**===============DOCKER=============**

**Запускаем Nginx в докер**

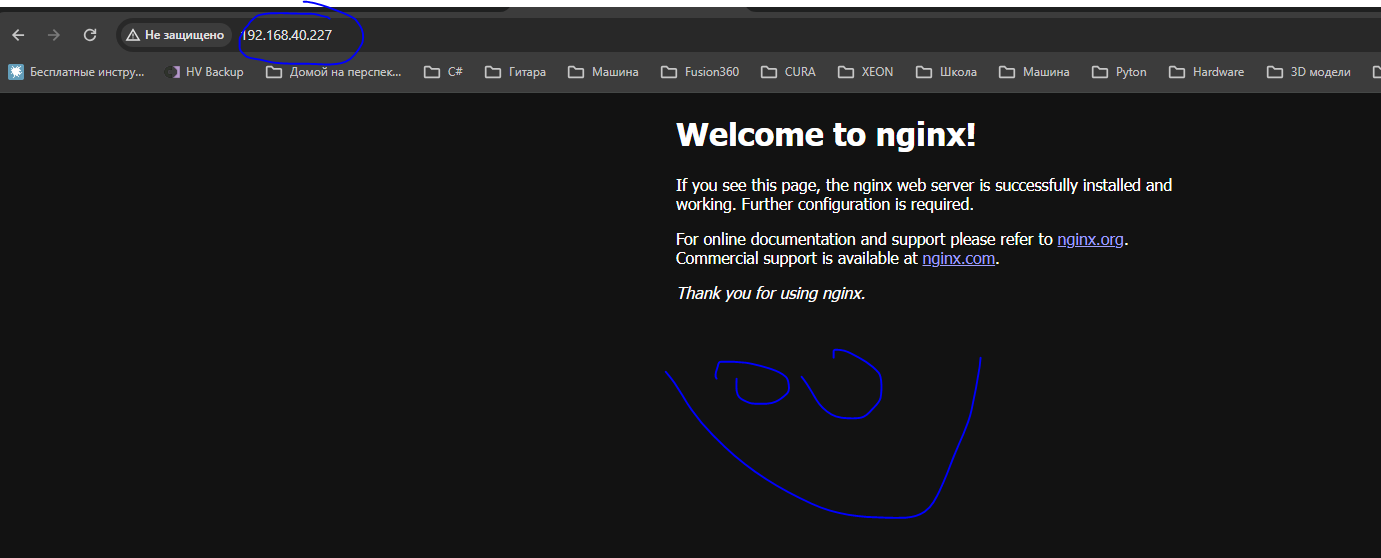
sudo docker run --name my-nginx -p 80:80 -d nginx

Проверяем его работу

curl localhost:80

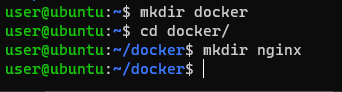
С своего ПК пробуем  
curl 192.168.40.227

Или в браузере

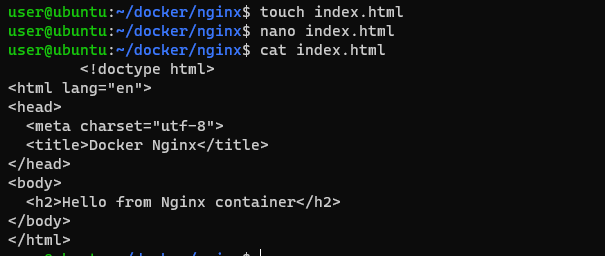


Далее с помощью Docker файла соберём образ и на его основе запустим контейнер. При этом в образе разместим свой index.html

Создадим папку nginx для проекта и перейдём в неё



Создадим наш кастомный index.html



<!doctype html>

<html lang="en">

<head>

<meta charset="utf-8">

<title>Docker Nginx</title>

</head>

<body>

<h2>Hello from Nginx container</h2>

</body>

</html>

Создадим Dockerfile

https://habr.com/ru/companies/ruvds/articles/439980/

Базовый образ — это то, что является исходным слоем (или слоями) создаваемого образа. Базовый образ ещё называют родительским образом.

