**Laboratorium 4 Chmury Obliczeniowe**

**Mateusz Markowski gr 7.4/7**

**Zadanie 1**

Proszę zweryfikować wykorzystanie wolumenów, według dwóch poniższych zadań:

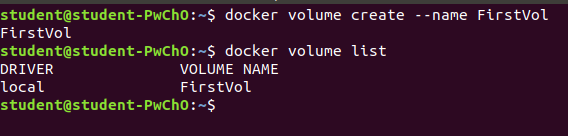
A.

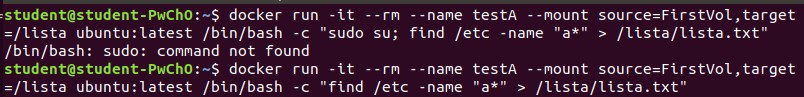
- Proszę utworzyć kontener na bazie obrazu ubuntu:latest wraz z wolumenem *FirstVol*,. Kontener ma za zadanie zapisać do pliku *lista.txt* nazwy wszystkich plików zawierających literę ***a*** z katalogu /etc kontenera. Plik *lista.txt* powinien być umieszczony w katalogu, w którym podmontowano wolumin *FirstVol* a kontener po wykonaniu tej pracy powinien zakończyć działanie.

- Proszę uruchomić kolejny kontener na bazie ubuntu:latest z podłączonym wolumenem *FirstVol*. Kontener powinien wykonać zadanie polegające na zliczeniu ilości słów w pliku *lista.txt* (plik powinien być dostępny na katalogu, w którym zamontowano ten wolumen) a następnie wyświetleniu tej liczby. Skrypt wraz z niezbędnymi komentarzami i ilustrację jego działania należy umieścić w sprawozdaniu.

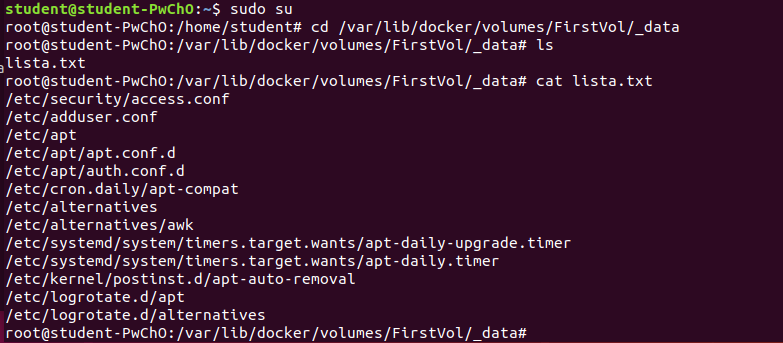
1. Proszę utworzyć własny, nowy kontener o nazwie RedisRob. Następnie uruchomić serwer redis w wersji 5.0.0 (latest) z wykorzystaniem tego wolumeny zamiast anonimowego kontenera tak by on właśnie był podnontowany w katalogu /data systemu plików serwera. Poprawność konfiguracji proszę potwierdzić wynikiem działania odpowiedniego polecenia. Użyte polecenia i wynik działania proszę umieścić w sprawozdaniu

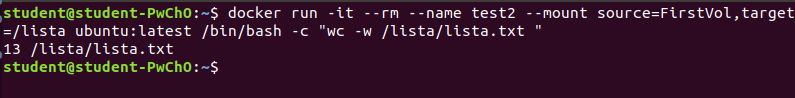
A.



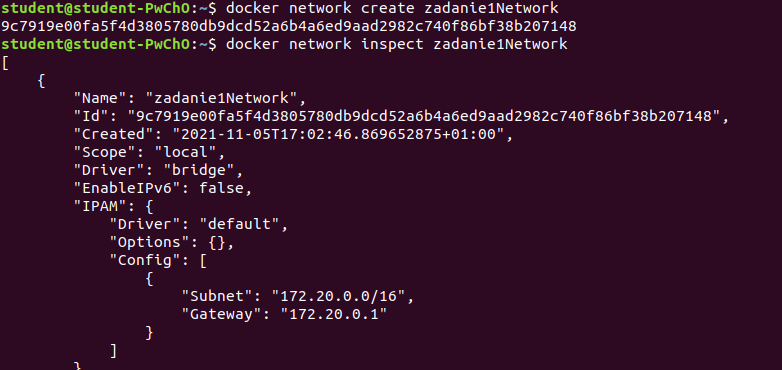


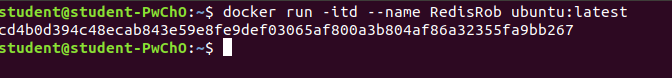
.

