Docker – wprowadzenie

Adam Stasiak – Automatyzacja Testowania w Praktyce 2023

Docker

- Platforma open-source
- Służy do uruchamiania aplikacji w kontenerach
- Udostępnia CLI do budowania,uruchamiania i publikowania obrazów

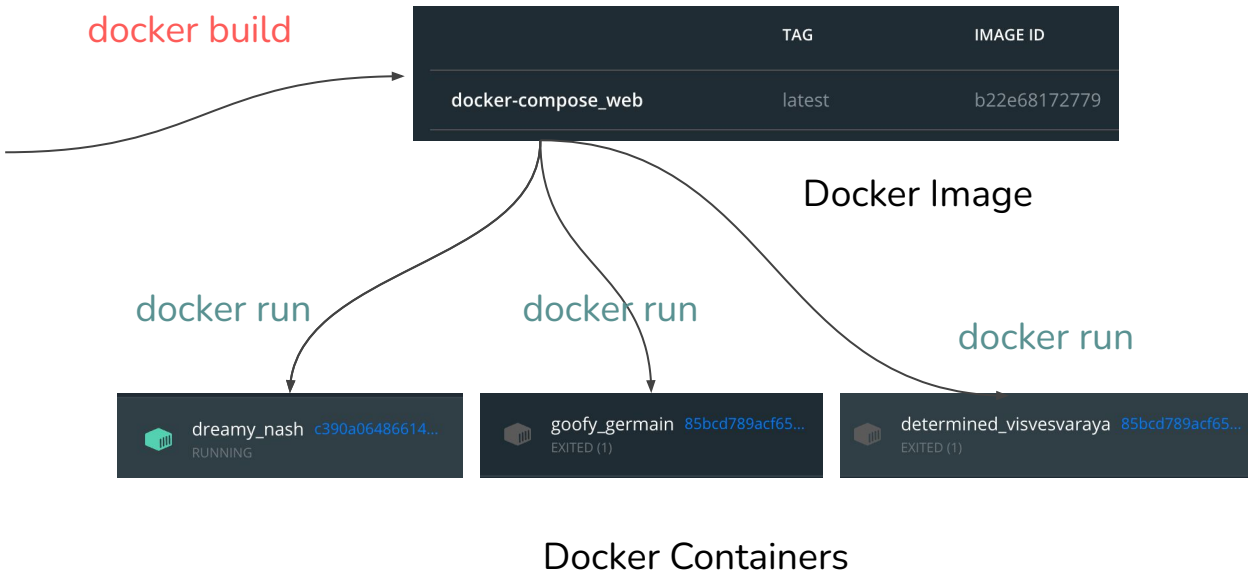
Kontener

- Proces uruchomiony w izolowanym środowisku
- Jest budowany z obrazów
- Ma dostęp do zasobów naszego komputera

Wykorzystanie Dockera

```
# zacznij od oficjalnego obrazu Pythona 3.9
FROM python:3.9-slim-buster

# Set the working directory in the container to /app
WORKDIR /app

# Copy app.py templates/index.html requirements.txt
COPY app.py /app
COPY templates/index.html /app/templates/
COPY requirements.txt /app

# Install any dependencies in requirements.txt
RUN pip install --no-cache-dir -r requirements.txt

# Expose port 8000 for the application to listen on
EXPOSE 8000

# Set the command to run the application
CMD ["python", "app.py"]
```

Dockerfile



Budowanie obrazu dockerowego

```
adamstasiak@MacBook-Pro-Adam-2 web % ls
Dockerfile              app.py                  requirements.txt        templates
adamstasiak@MacBook-Pro-Adam-2 web % docker build .
[+] Building 7.3s (11/11) FINISHED
 => [internal] load build definition from Dockerfile                                0.1s
 => => transferring dockerfile: 580B                                                0.0s
 => [internal] load .dockerignore                                                   0.0s
 => => transferring context: 2B                                                     0.0s
 => [internal] load metadata for docker.io/library/python:3.9-slim-buster           0.0s
 => [1/6] FROM docker.io/library/python:3.9-slim-buster                             0.0s
 => [internal] load build context                                                   0.1s
 => => transferring context: 1.26kB                                                 0.0s
 => CACHED [2/6] WORKDIR /app                                                       0.0s
 => CACHED [3/6] COPY app.py /app                                                   0.0s
 => [4/6] COPY templates/index.html /app/templates/                                 0.1s
 => [5/6] COPY requirements.txt /app                                                0.1s
 => [6/6] RUN pip install --no-cache-dir -r requirements.txt                        6.7s
 => exporting to image                                                              0.2s
 => => exporting layers                                                             0.2s
 => => writing image sha256:75c835e41881fb6eeea54e216b631ce9c4ae8c9ef4e178ca5edd05db4d4b1529  0.0s
adamstasiak@MacBook-Pro-Adam-2 web %
```

