infoShare ACADEMY

# Docker Compose

# Agenda

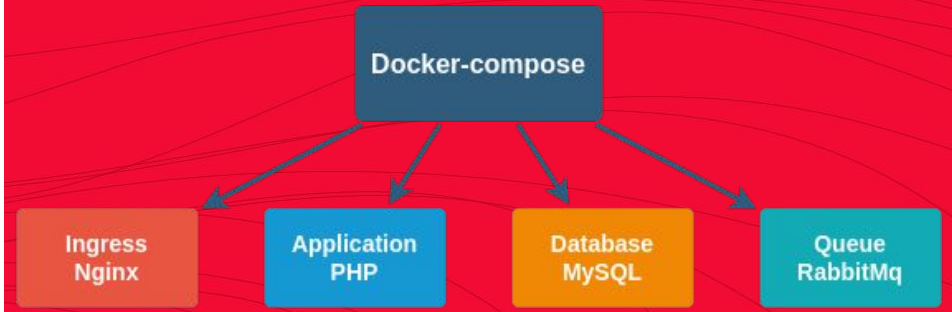
1. Docker compose co to
2. Użycie lokalne
3. Compose na serwerach?
4. Budowa pliku compose
5. Image vs build
6. Jak używać w developmencie?
7. Komendy docker compose
8. Mapowanie Docker do docker compose

# Co to ten compose?

- System zarządzania środowiskami kontenerowymi.

Zalety:

- Niezależny od systemu operacyjnego
- ułatwia tworzenie przenośnych środowisk złożonych z wielu elementów (aplikacji, baz danych, cache, kolejek)
- w jednym pliku `compose.yml` można opisać całą konfiguracje wymaganą do uruchomienia.



# Główne zastosowania

1. Tworzenie dem technologicznych
   Klaster Apache Kafki składający się z  3 nodów kafki + 3 nody zookeepera zamiast uruchamiać ca. 6-ma komendami docker run, uruchomisz jednym docker compose up
2. Tworzenie środowisk testowych
3. Tworzenie lokalnych środowisk developerskich

# Docker compose na produkcji?

Zasadniczo można, ALE:

1. Nie był tworzony z myślą o wdrożeniach produkcyjnych
2. Trzeba go uruchomić na jakiejś maszynie wirtualnej z dockerem (trzeba taką maszynę przygotować i zarządzać uruchomionym na niej composem — sam musisz wymyślić jak to zabezpieczysz czy wyskalujesz.
3. ALTERNATYWNIE: można uruchomić compose w trybie SWARM
    a. … ale trzeba maszyny dla trybu swarm uruchomić,
    b. … a potem skonfigurować,
    c. … zabezpieczyć
    d. … utrzymywać
    e. … i dorabiać wszystko, co daje Kubernetes albo serwisy cloudowe z pudła.

Dlatego, jeśli myślisz o użyciu compose file, to rozważ np użycie go razem z AWS ECS

# Zapalmy to!

# Budowa pliku docker-compose.yml

Konfiguracja w pliku compose.yml
(compose.yml/compose.yaml/docker-compose.yml/docker-compose.yaml)

- Hierarchiczna
- Opisana w formacie YAML (YAML.org, O YAMLu na wikipedii (PL) )
  - Struktura przypomina wielopoziomowy słownik/mapę/listę
  - Jest nadzbiorem formatu JSON (tj. Można listę wewnątrz opisać w formie listy JSONowej)
  - Opisuje strukturę danych
  - Obsługuje mechanizmy szablonów
- Wersjonowana
  - v1 - wycofana, możliwa do rozpoznania gdy docker-compose nie zawiera klucza `version` w konfiguracji
  - v2 - oznaczona jako 2.x w kluczu `version`
  - v3 - oznaczona jako 3.x

# Główne klucze konfiguracyjne (top level)

- Version - określa wersje konfiguracji opisanej w pliku `docker-compose.yml`
- Services - opisuje serwisy(kontenery) jakie zostaną uruchomione wraz z ich konfiguracją
- Networks - definiuje dostępne sieci do użycia w serwisach
- Volumes - definiuje przestrzenie dyskowe (volumes) do użycia w serwisach
- Configs - definicja dostępnych konfiguracji
  - montowane jako pliki w kontenerach
  - Dostępne w `/nazwa-configu` wewnątrz kontenera
- Secrets - definicja sekretów
  - Podobne do configów, ale nastawiona na wrażliwe dane
    - Certyfikaty
    - Klucze prywatne
    - Konfiguracje zawierające hasła
  - Mogą być dostępne w kontenerze jako
    - Pliki
    - Zmienne środowiskowe

# Services (docs)

- Opisują kontenery, które Compose może uruchomić
- Mapują opcje komendy `docker run` na strukture danych YAML
- Wskazują który obraz uruchomić bądź który Dockerfile zbudować i użyć
- Mogą definiować limity CPU, pamięci, blkio
- Można w nich definiować volumes, networks, configs, secrets

```yaml
services:
  foo:
    image: busybox
    environment:
      - COMPOSE_PROJECT_NAME
    command: echo "I'm running
${COMPOSE_PROJECT_NAME}"
```

