**Sprawozdanie Laboratorium 2**

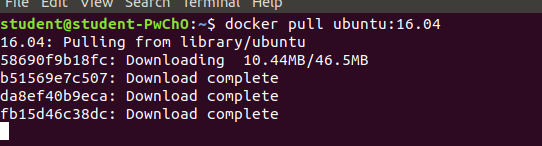
**Mateusz Markowski gr. 7.4/7**

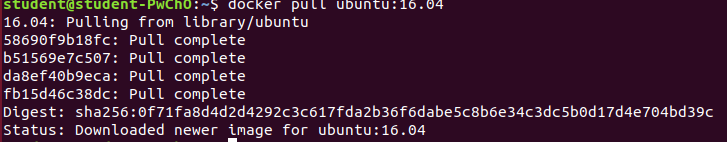
**Zadanie 1**

Proszę przeszukać i pobrać bazowy obraz dla dystrybucji ubuntu:16.04.

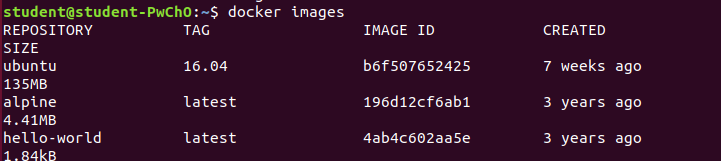
Następnie sprawdzić, że jest on dostępny w lokalnym środowisku Docker. W

sprawozdaniu umieścić zrzuty ekranowe użytych poleceń i wyniki ich działania.





Do pobrania obrazu wykorzystałem polecenie docker pull ubuntu:16.04. Polecenie to zakończyło się pobranie obrazu Ubuntu z bazy dostępnych obrazów znajdujących się w repozytorium DockerHub. W celu sprawdzenia czy obraz został pobrany wykorzystałem polecenie docker images, którego rezultat przedstawiłem na poniższym zrzucie ekranowym. Wynika z niego, iż obraz ubuntu został pobrany.



**Zadanie 2**

Uruchom kontener na bazie obrazu Fedora w najnowszej wersji (lub innej wersji

Linux-a niż Ubuntu), który wypisze na konsoli informacje o dystrybucji i tym samym

potwierdzi, że jest to inna dystrybucja niż Ubuntu, na którym działa środowiskoDocker. W sprawozdaniu umieść zrzuty ekranowe zawierające wszystkie użyte polecenia wraz z wynikiem ich działania.



Wykonanie tego zadania zostało zrealizowane za pomocą jednej komendy. W poleceniu można było zastosować wiele komend, ale uważam, że zrobienie tego jednym poleceniem było zdecydowanie prostsze. W tym celu wykorzystałem polecenie docker run -it --name fedora\_test fedora /bin/bash -c ”cat /etc/os-release ”.

Polecenie docker run odpowiedzialne jest za uruchomienie kontenera jeśli nie zostanie znaleziony obraz systemu nastapi jego automatyczne pobranie z DockerHub’a. Wystarczyło podać nazwę obrazu fedora, bez konkretnej wersji dlatego, że domyślnie pobierana jest zawsze najnowsza wersja obrazu. Parametry -it pozwoliły mi na uruchomienie kontenera w trybie interaktywnym tzn. spowodowało to możliwość wejścia do powłoki bash. Ważne jest aby dopisać /bin/bash jest to komenda, która zostanie przyjęta przez konter. Dodanie flagi -c powoduje możliwość wydania dodatkowego polecenia dla kontenera. W tym przypadku polecenie cat /etc/os-release powoduje wyświetlenie informacji o wersji systemu. Rzeczywiście zostało pokazane, że kontener został zbudowany na bazie obrazu Fedora. Flaga --name posłużyła w celu nadania konkretnej nazwy dla obrazu. Łatwiej potem będzie go usunąć.

**Zadanie 3**

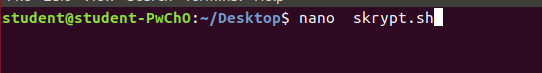
Stwórz a następnie uruchom w tle kontener na bazie obrazu ubuntu:latest.