Uruchamianie obrazu dockerowego (kontenera)

```
adamstasiak@MacBook-Pro-Adam-2 web % docker build .
[+] Building 7.3s (11/11) FINISHED
 => [internal] load build definition from Dockerfile                                0.1s
 => => transferring dockerfile: 580B                                                0.0s
 => [internal] load .dockerignore                                                   0.0s
 => => transferring context: 2B                                                     0.0s
 => [internal] load metadata for docker.io/library/python:3.9-slim-buster           0.0s
 => [1/6] FROM docker.io/library/python:3.9-slim-buster                             0.0s
 => [internal] load build context                                                   0.1s
 => => transferring context: 1.26kB                                                 0.0s
 => CACHED [2/6] WORKDIR /app                                                       0.0s
 => CACHED [3/6] COPY app.py /app                                                   0.0s
 => [4/6] COPY templates/index.html /app/templates/                                 0.1s
 => [5/6] COPY requirements.txt /app                                                0.1s
 => [6/6] RUN pip install --no-cache-dir -r requirements.txt                        6.7s
 => exporting to image                                                              0.2s
 => => exporting layers                                                             0.2s
 => => writing image sha256:75c835e41881fb6eeea54e216b631ce9c4ae8c9ef4e178ca5edd05db4d4b1529  0.0s
adamstasiak@MacBook-Pro-Adam-2 web % docker run -it 75c835e41881fb6eeea54e216b631ce9c4ae8c9ef4e178ca5edd05db4d4b1529
```

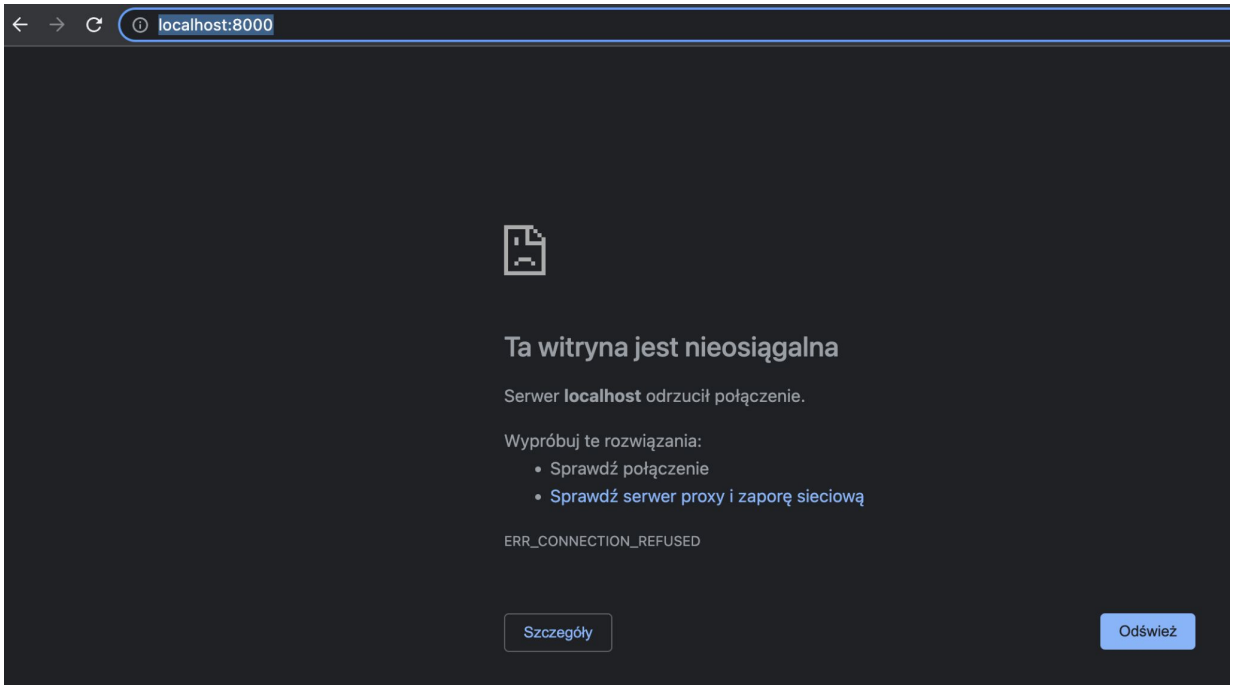
Uruchamianie obrazu dockerowego (kontenera)

```
adamstasiak@MacBook-Pro-Adam-2 web % docker run -it 75c835e41881fb6eeea54e216b631ce9c4ae8c9ef4e178ca5edd05db4d4b1529
 * Serving Flask app 'app'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on all addresses (0.0.0.0)
 * Running on http://127.0.0.1:8000
 * Running on http://172.17.0.2:8000
Press CTRL+C to quit
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 131-266-772
```