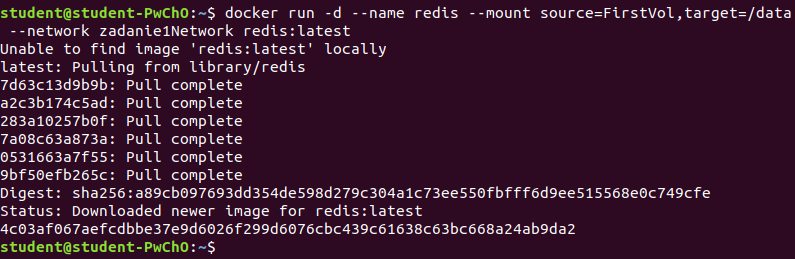




B.









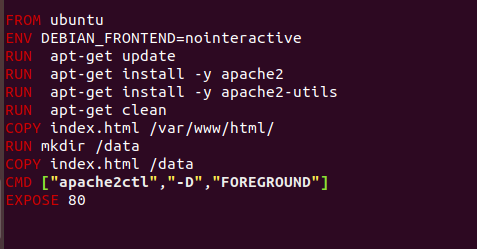
**Zadanie 2**

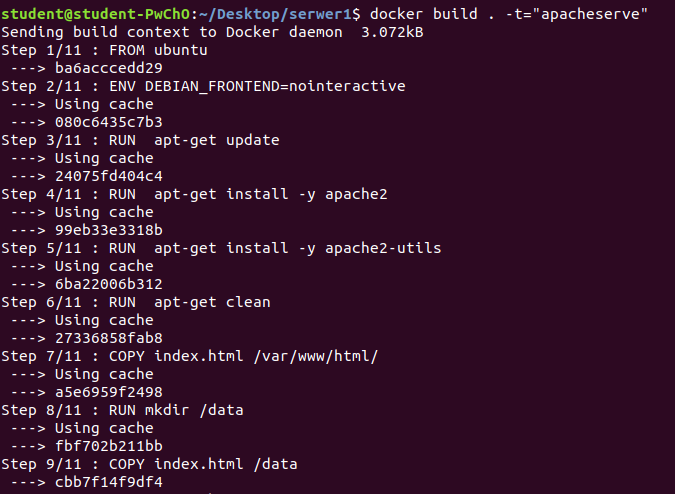
Wykorzystując umiejętności z jednego z poprzednich ćwiczeń, proszę opracować możliwie najprostsze pliki Dockerfile dla 3 serwerów WWW na bazie serwera Apache.

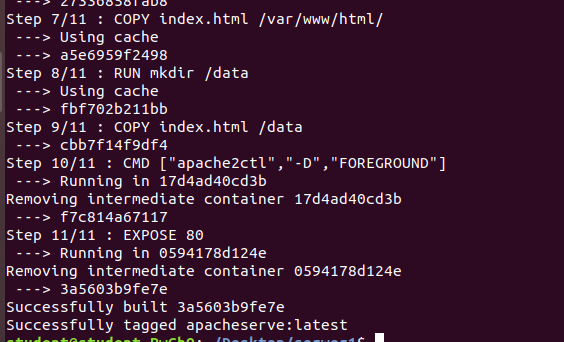
- Należy też przygotować własną, prostą, statyczną stronę html. Pierwszy Dockerfile powinien pozwolić na stworzenie kontenera Docker z serwerem WWW, który wyświetla zawartość pliku html umieszczonego na utworzonym, zewnętrznym wolumenie.

