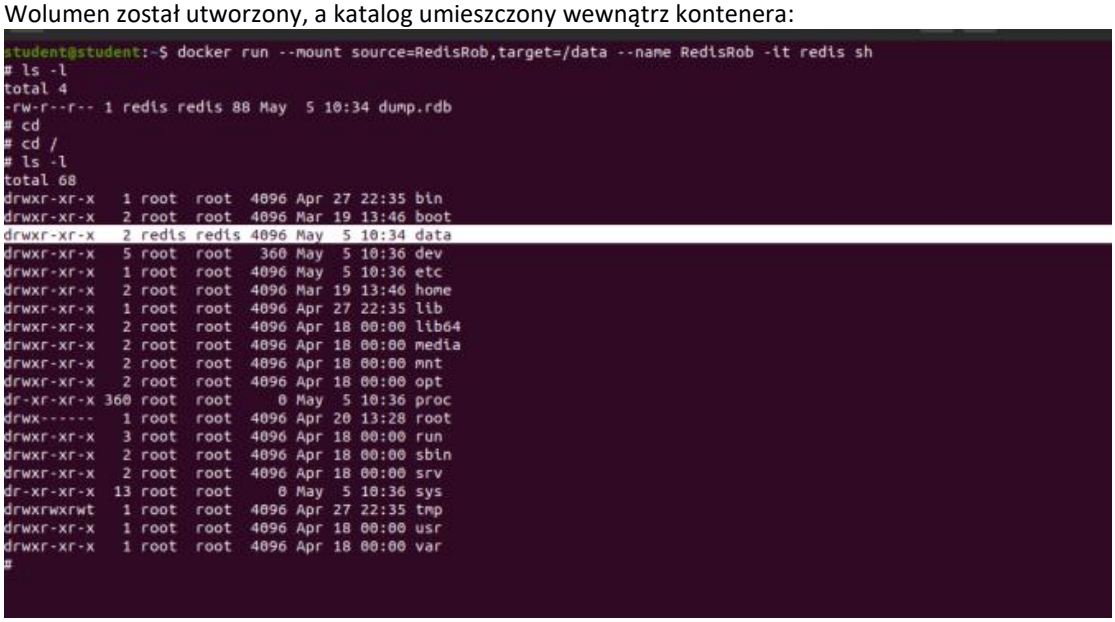
Sprawozdanie z laboratorium 5

Jakub Zelmanowicz, IO 6.10, 93075

Zadanie 1:

Wolumen został utworzony, a katalog umieszczony wewnątrz kontenera:



Wnioski:

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

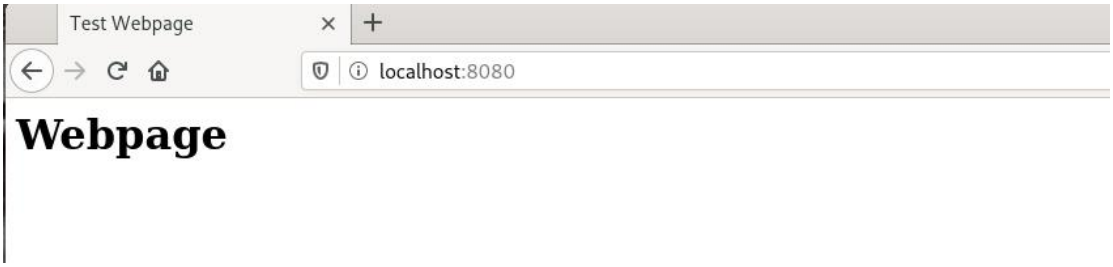
Zadanie 2:

Opracowano trzy identyczne pliki Dockerfile o zawartości:

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

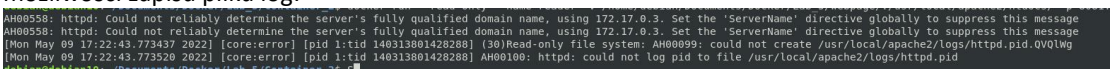
Uruchomiono główny kontener za pomocą komendy:

```
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

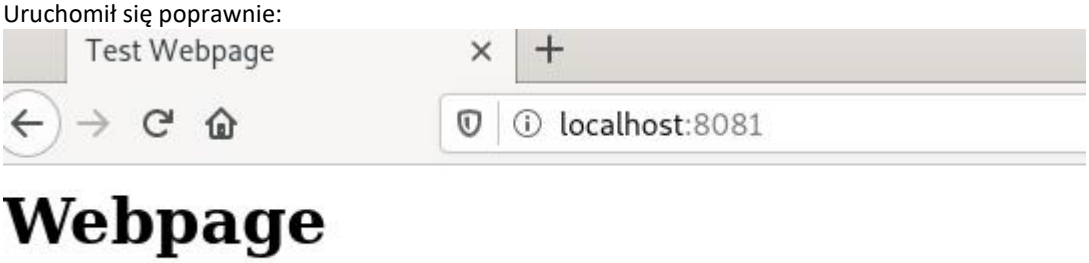
Uruchomienie kontenera w trybie read-only z obrazem httpd nie jest możliwe z powodu braku możliwości zapisu pliku log:



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$
```

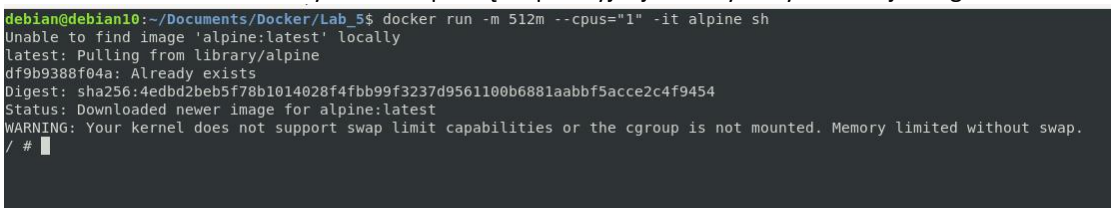
Obraz został zapisany w systemie hosta pomyślnie:

```
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$ S
```

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:



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.