Wycinek z Dockerfile tej usługi

```
# Expose port 8000 fo                ion to listen on
EXPOSE 8000
```

Uruchamianie obrazu dockerowego (kontenera)

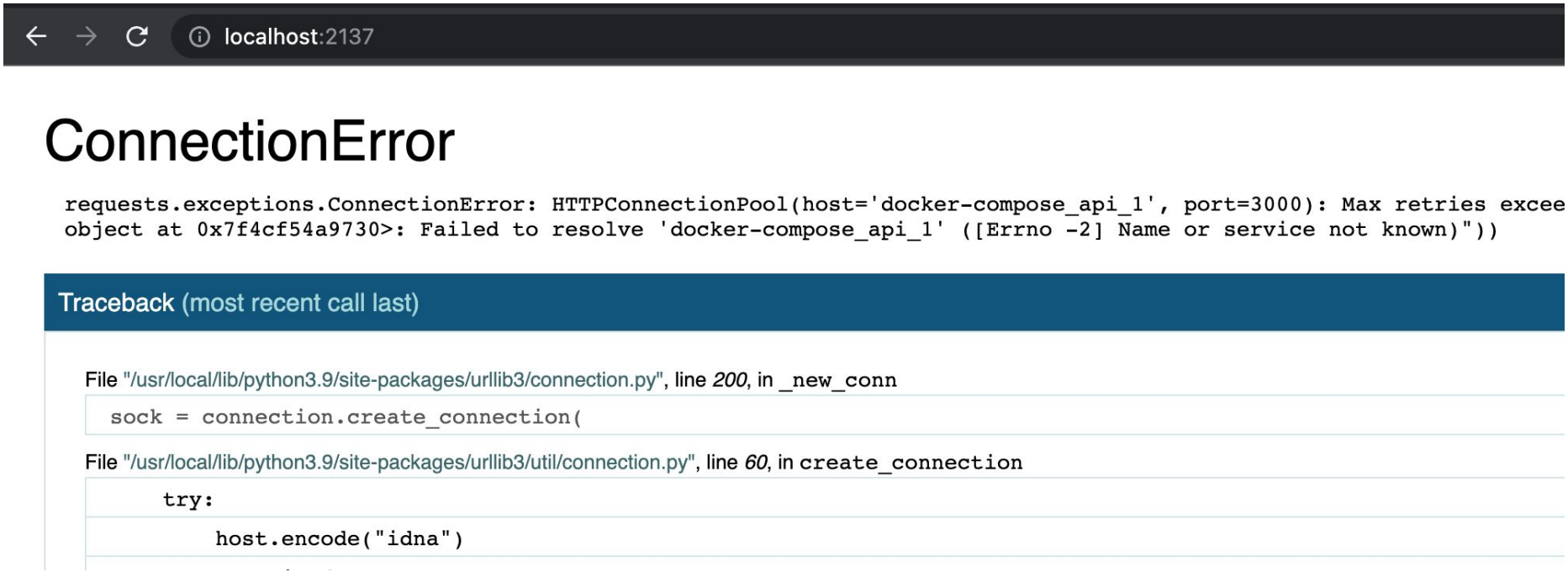


Mapowanie lokalnego portu z portem w kontenerze

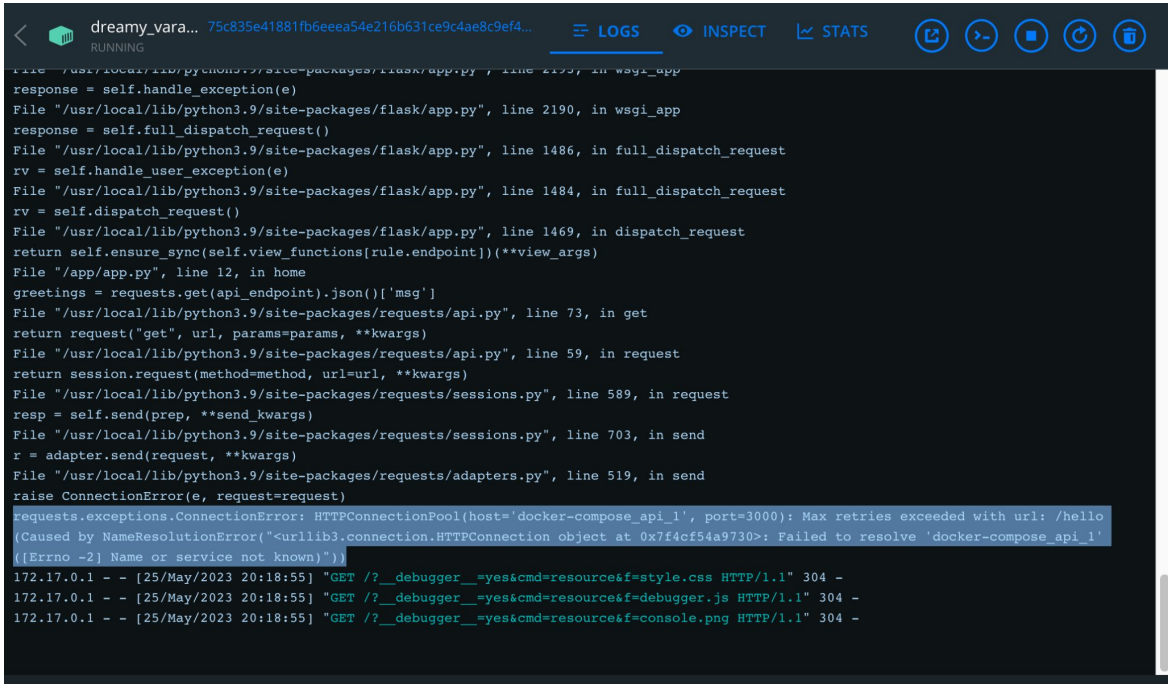
```
docker run -it -d -p [port_naszego_komputera]:[port_wewnątrz_kontenera]
 [id_obrazu]
```

```
docker run -it -d -p 2137:8000
75c835e41881fb6eeea54e216b631ce9c4ae8c9ef4e178ca5edd05db4d4b1529
```

Kontener z ustawionymi portami



Kontener z ustawionymi portami



Pozostałe komendy

Służy do wylistowania wszystkich kontenerów

docker ps

```
adamstasiak@MacBook-Pro-Adam-2 ~ % docker ps
CONTAINER ID   IMAGE                                   COMMAND                  CREATED          STATUS          PORTS                                                                                        NAMES
b5f3131f7059   75c835e41881                            "python app.py"          10 minutes ago   Up 10 minutes   0.0.0.0:2137->8000/tcp, :::2137->8000/tcp                                                    dreamy_varahamihira
bf60f7c9b800   bitnami/testlink:1.9.20-debian-10-r301  "/opt/bitnami/script…"   31 hours ago     Up 30 hours     0.0.0.0:80->8080/tcp, :::80->8080/tcp, 0.0.0.0:443->8443/tcp, :::443->8443/tcp               localenvronment_ex_testlink_1
1e93bb875dba   jenkins/jenkins:lts-jdk11               "/sbin/tini -- /usr/…"   31 hours ago     Up 30 hours     50000/tcp, 0.0.0.0:5001->5000/tcp, :::5001->5000/tcp, 0.0.0.0:8089->8080/tcp, :::8089->8080/tcp   jenkinsLocal
b24bc92eb07d   bitnami/mariadb:latest                  "/opt/bitnami/script…"   31 hours ago     Up 30 hours     3306/tcp                                                                                     localenvronment_ex_mariadb_1