В файлах Dockerfile содержатся инструкции по созданию образа. С них, набранных заглавными буквами, начинаются строки этого файла. После инструкций идут их аргументы. Инструкции, при сборке образа, обрабатываются сверху вниз

#Образ nginx:alpine очень производителен, хотя занимает лишь около 20 Мб дискового пространства.

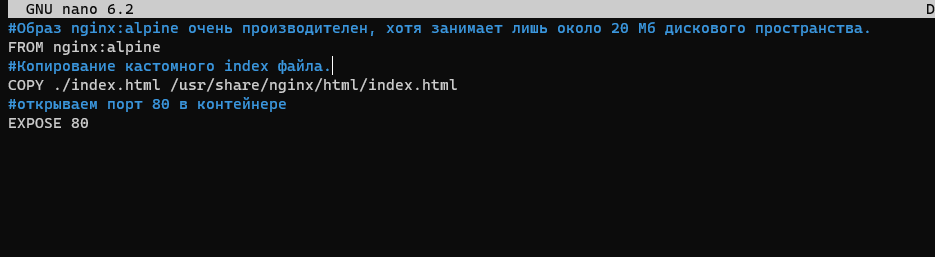
FROM nginx:alpine

#Копирование кастомного index файла.

COPY ./index.html /usr/share/nginx/html/index.html

#открываем порт 80 в контейнере

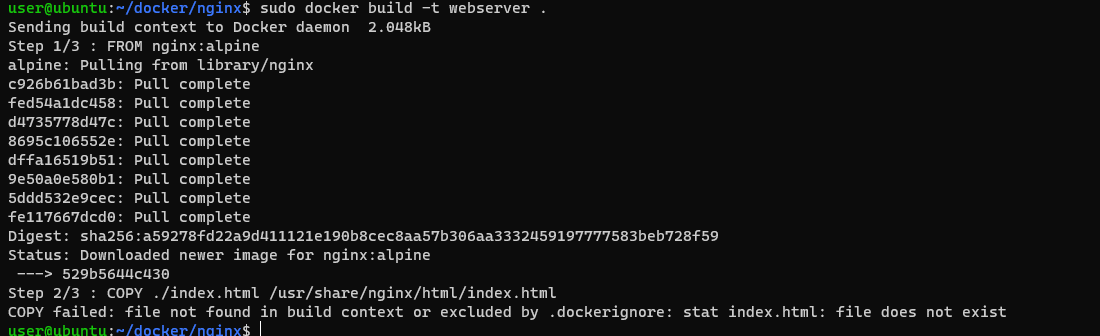
EXPOSE 80



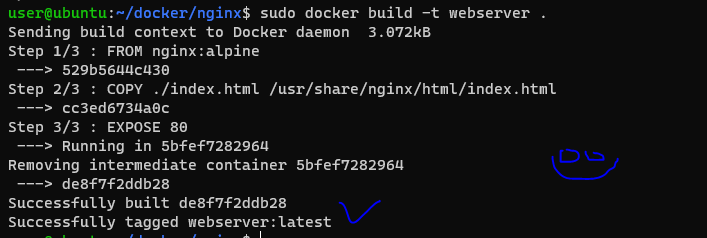


Создаём образ с именем webserver

docker build -t webserver .

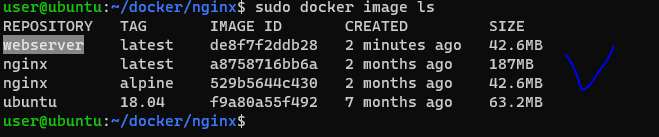


(Тут ошибка, нет index файла)



Проверить список локальных образов

sudo docker image ls



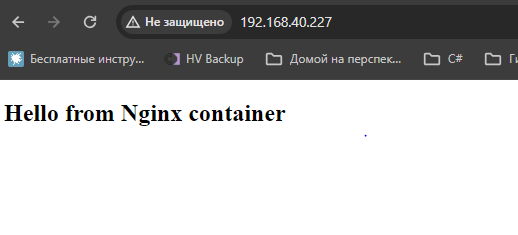
Проверим, какие контейнеры запущены?  
И остановим их по ID