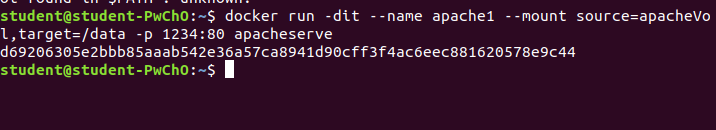
- Kolejne dwa serwery WWW powinny korzystać z tego (utworzonego dla I serwera) wolumenu z tym, że pierwszy z nich z prawami „read-write” a drugi z prawami „read only). Oba mają wyświetlać tą samą stronę

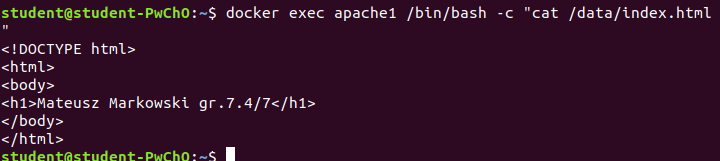
1. Stworzenie Dockerfile z serwerem Apache oraz Volumenem

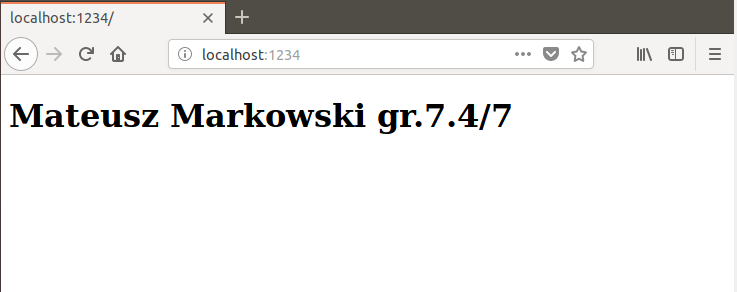


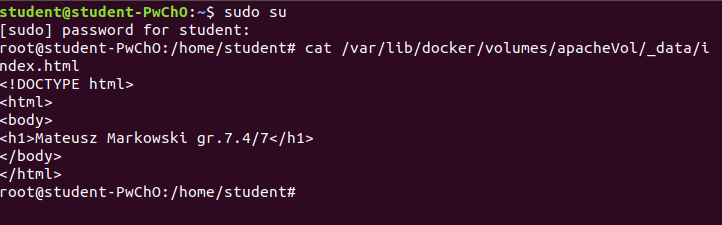




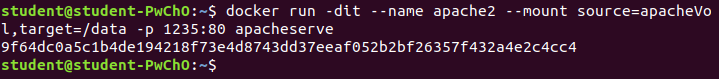


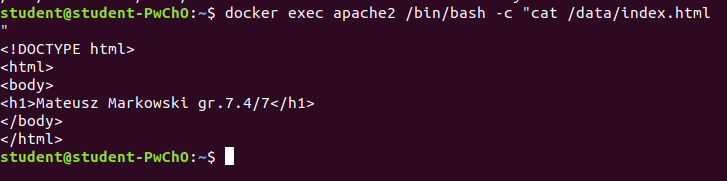


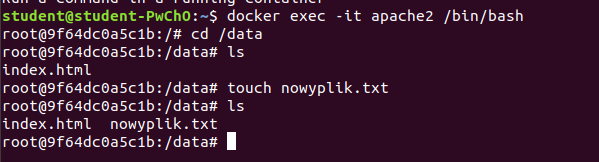


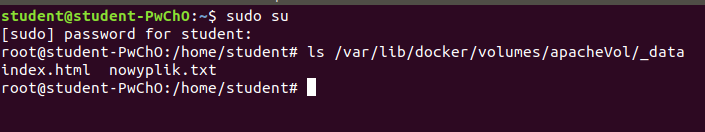


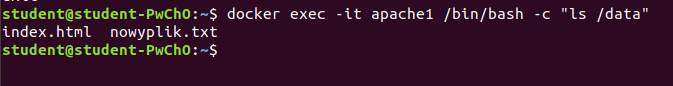
2)