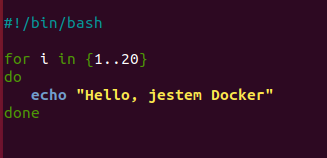
Kontener ma w pętli wyświetlać 20 razy tekst ”Hello, jestem Docker”. Sprawdź

działanie poleceń przedstawionych powyżej (*docker ps*, *docker log* oraz *docker*

*exec*). Zatrzymaj kontener a następnie go usuń. W sprawozdaniu umieść zrzuty

ekranowe zawierające wszystkie użyte polecenia wraz z wynikiem ich działania.



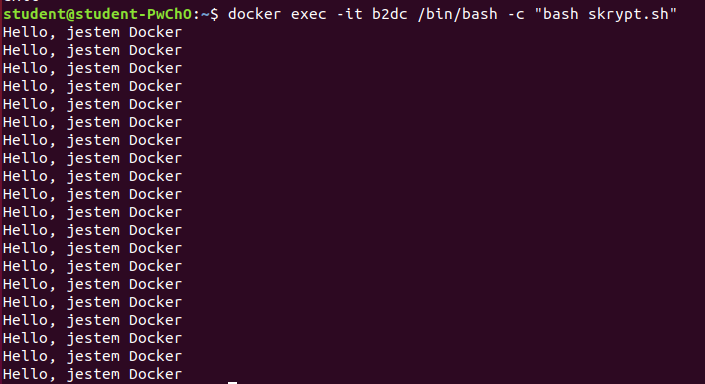


Najpierw stworzyłem skrypt za pomocą polecenia nano skrypt.sh. W skrypcie tym utworzyłem pętle for wyświetlającą tekst „Hello, jestem Docker” dwadzieścia razy. Skrypt został przedstawiony na powyższym rysunku.





Polecenie docker run -itd ubuntu stworzyłem kontener na bazie ubuntu. Następnie do stworzonego kontenera o id b2dc przeniosłem plik ze skryptem utworzynm wcześniej.



Następnie wydałem polecenie docker exec -it b2dc /bin/bash -c „bash skrypt.sh”. Pozwoliło to na uruchomienie skryptu co prezentuje powyższy zrzut ekranu.

**Zadanie 4**

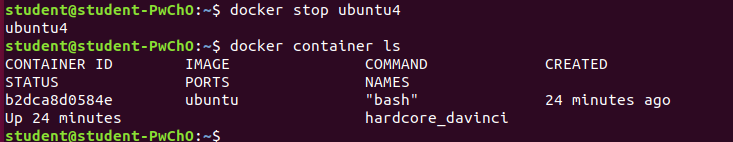
Na podstawie wybranego przez siebie obrazu, zilustruj kolejnymi poleceniami

wszystkie etapy przedstawione na rysunku powyżej. Polecenia te przyporządkuj do

konkretnego etapu i przedstaw wraz z wynikiem ich działania w sprawozdaniu.

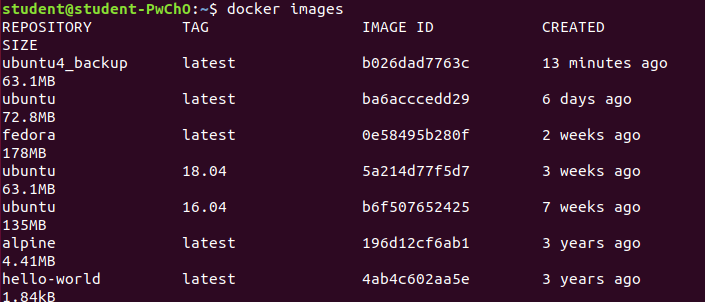


Etap pierwszy oraz drugi został zaprezentowany przez stworzenie oraz nazwanie kontenra na bazie obrazu ubuntu 18.04.

Etap trzeci został zrealizowany przez wykonanie polecenia docker stop ubuntu4.