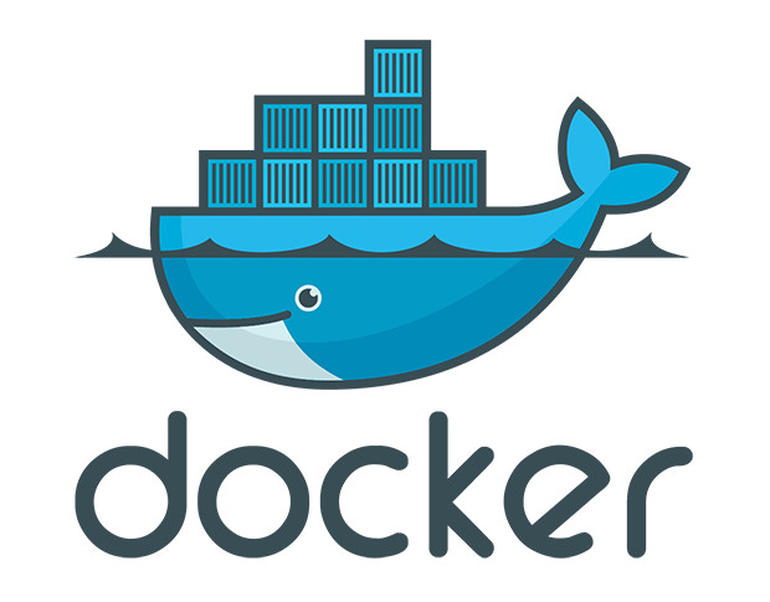
Запускаем контейнер из созданного образа

sudo docker run --name MyCustomNginx -p 80:80 -d webserver

Проверяем его статус и его работу:

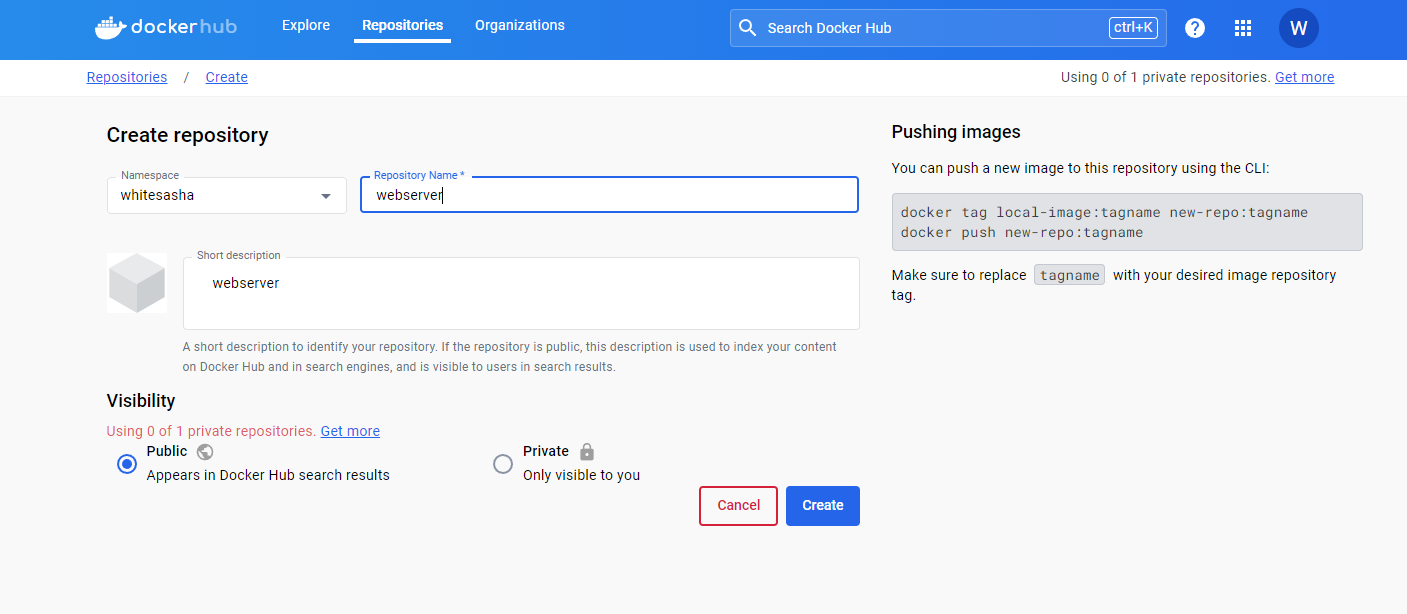


**Публикация образа в Docker registry**

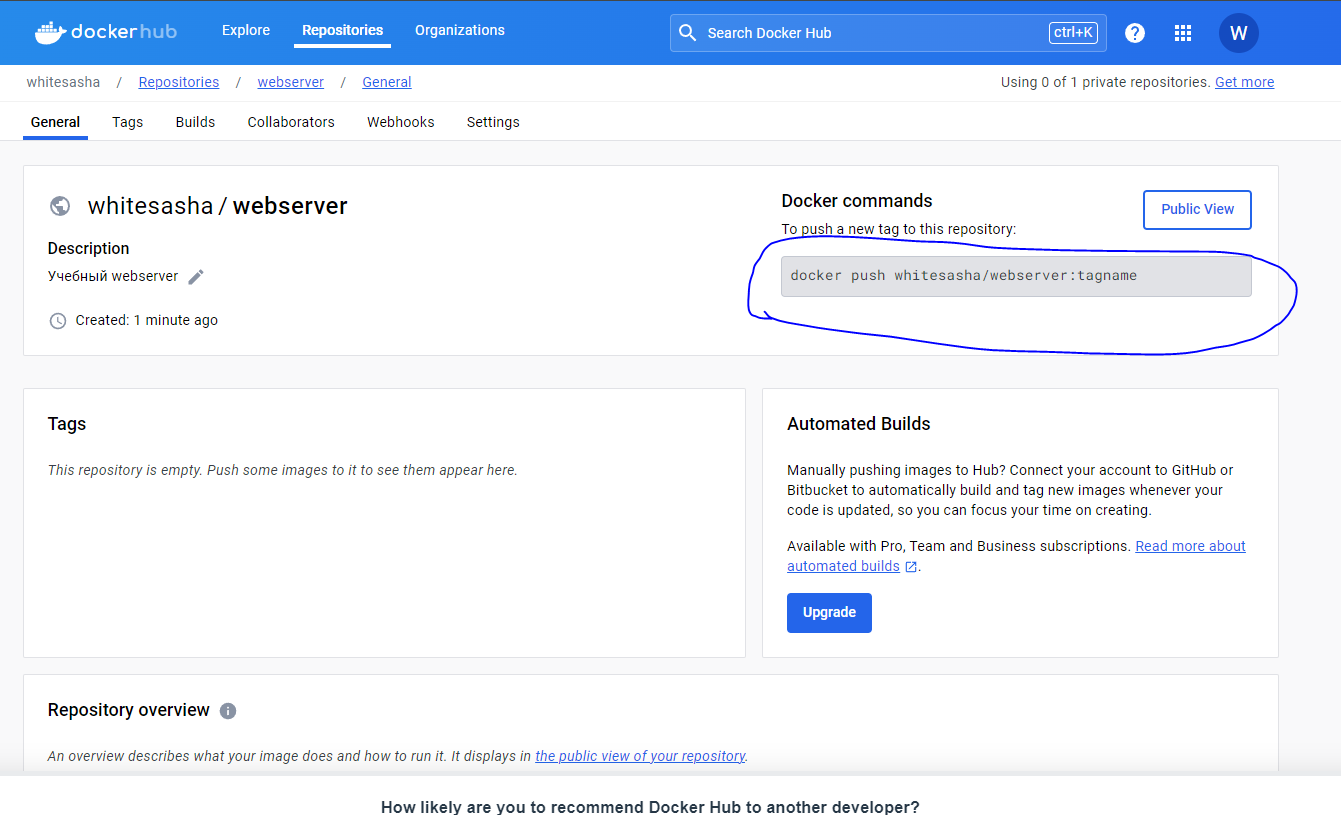


В качестве docker registry будем использовать <https://hub.docker.com/>

Авторизуемся в нём и создаём публичный репозиорий.