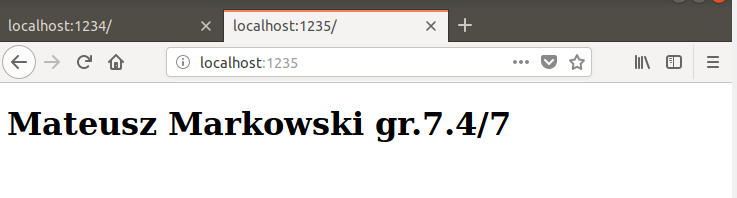




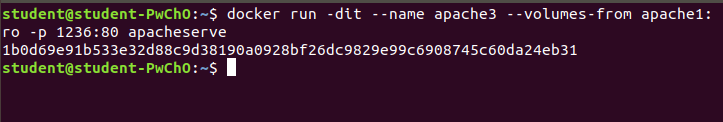


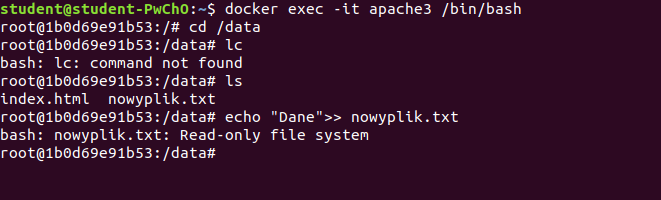


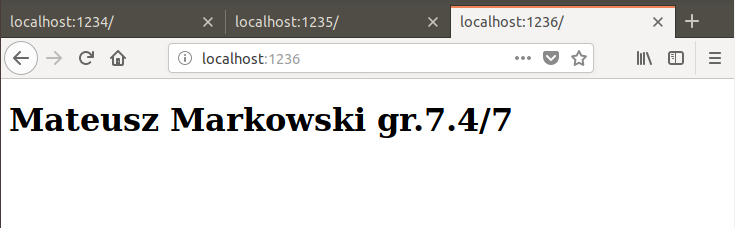




Utworzono nowy kontener na bazie dockerfile zawierającego obraz apache nazwano go apache2 wystawiono go na porcie 1235. Podmonotwano ten sam volumen co w przypadku pierwszego kontenera apache1. Dodatkowo w celu potwierdzenia, że kontener apache2 ma możliwość nadpisywania volumenu apacheVol. Poleceniem docker exec uruchomiono kontener oraz w folderze /data utworzono nowyplik.txt. W celu potwierdzenia, że operacja się udała wyświetlono zawartość katalogu \_data volumenu apacheVol oraz poleceniem docker exec wyświetlono zawartość pierwszego kontenera apache1. Przedstawiają to powyższe zrzuty ekranowe. Wywnioskować z tego można, iż volumen jest współdzielony przez dwa kontenery apache1 oraz apache2 oraz jest w trybie nie tylko do odczytu.







Powyżej przedstawiono utworzenie trzeciego kontenera z serwerem apache tym razem z wykorzystaniem volumenu apacheVol w trybie read-only. Poprawne działanie tego trybu pokazuje środkowy zrzut ekranowy. Gdzie widać komunikat z konsoli bash, iż nie można nic zapisać do pliku nowyplik.txt. Tryb tylko do odczytu.

**Wnioski:**

Najprostszym sposobem stworzenia dockerfile z serwerem apache jest stworzenie prostej strony internetowej html w tym samym folderze, w którym umieszczony jest plik Dockerfile. Podmontowanie volumenu zewnętrznego należy wykonać w momencie uruchamiania kontenera, a nie w trakcie pisania dockerfile. Plik Dockerfile znacząco poprawia sposób tworzenia obrazów. Komendy są prostsze, a co za tym idzie czytelniejsze. W celu dostępu do katalogu, w którym znajduje się volumen należy stworzyć go w Dockerfile, a w trakcie uruchamiania kontenera podłączyć volumen do tego kontenera.

**Zadanie 3**

Uruchom trzy serwery nginx w taki sposób by:

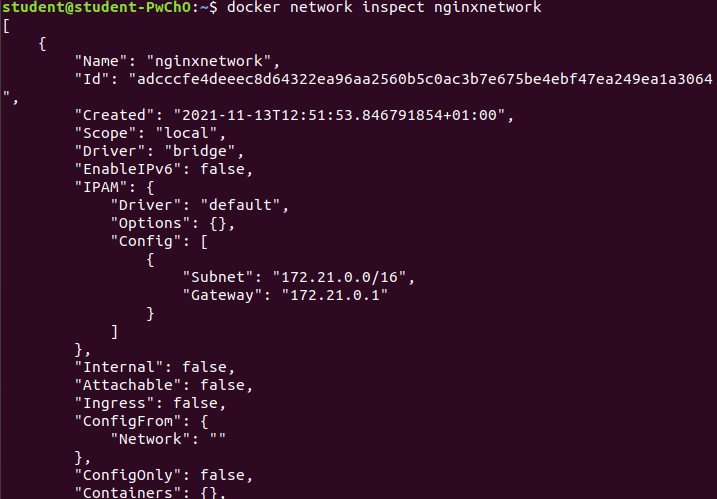
1)wszystkie serwery były podłączone do jednej sieci mostkowej definiowanej przez użytkownika,