Etap czwarty został przedstawiony na dwóch powyższych zrzutach ekranowych. Widać że utworzenie kontenera powiodło się. Zostało wygenerowane id kontenera.

**Zadanie 5**

Na podstawie przedstawionego opisu i dokumentacji Docker należy:

1. Pobrać i uruchomić kontener registry w najnowszej wersji.

2. Wgrać do niego obraz o nazwie mutant (utworzony w wyniki polecenia docker

commit w poprzednim zadaniu) i nadać mu tag „nowy”.

3. Pobrać z lokalnego registry obraz „localhost:5000/mutant.nowy” i sprawdzić jego

dostępność w środowisku lokalnym.

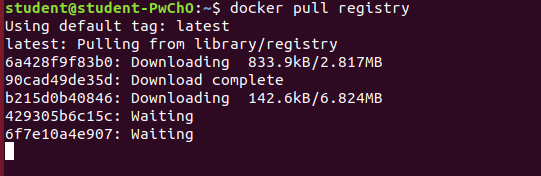
4. Zatrzymać kontener pełniący funkcję registru. Usunąć go oraz usunąć obraz

registry:2 oraz zgromadzone przez niego dane.Wszystkie punkty powinny być w sprawozdaniu zilustrowane: użytym poleceniem,

wynikiem jego działania oraz (tam gdzie to celowe, komentarzem).

1.

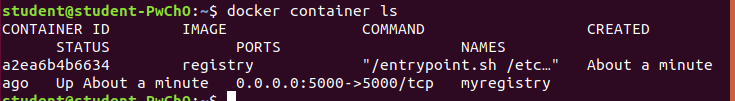
Aby pobrać kontener registry wydałem polecenie polecenie docker pull registry. Brak wersji przy registry oznacza, że pobierze się najnowsza wersja registry.



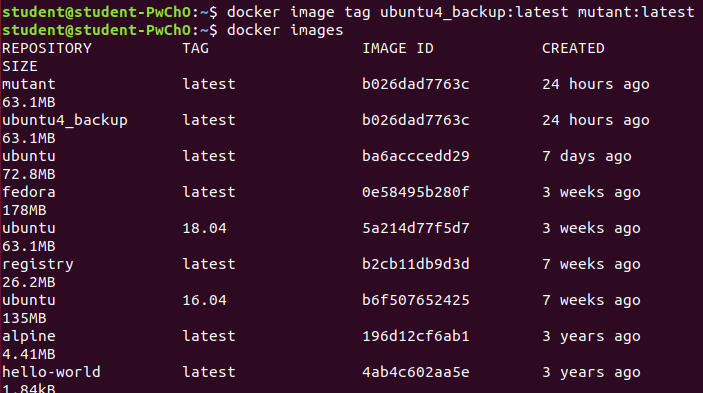


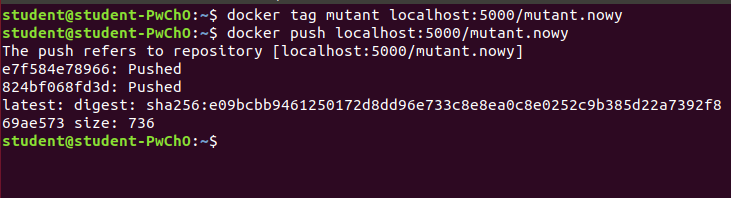
Kontener registry został uruchomiony przez wydanie polecenia docker run -d -p 5000:5000 --name myregistry registry. Sprawdzenie czy kontener rzeczywiście został uruchomiony sprawdziłem przez wydanie polecenia docker container ls.

Wynik działania pokazuje, że kontener został uruchomiony, a jego status to UP.