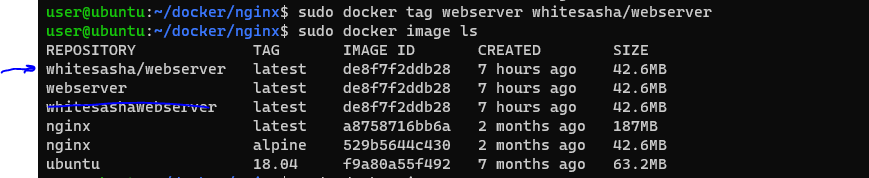


Публикуем созданный образ в репозиторий



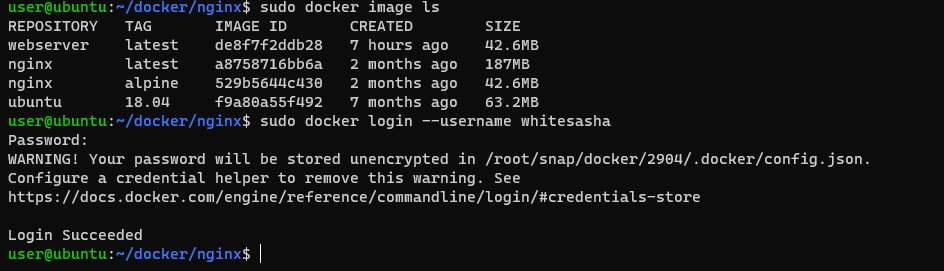
Тегируем образ webserver

sudo docker tag webserver whitesasha/webserver



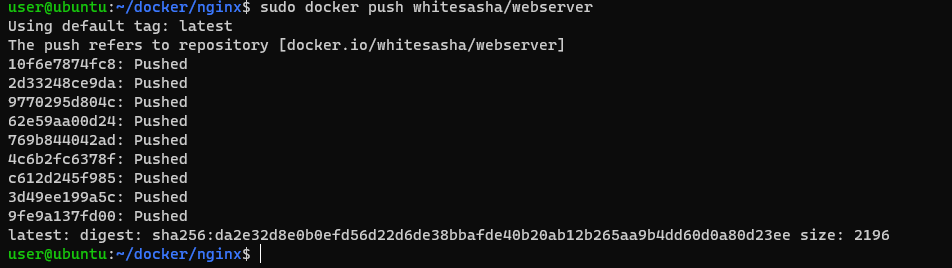
Логинимся докером в нашу учётку

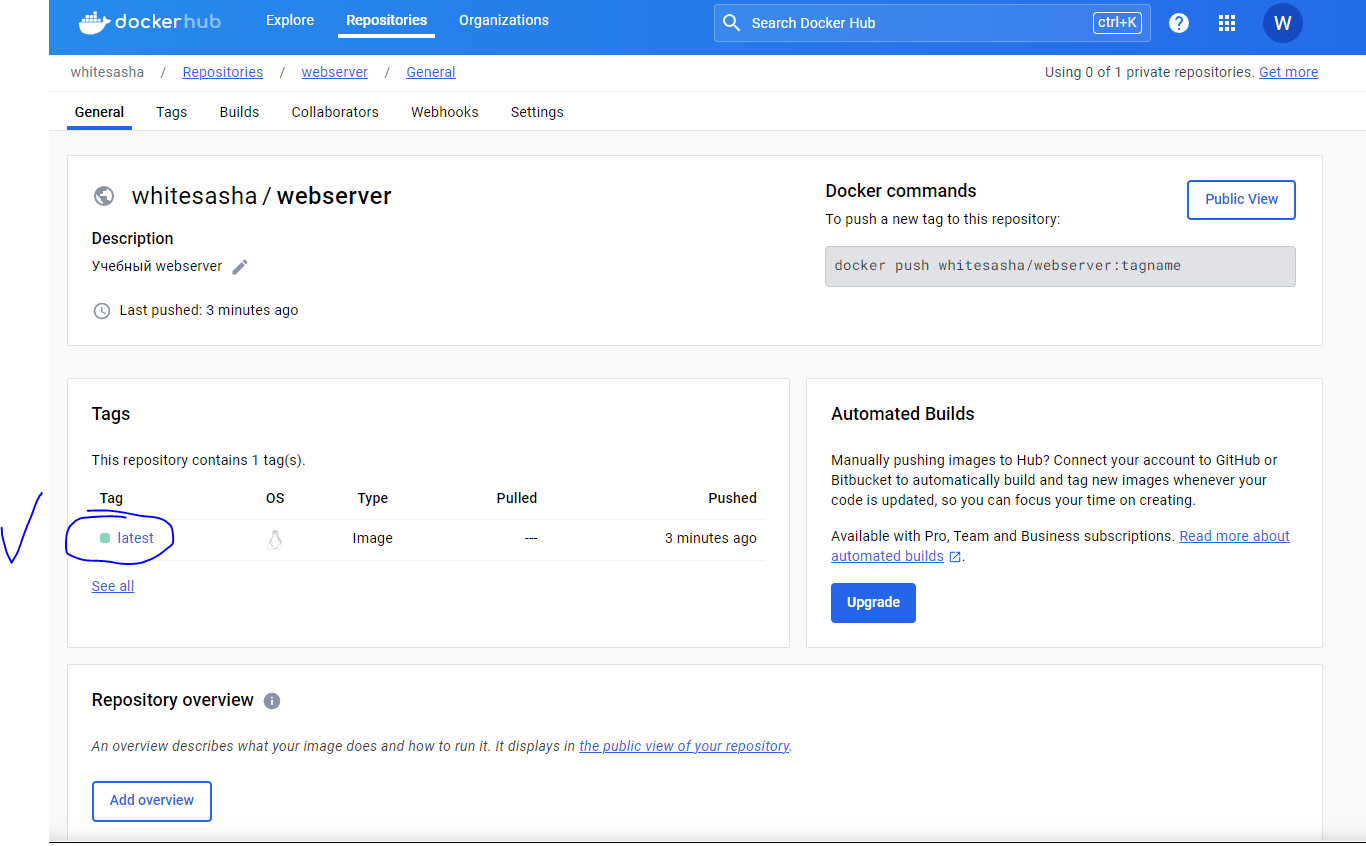
sudo docker login --username whitesasha



Публикуем образ