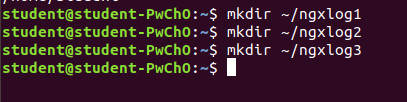
2)wszystkie serwery były dostępne z sieci zewnętrznej,

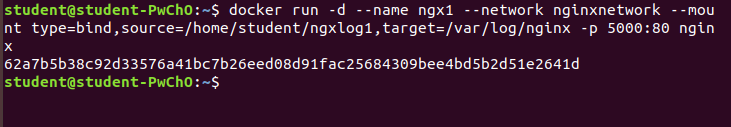
3)poszczególne serwery zapisywały logi w trzech dedykowanych podkatalogach katalogu domowego na systemie macierzystym.

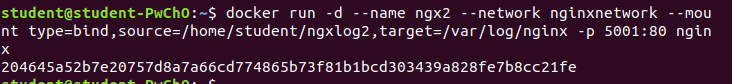
1)

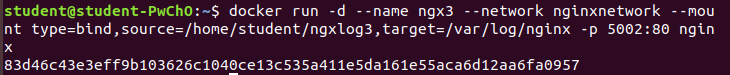










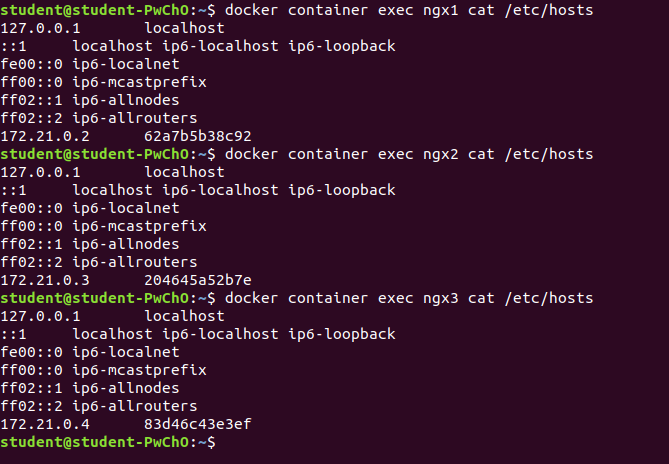


2)

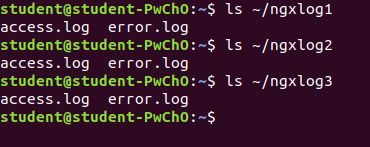


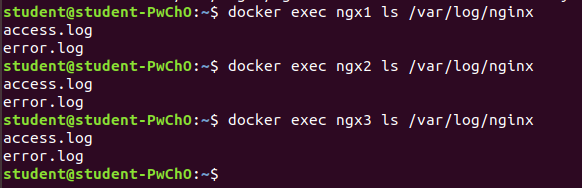






3)





**Wnioski:**

W powyższym ćwiczeniu należało utworzyć 3 kontenery nginx w jednej sieci oraz odpowiednio je sparametryzować. W tym celu na samym początku poleceniem docker create network utworzona sieć, w której potem umieszczono wszystkie kontenery. Stworzono też 3 foldery dla każdego kontenera przechowujące logi o kontenerze. Wykorzystano poleceni mkdir. W celu utworzenia kontenerów w jednej sieci należało użyć polecenia docker run z odpowiednimi parametrami. Wykorzystanie tego polecenia przedstawiano na powyższych zrzutach ekranowych. W celu udostępnienia kontenerów zewnętrznie należało zmapować port :80 na dowolny inny port pod którym serwer nginx będzie odpowiadać. W ćwiczeniu zdecydowano się na porty 5000, 5001 oraz 5002. Poleceniem docker exec sprawdzono czy w kontenerach znajdują się logi takie same jak w systemie macierzystym. Powyższe zrzuty ekranowe przedstawiają że utworzeni kontenerów powiodło się oraz są one dostępne z zewnętrznej sieci. Obrazuje to punkt drugi zadania.