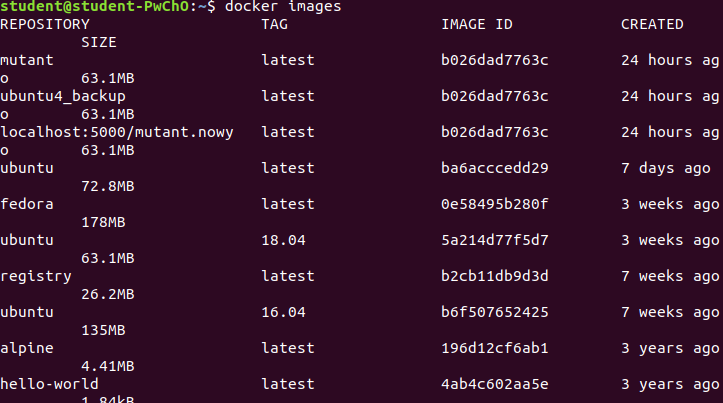


1. Jako, że w poprzednim zadaniu utorzyłem obraz z commita i nazwałem go ubuntu4\_backup. To pierwszym poleceniem jakie wykonałem w tym zadaniu to wydanie polecenia docker image tag ubuntu4\_backup:latest mutant:latest. Dzięki temu poleceniu zmieniłem nazwę obrazu utworzonego w zadaniu 4.

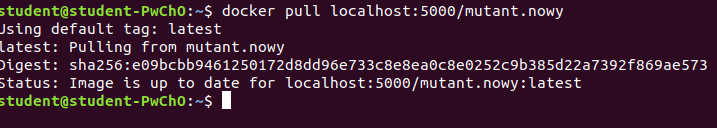


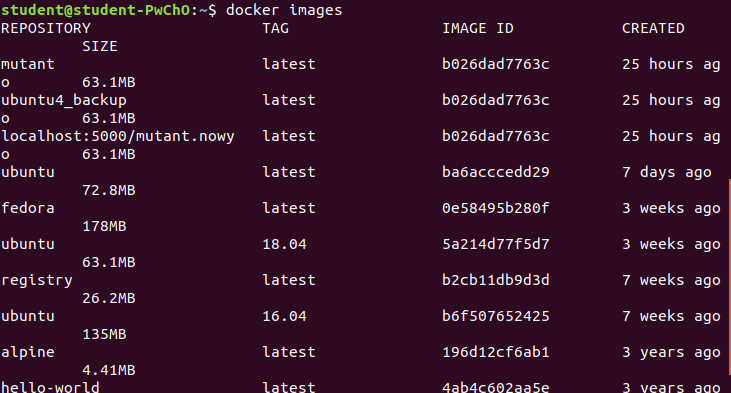


Obraz mutant zmieniłem na obrazu mutant.nowy wykonując to przez polecenie docker tag. Użyłem tego polecenia, ponieważ w systemie nie może być dwóch obrazów o takiej samej nazwie. Przesłanie obrazu mutant do lokalnego registry zrealizowałem wykonując polecenie docker push localhost:5000/mutant.nowy.



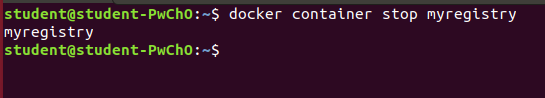
3.

