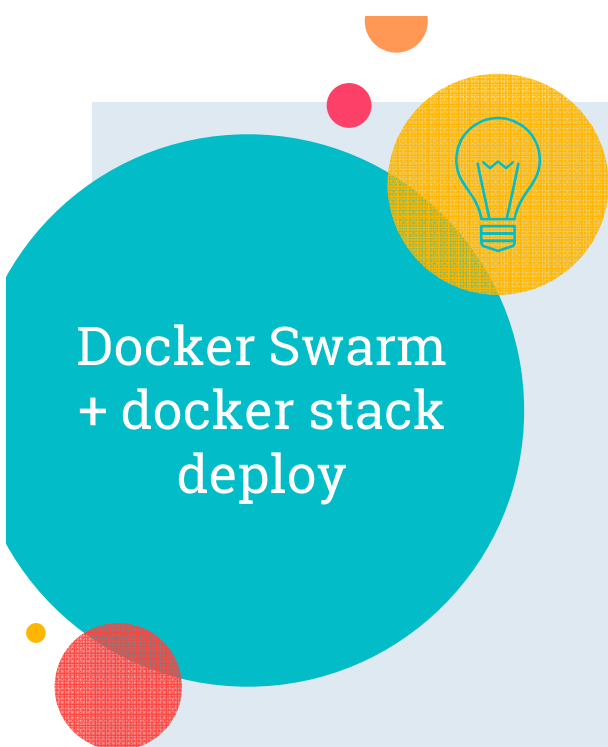
`docker service ps` <nazwa-aplikacji>_web =>
lista kontenerów (tasków) w usłudze web

`docker stack ps` <nazwa-aplikacji> => lista
wszystkich zadań aplikacji (stack'a)

Zatrzymanie aplikacji:

`docker stack rm` <nazwa-aplikacji>

`docker swarm leave --force` - zakończenie swarm
mode



Docker Swarm + docker stack deploy

-> 07_docker-compose.yml_examples.docx
(przykład 5)



docker- compose vs. docker stack deploy



docker-compose (starsze narzędzie):

- nie wymaga swarm'a; uruchamianie kontenerów na 1 maszynie
- development-oriented
- konieczna dodatkowa instalacja na Linuxie
- pliki Compose w wersji 2+ (YAML)
- ignoruje instrukcję deploy w pliku YAML

docker stack deploy (nowsze narzędzie):

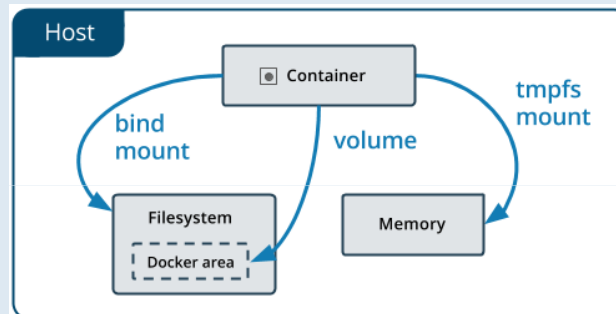
- wymaga swarm'a (przynajmniej "one-machine swarm")
- production-oriented
- zintegrowane narzędzie (uruchamiane na węźle menedżerze)
- tylko pliki Compose w wersji 3+ (YAML)
- obsługuje instrukcję deploy w pliku YAML
- ignoruje instrukcje: build, container_name, depends_on

17



Docker – pozostałe zagadnienia

- [Docker development best practices](#)
- [Docker secrets](#)
- [Docker configs](#)
- [Bind mounts/volumes/tmpfs mounts \(--mount\):](#)



- [Warstwy](#) (layers) obrazów/kontenerów
- [Networks](#) (sieci typu: bridge/host/overlay/macvlan)
- [docker machine](#)



Dziękuję!

Pytania?