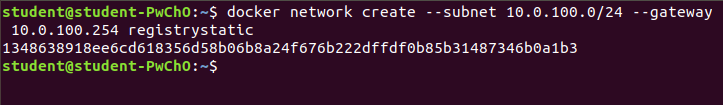
**Zadanie 4**

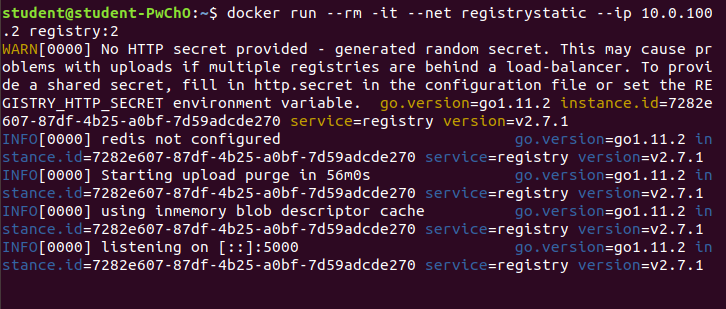
Na jednym z poprzednich ćwiczeń instalowany był prywatny rejestr (registry) na podstawie obrazu https://hub.docker.com/\_/registry/ . Proszę zapoznać się z zawarto-ścią Dockerfile dla najnowszej wersji tego repozytorium. Na bazie tego tego obrazu proszę uruchomić rejestr w taki sposób by:

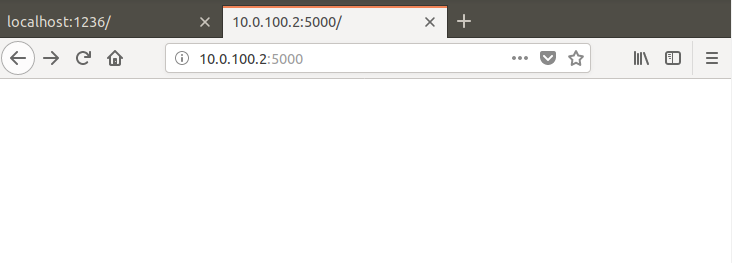
- 1) był on dostępny pod adresem 10.0.100.2 oraz portem 6677,

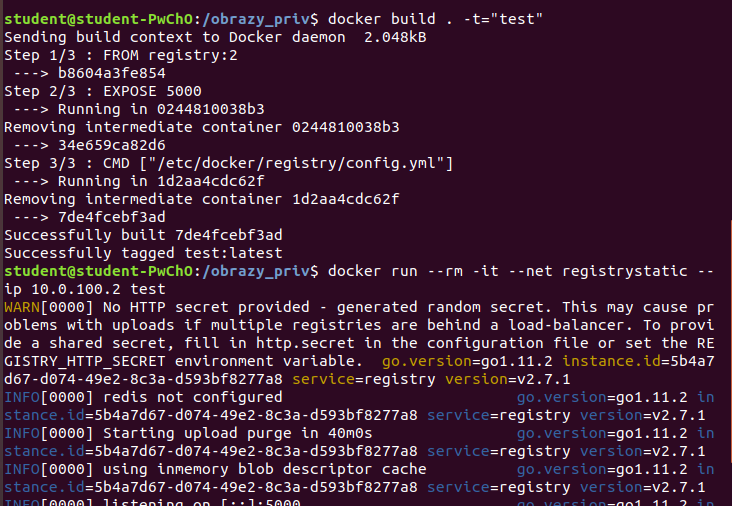
- 2)obrazy były przechowywane w katalogu /obrazy\_priv w systemie macierzystym

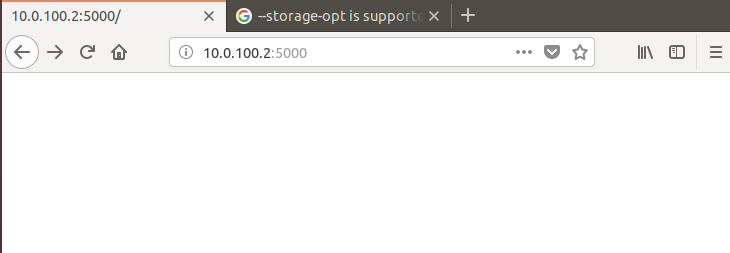
W sprawozdaniu proszę umieścić dowód na poprawne wykonanie tego zadania.