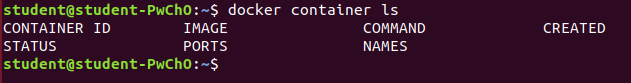


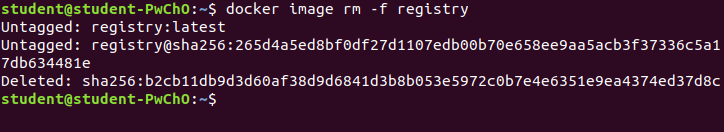


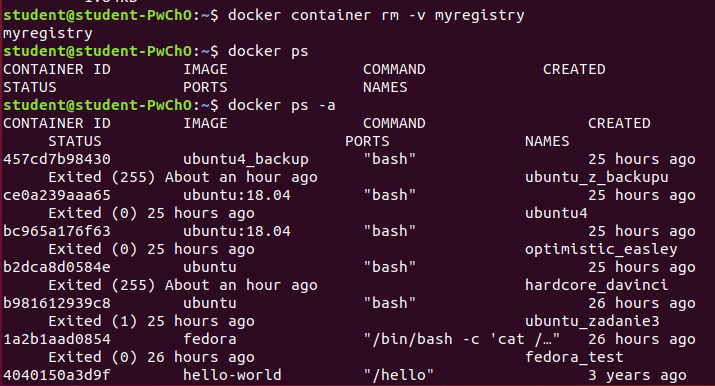
Obraz pobrałem z registry używając polecenia docker pull localhost:5000/mutant.nowy. Na pierwszym z dwóch wyżej zamieszczonych zrzutach ekranowych widać dwa komunikaty informujące o pobieraniu obrazu z mutant.nowy oraz to że jest aktualny z tym, który znajduje się lokalnie na maszynie hosta.

4.









Kontener zatrzymałem wydając polecenie docker container stop myregistry. Następnie wydałem polecenie docker container ls, wynik tego polecenia przedstawiono na drugim zrzucie ekranowym. Widać, że nie ma już żadnego działającego kontenera. Kontener usunąłem wydając polecenie docker registry rm -v myregistry. Potwierdzenie, że kontener został usunięty widać po tym jak wykonałem polecenie docker ps -a. Nie znajduje się tam kontener myregistry. Obraz registry usunąłem wydając polecenie docker image rm -f registry. Ważne jest dodanie flagi -f, ponieważ z obrazu korzystają inne kontenery. Flaga minus f pozwala na bezproblemowe usunięcie obrazu.

**Zadanie 6**

Proszę wykonać następujące zadania i odpowiedzieć na pytania:

1. Pobrać z Docker Hub obraz „busybox” w najnowszej wersji.

2. Zapisać ten obraz jako plik „alfa.tar”.

a. Gdzie domyślnie utworzony plik „alfa.tar” został zapisany ?

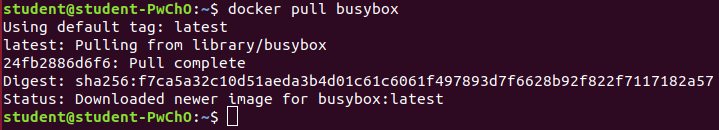
b. Czy ten plik można samodzielnie rozpakować a jeśli tak to co on zawiera ?

3. Usunąć obraz „busybox” z lokalnego środowiska Docker.

4. Załadować obraz „busybox” z pliku „alfa.tar” i sprawdzić czy jest on dostępny w

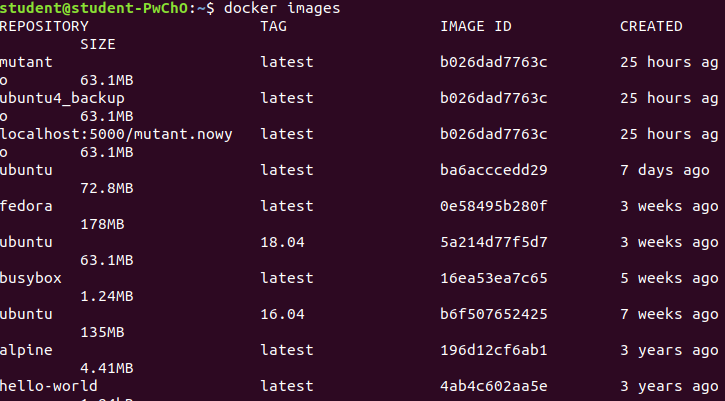
lokalnym środowisku Docker.

1.

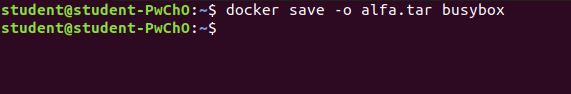


Pobranie obrazu busybox zostało zaprezentowane na powyższym zrzucie ekranowym.

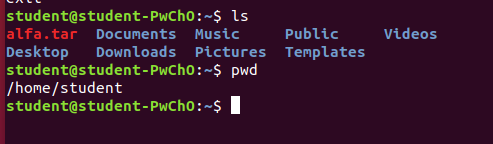
Poleceniem docker images wyświetlono dostępne obrazy co prezentuje poniższy zrzut ekranowy.