sudo docker push whitesasha/webserver

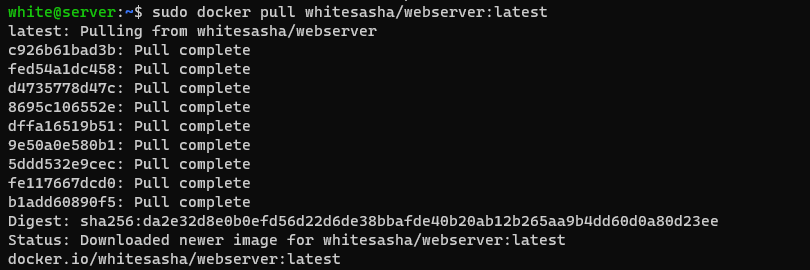


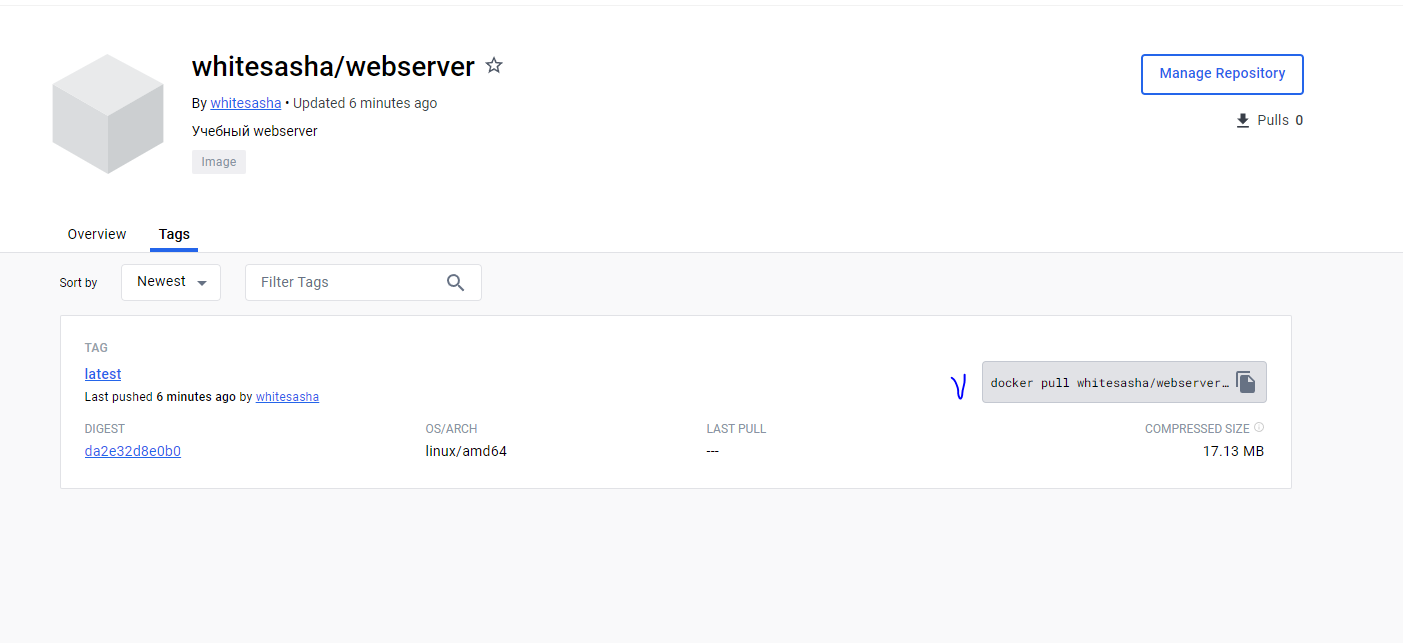


Запускаем контейнер из опубликованного образа на другом сервере

Скачиваем образ из реджистри

Sudo docker pull whitesasha/webserver:latest

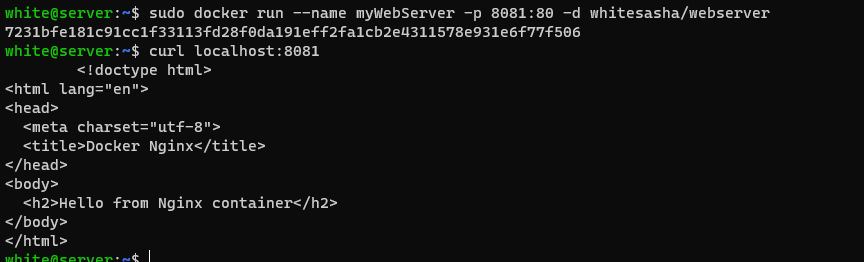




Создаём контейнер из скаченного образа

sudo docker run --name myWebServer -p 8081:80 -d whitesasha/webserver

8081:80 Сопоставляет TCP-порт 80 в контейнере с портом 8080 на узле Docker

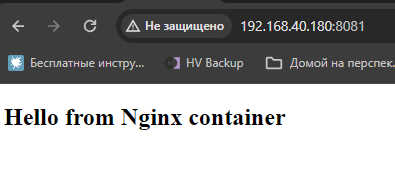


И проверим его работу

При помощи утилиты curl

curl <http://localhost:8081>

B браузере



Удалим контейнер

sudo docker container rm [CONTAINER ID] –f

-f force удалить работающий контейнер