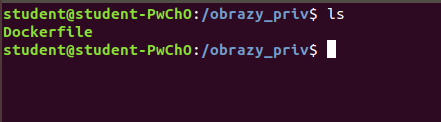














**Wnioski:**

W tym zadaniu udało się stworzyć sieć do której przypisaliśmy kontener dockera. Udało się przypisać registry:2 oraz uruchomić dla adresu 10.0.100.2. Jednak podczas ustawiania portu opcją -p 6677:5000 i uruchamiania w przeglądarce adresu 10.0.100.2:6677 otrzymano błąd. Możliwym jest, że problem ten wynika z braku ustawienia HTTP provided. Informacja o tym znajduje się w logach podczas uruchamiania kontenera. Odnośnie drugiego podpunktu nie wiadomo czy zadanie zostało wykonane poprawnie. Utworzono plik Dockerfile w katalogu obrazy\_priv. Na podstawie tego stworzono kontener dockerowy jak w pierwszym podpunkcie. W dokumentacji docker run --help nie znaleziono odpowiedniego polecenia do ustawiania obrazów w systemie macierzystym.

**Zadanie 5**

Wykorzystując informację z podpunktu D proszę wprowadzić modyfikację w wykorzystaniu zasobów przez kontener o nazwie *limiter* utworzonego na podstawie najnowszej wersji obrazu alpine. Modyfikacje te moją polegać na:

1. ograniczeniu wykorzystywanej pamięci RAM do 512 MB,

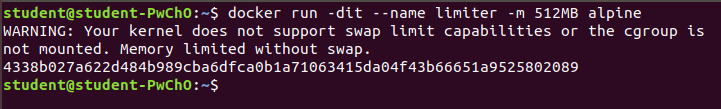
2. ograniczenie działania kontenera do tylko tylko jednego rdzenia.

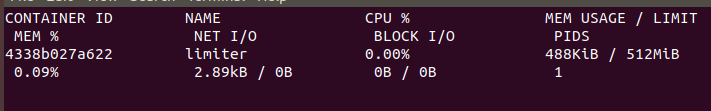
W sprawozdaniu, wykorzystując poznane narzędzia do monitorowania środowiska

Docker (tekstowe jak i graficzne), należy umieścić zrzuty ekranów, które potwierdzają

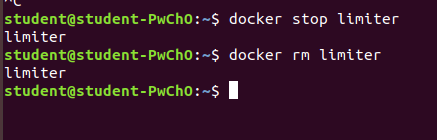
wprowadzenie w życie wymienionych wyżej ograniczeń.

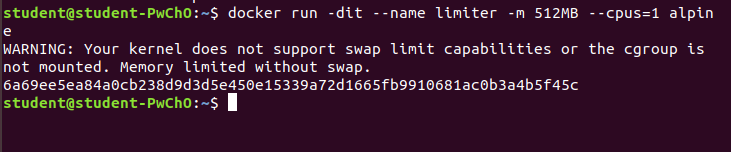
1)





2)





**Wnioski:**

Przed ustawieniem odpowiednich parametrów dla kontenera dockerowego należy sprawdzić czy są możliwe włączenia ustawienia pamięci. Ustawienie pamięci na wartość 512MB działa poprawnie co pokazuje polecenie docker stats limiter. W przypadku polecenia z parametrem --cpus=1, które ustawia ilość działających procesorów jest inaczej. Polecenie nie wyrzuca błędu jednak informacja o ilości uruchamianych procesorów nie pojawia się po wpisaniu polecenia docker stats limiter. Możliwe więc że przyczyną tego jest to, że kontener zbudowany na bazie obrazu alpine nie wykonuje czynności, które akurat wymagają użycia procesora. Możliwe jest, że wszystko działa poprawnie jednak informacja o ilości ustawionych procesorów znajduje się w systemie po wpisaniu innego polecenia. Dokumentacja opcji uruchomieniowych kontenera mówi jednak, że polecenie cpus powinno ustawić ilość potrzebnych procesorów.