# Volumes (docs)

- Tworzą trwałe przestrzenie dyskowe
- Miejsce definicji
  - Można definiować je na top-level i użyć w kilku serwisach
- Definicja mapuje opcje komendy `docker volume create`

```yaml
services:
  backend:
    image: awesome/database
    volumes:
      - db-data:/etc/data

volumes:
  db-data:
    external: true
    name: mysql-disk
```

# Networks (docs)

- Tworzą wirtualne sieci, do użycia przez kontenery Dockerowe
- Mapuje opcje komendy `docker network create` w YAMLu
- Definiowane na top-levelu, można później zmapować w serwisach

```yaml
services:
  web:
    networks:
      hostnet: {}

networks:
  hostnet:
    external: true
    name: host
```

# Configs (docs) i secrets (docs)

- Definiują wartości konfiguracyjne
- Działają tylko w trybie SWARM
- Secrets te wrażliwe (hasła, klucze kryptograficzne, certyfikaty, tokeny)
  - Podawane jako envy
  - I jako pliki
  - Mapuje komendę `docker secret create`
- Configs te których nie musimy ukrywać tak skrzętnie
  - Podawane jako pliki
  - Montowane do kontenera z uprawnieniami do odczytu
  - Mapuje komendę `docker config create`

```yaml
secrets:
  server-certificate:
    external: true
    name: "${CERTIFICATE_KEY}"


configs:
  http_config:
    external: true
    name: "${HTTP_CONFIG_KEY}"

# external oznacza, że secret/config był już uprzednio
#  utworzony i docker-compose nie będzie próbował go
#  dla nas utworzyć podczas startu
```

# Docker Compose - Obrazy

- Z DockerHuba po prostu podajemy w polu image w serwisie
- Z Prywatnego/Innego registry również podajemy w polu image
  - Pamiętaj, żeby wcześniej wykonać komendę docker login i zalogować się do registry
- Zbudować/przebudować w trakcie startu compose
  - Przydatne w developmencie obrazów
    - Zmiana
    - Rebuild serwisu
    - Zmiana
    - Rebuild serwisu
    - ...
  - Niekoniecznie zalecane przy pracy nad aplikacją
    - Języki skryptowe w trybie developerskim szybciej przeładują się z volume
    - Przebudowany obraz wcześniej oszczędza czas całego zespołu

```
version: '3.8'
services:
 test-dockerhub:
   image: busybox
   command: sleep 100000

 test-build:
   build:
     context: .
   command: 'echo "Zbudowanym
w Compose" && sleep 100000'
```

# Docker Compose - Deploy

Może deployować Docker Compose na:

- Zewnętrzne hosty Dockera (link)
- Chmury
  - AWS ECS (link)
  - Azure ACI (link)
- Klastry Docker Swarm (link)

Chyba nigdy osobiście nie użyłem tej możliwości...

# Inne specjalności

- Fragmenty i kotwice YAMLowe
- Ustawianie parametrów rozszerzeń (x-*)
- Zmienne i interpolacja
- Nazwa deploymentu
  - Dostępna jako zmienna $COMPOSE_PROJECT_NAME do interpolacji
  - Może być użyta jako nazwa środowiska (gdy np. Uruchamiasz kilka razy ten sam compose na swoim komputerze)
  - Domyślnie to nazwa katalogu w którym znajduje się `compose.yml`
  - Można nadpisać ją przy uruchamianiu stacka (wyeksportuj zmienną `COMPOSE_PROJECT_NAME`)
- Profile
  - Możesz otagować sobie zasoby compose nazwą profilu i w zależności od wybranego profilu przy starcie startować wybraną część dostępnych serwisów

# docker-compose - polecenia

| | |
|---|---|
| build | Zbuduj lub przebuduj usługi |
| config | Sprawdź poprawność i wyświetl plik docker-compose |
| create | Utwórz usługi |
| down | Zatrzymaj i usuń zasoby |
| events | Otrzymuj zdarzenia w czasie rzeczywistym z kontenerów |
| exec | Wykonaj polecenie w uruchomionym kontenerze |
| help | Uzyskaj pomoc w sprawie polecenia |
| images | Wyświetl listę obrazów |
| kill | Zabij kontenery |
| logs | Zobacz dane wyjściowe / logi z kontenerów |
| pause | Wstrzymaj usługi |
| port | Wyświetl port publiczny w celu powiązania portu |
| ps | Lista kontenerów |
| pull | Pobierz obrazy źródłowe |
| push | Wypchnij obrazy |
| restart | Uruchom ponownie usługi |
| rm | Usuń zatrzymane kontenery |
| run | Uruchom jednorazowe polecenie |
| scale | Ustaw liczbę kontenerów na usługę |
| start | Uruchom usługi |
| stop | Zatrzymaj usługi |
| top | Wyświetl uruchomione procesy |
| unpause | Wznów usługi |
| up | Twórz i uruchamiaj kontenery |
| version | Pokaż informacje o wersji |

# Więcej dobrych zasobów?

- compose.yml specyfikacja (dokumentacja)
- Docker Compose CLI (dokumentacja)
- Jerome Petazzo - Container Training - kompleksowa prezentacja (900 slajdów) Jerome Petazzo twórcy dockera rozbijająca w zasadzie całego dockera, docker compose i jeszcze więcej na atomy, pisana prostym językiem (EN)
- Awesome Compose - lista dodatków, aplikacji, przykładów, wykładów nt. Docker-compose i powiązanych