```

możemy zawęzić wyniki z flagą –filter

Pozostałe komendy

Pokazuje nasze wszystkie obrazy, które posiadamy lokalnie

docker images

```
adamstasiak@MacBook-Pro-Adam-2 ~ % docker images
REPOSITORY              TAG                                     IMAGE ID       CREATED          SIZE
<none>                  <none>                                  75c835e41881   24 minutes ago   132MB
docker-compose_web      latest                                  b22e68172779   4 days ago       132MB
docker-compose_api      latest                                  b57fa4ca368e   4 days ago       916MB
<none>                  <none>                                  2cf2cebdc4b1   6 days ago       132MB
<none>                  <none>                                  91230d6efd81   6 days ago       132MB
<none>                  <none>                                  08cf86d5f2ac   6 days ago       132MB
<none>                  <none>                                  9fea868e09ab   6 days ago       132MB
<none>                  <none>                                  c3ff17eb0c79   6 days ago       132MB
<none>                  <none>                                  af4eeef14ddb   6 days ago       132MB
<none>                  <none>                                  6ca4fd96b635   6 days ago       132MB
<none>                  <none>                                  e6bb61c8a767   6 days ago       132MB
<none>                  <none>                                  314099881d36   6 days ago       132MB
my-custom-image         latest                                  0feae49216e6   7 days ago       2.81GB
<none>                  <none>                                  1ce2cc011665   7 days ago       2.81GB
```