2.

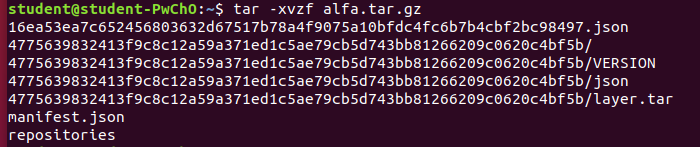


a)



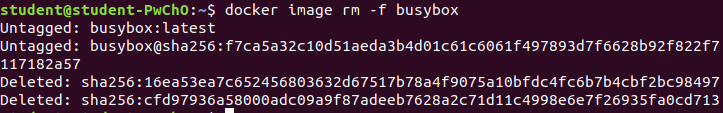
Domyślnie pliki tar przy używaniu polecenia docker save zapisywane są w tym miejscu z którego wykonano polecenie docker save. W moim przypadku plik ten znajduje się w ścieżce /home/student. Związane jest to z tym, że polecenie docker save wykonałem właśnie znajdując się w tej ścieżce.

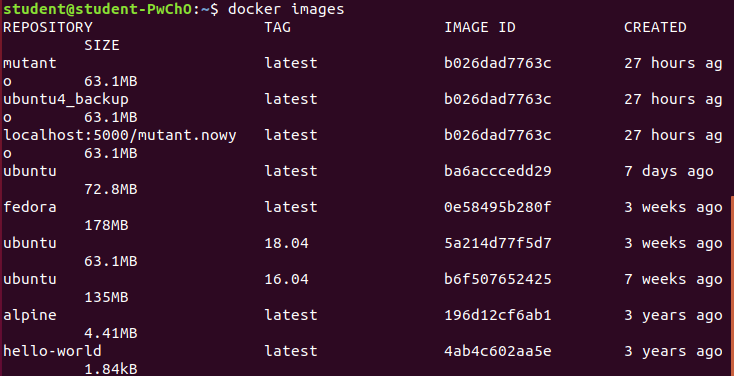
b)



Odpowiadając na pytanie postawione w zadaniu można samodzielnie rozpakować ten plik używając polecenia tar -xvzf. W środku znajdują się pliki, które przedstawiono na powyższym zrzucie ekranowym.

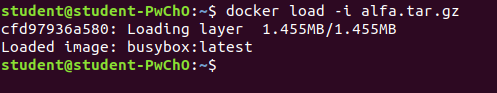
3.

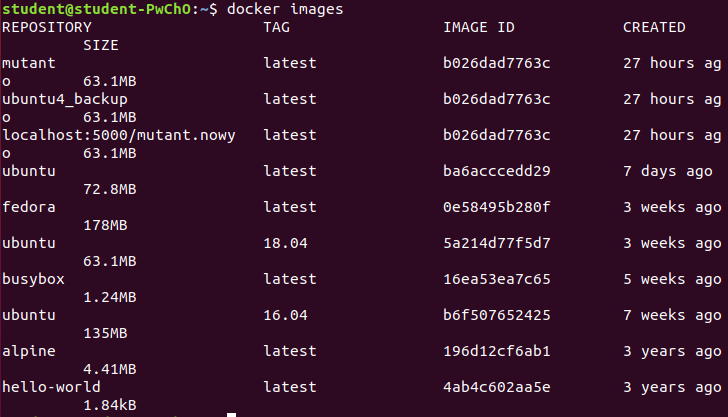




Obraz usunąłem korzystając z polecenia docker image rm -f busybox. Potwierdzenie usunięcia prezentuje drugi zrzut ekranowy.

4.





W celu załadowania obrazu a z pliku tar użyłem polecenia docker load -i alfa.tar.gz. W celu sprawdzenia poprawności załadowania obrazu wykonałem polecenie docker images. Wynik tego polecenia ilustruje powyższy zrzut ekranowy, widać na nim, że obraz busybox z pliku alfa.tar.gz został załadowany.