

Sprawozdanie z laboratorium 5

Jakub Zelmanowicz, IO 6.10, 93075

Zadanie 1:

Wolumen został utworzony, a katalog umieszczony wewnątrz kontenera:

```
student@student:~$ docker run --mount source=RedisRob,target=/data --name RedisRob -it redis sh
# ls -l
total 4
-rw-r--r-- 1 redis redis 88 May  5 10:34 dump.rdb
# cd /
# ls -l
total 68
drwxr-xr-x 1 root root 4096 Apr 27 22:35 bin
drwxr-xr-x 2 root root 4096 Mar 19 13:46 boot
drwxr-xr-x 2 redis redis 4096 May  5 10:34 data
drwxr-xr-x 5 root root 360 May  5 10:36 dev
drwxr-xr-x 1 root root 4096 May  5 10:36 etc
drwxr-xr-x 2 root root 4096 Mar 19 13:46 home
drwxr-xr-x 1 root root 4096 Apr 27 22:35 lib
drwxr-xr-x 2 root root 4096 Apr 18 00:00 lib64
drwxr-xr-x 2 root root 4096 Apr 18 00:00 media
drwxr-xr-x 2 root root 4096 Apr 18 00:00 mnt
drwxr-xr-x 2 root root 4096 Apr 18 00:00 opt
dr-xr-xr-x 360 root root  0 May  5 10:36 proc
drwxr-xr-x 1 root root 4096 Apr 20 13:28 root
drwxr-xr-x 3 root root 4096 Apr 18 00:00 run
drwxr-xr-x 2 root root 4096 Apr 18 00:00/sbin
drwxr-xr-x 2 root root 4096 Apr 18 00:00/srv
dr-xr-xr-x 13 root root  0 May  5 10:36 sys
drwxrwxrwt 1 root root 4096 Apr 27 22:35 tmp
drwxr-xr-x 1 root root 4096 Apr 18 00:00 usr
drwxr-xr-x 1 root root 4096 Apr 18 00:00 var
#
```

Wnioski:

W prosty sposób możemy zamieścić nasz katalog roboczy wewnątrz kontenera. Mountpoint musi być zawsze określony w momencie uruchamiania (run) kontenera.

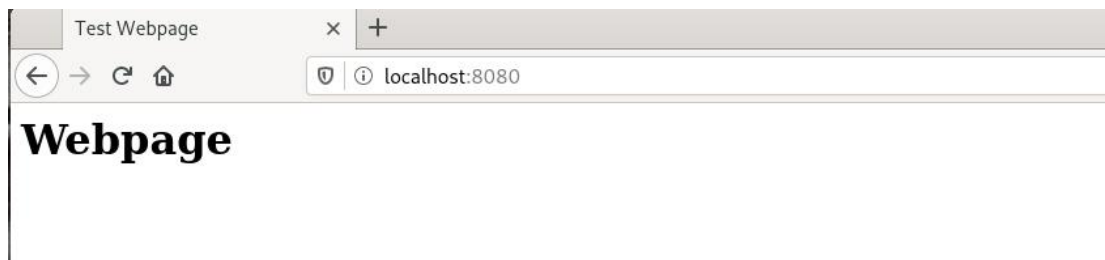
Zadanie 2:

Opracowano trzy identyczne pliki Dockerfile o zawartości:

```
FROM httpd
VOLUME [ "webpage" ]
```

Uruchomiono główny kontener za pomocą komendy:

```
docker run --read-only --name reader -v /home/debian/Documents/Docker/Lab_5/webpage:/usr/local/apache2/htdocs/ -p 8081:80 -d webpage
```



Jak widzimy na powyższym obrazku, kontener wraz ze stroną działają

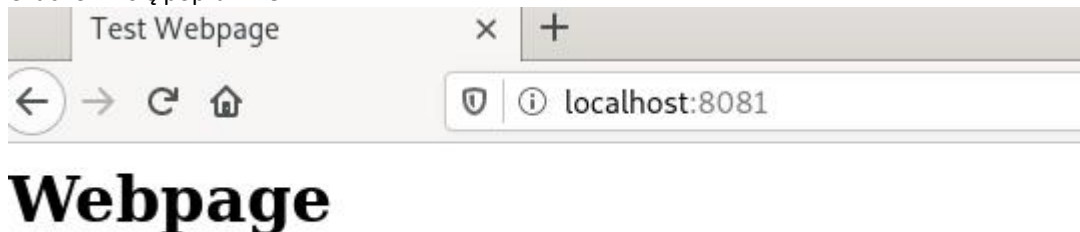
Uruchomienie kontenera w trybie read-only z obrazem httpd nie jest możliwe z powodu braku możliwości zapisu pliku log:

```
AH00558: httpd: could not reliably determine the server's fully qualified domain name, using 172.17.0.3. Set the 'ServerName' directive globally to suppress this message
AH00558: httpd: could not reliably determine the server's fully qualified domain name, using 172.17.0.3. Set the 'ServerName' directive globally to suppress this message
[Mon May 09 17:22:43.773437 2022] [core:error] [pid 1:tid 140313801428288] (30)Read-only file system: AH00899: could not create /usr/local/apache2/logs/httpd.pid.QVQlwg
[Mon May 09 17:22:43.773520 2022] [core:error] [pid 1:tid 140313801428288] AH00100: httpd: could not log pid to file /usr/local/apache2/logs/httpd.pid
```

Trzeci kontener uruchomiony przy pomocy komendy:

```
docker run --name rw -v /home/debian/Documents/Docker/Lab_5/webpage:/usr/local/apache2/htdocs/ -p 8081:80 webpage-rw
```

Uruchomił się poprawnie:



Wnioski:

Volume to bardzo przydatne narzędzie umożliwiające pracę live nad projektem i uruchomienie go na żywo.

Zadanie 3:

Kontener na bazie obrazu 'registry' został uruchomiony:

```
debian@debian10:~/Documents/Docker/Lab_5/Registry$ docker run -p 6677:5000 -d -v /home/debian/Documents/Docker/Lab_5/Registry:/var/lib/registry --name reg registry
1404ab4c76ab589f44cad94402d3bb056b96c7ca86e94db5fb3177dd104eda90
debian@debian10:~/Documents/Docker/Lab_5/Registry$
```

Do utworzonego kontenera registry wysłano pobrany obraz ubuntu o nazwie 'ubuntu_local':

```
debian@debian10:~/Documents/Docker/Lab_5$ docker push localhost:6677/ubuntu_local
Using default tag: latest
The push refers to repository [localhost:6677/ubuntu_local]
e59fc9495612: Pushed
latest: digest: sha256:aa6c2c047467afc828e77e306041b7fa4a65734fe3449a54aa9c280822b0d87d size: 529
debian@debian10:~/Documents/Docker/Lab_5$
```

Obraz został zapisany w systemie hosta pomyślnie:

```
debian@debian10:~/Documents/Docker/Lab_5/Registry$ ls -l
total 4
drwxr-xr-x 3 root root 4096 May  9 13:17 docker
debian@debian10:~/Documents/Docker/Lab_5/Registry$ cd docker/
debian@debian10:~/Documents/Docker/Lab_5/Registry/docker$ ls
registry
debian@debian10:~/Documents/Docker/Lab_5/Registry/docker$ cd registry/
debian@debian10:~/Documents/Docker/Lab_5/Registry/docker/registry$ ls
v2
debian@debian10:~/Documents/Docker/Lab_5/Registry/docker/registry$ cd v2
debian@debian10:~/Documents/Docker/Lab_5/Registry/docker/registry/v2$ ls
blobs repositories
debian@debian10:~/Documents/Docker/Lab_5/Registry/docker/registry/v2$ cd repositories/
debian@debian10:~/Documents/Docker/Lab_5/Registry/docker/registry/v2/repositories$ ls
ubuntu_local
debian@debian10:~/Documents/Docker/Lab_5/Registry/docker/registry/v2/repositories$
```

Wnioski:

To rozwiązanie tworzy niesamowite możliwości budowy własnych repozytoriów np. w firmach.

Zadanie 4:

Kontener został uruchomiony z 512MB pamięci operacyjnej oraz wykorzystaniem jednego rdzenia:

```
debian@debian10:~/Documents/Docker/Lab_5$ docker run -m 512m --cpus="1" -it alpine sh
Unable to find image 'alpine:latest' locally
latest: Pulling from library/alpine
df9b9388f04a: Already exists
Digest: sha256:4edbd2beb5f78b1014028f4fbb99f3237d9561100b6881aabbf5acce2c4f9454
Status: Downloaded newer image for alpine:latest
WARNING: Your kernel does not support swap limit capabilities or the cgroup is not mounted. Memory limited without swap.
/ #
```

Wnioski:

Rozwiązanie to umożliwia zarządzanie kontenerami w momencie przykładowo dużego obciążenia.

Możemy też decydować, ile kontenerów uruchomić, by mieć pełną kontrolę nad zasobami wykorzystywanymi przez nasze kontenery.