Pozostałe komendy

Pobiera ale nie uruchamia obraz dockerowy

docker pull

```
adamstasiak@MacBook-Pro-Adam-2 ~ % docker pull alpine
Using default tag: latest
latest: Pulling from library/alpine
8a49fdb3b6a5: Pull complete
Digest: sha256:02bb6f428431fbc2809c5d1b41eab5a68350194fb508869a33cb1af4444c9b11
Status: Downloaded newer image for alpine:latest
docker.io/library/alpine:latest
```

Pozostałe komendy

`docker stop`

Zatrzymaj kontener

`docker rm`

Usuń kontener

`docker rmi`

Usuń obraz

Komenda exec

```
adamstasiak@MacBook-Pro-Adam-2 ~ % docker exec -it dreamy_varahamihira sh
```

- Pozwala na wykonanie dowolnego polecenia powłoki wewnątrz kontenera
- Jeżeli nasz kontener zajmuje się tylko wykonaniem jednego polecenia a potem “znika” możemy otworzyć w nim terminal i przejrzeć zawartość

Docker multi-stage

```
# Etap 1: Budowanie aplikacji
FROM node:14 AS builder
WORKDIR /app
COPY . .
RUN npm install
RUN npm run build

# Etap 2: Produkcja
FROM nginx:latest
COPY --from=builder /app/build /usr/share/nginx/html
EXPOSE 80
CMD ["nginx", "-g", "daemon off;"]
```

- `FROM` – Określa bazowy obraz dla danego etapu.
- `COPY/ADD` – Kopiuje pliki z jednego etapu do drugiego.
- `RUN` – Wykonuje polecenia wewnątrz danego etapu.
- `--from` – Określa, z którego etapu ma zostać skopiowany dany plik lub folder.

Docker compose

- Pozwala na uruchamianie wielu kontenerów równolegle
- Określa konfigurację dla zestawu kontenerów - porty, zmienne środowiskowe itd.
- Ułatwia tworzenie całego systemu pojedynczą komendą

Podstawowa składnia

```yaml
version: '3'
services:
  web:
    build: .
    ports:
      - "80:80"
  db:
    image: postgres
    environment:
      - POSTGRES_PASSWORD=secret
```

- version – określa wersję składni **docker-compose.yaml**
- services – definiuje listę usług do uruchomienia
- build/image – określa sposób budowania dockera – na podstawie pliku Dockerfile lub obrazu
- ports – mapuje porty kontenera na porty hosta
- environment – ustawia zmienne środowiskowe dla **kontenera**

Dodatkowe pola

```yaml
tl-testlink:
  container_name: tl-testlink
  image: bitnami/testlink:1.9.20
  ports:
    - 80:8080
    - 443:8443
  volumes:
    - ./testlink/testlink_data:/bitnami/testlink
  environment:
    - ALLOW_EMPTY_PASSWORD=yes
    - TESTLINK_DATABASE_HOST=tl-mariadb
    - TESTLINK_DATABASE_PORT_NUMBER=3306
    - TESTLINK_DATABASE_USER=tl_testlink
    - TESTLINK_DATABASE_NAME=tl_testlink
    - TESTLINK_USERNAME=admin
    - TESTLINK_PASSWORD=admin
    - TESTLINK_EMAIL=admin@admin.com
  depends_on:
    - tl-mariadb
  networks:
    - host
```

- container_name – nazwa dla kontenera
- volumes – pozwala na montowanie woluminów np. udostępnianie danych między kontenerami lub zapisywanie stałego stanu
- depends_on – wymusza oczekiwanie na start danego kontenera zanim wystartuje konfigurowany
- networks – ustala sieci dla kontenerów (mogą być niestandardowe)

Podstawowe komendy

- docker-compose up: Uruchamia kontenery na podstawie konfiguracji z pliku docker-compose.yml
- docker-compose down: Zatrzymuje i usuwa kontenery
- docker-compose build: Buduje obrazy kontenerów na podstawie konfiguracji

Przykład użycia: