Министерство науки и высшего образования Российской Федерации

федеральное государственное автономное образовательное учреждение высшего образования

«Национальный исследовательский университет ИТМО»

*Факультет программной инженерии и компьютерной техники*

**Лабораторная работа** **3.1**

Группа: P3416

Выполнил: Сиразетдинов А.Н.

Проверил:

к.т.н. преподаватель Белозубов А.В.

Санкт-Петербург

2025г.

Оглавление

[ВЫПОЛНЕНИЕ РАБОТЫ 3](#_Toc214218010)

[Установка гостевой ОС 3](#_Toc214218011)

[Настройка сетевых интерфейсов гостевых ОС 4](#_Toc214218012)

[Создание сети NAT 7](#_Toc214218013)

[Использование собственной сети NAT 8](#_Toc214218014)

[Работа со снимками системы 10](#_Toc214218015)

[Создание общей папки и буфер обмена 15](#_Toc214218016)

[Работа через командную строку 16](#_Toc214218017)

[Заключение 18](#_Toc214218018)

# Выполнение работы

## Установка docker

Для удобства сразу установим docker desktop

sudo apt install docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin

Изображение выглядит как текст, снимок экрана, Шрифт, Графика

Содержимое, созданное искусственным интеллектом, может быть неверным.

## Запуск тестового контейнера



Получаем ошибку вследствие того что образа Hello-sirazetdinov не сущетсвует

## Создание контейнера nginx

Создадим Dockerfile в котором укажем рабочую директорию, откроем порт 8080 и запустим nginx

Изображение выглядит как текст, снимок экрана, Шрифт

Содержимое, созданное искусственным интеллектом, может быть неверным.

Создадим директорию /home/www/html в которой сделаем простой html файл который будет использоваться в nginx

Изображение выглядит как текст, снимок экрана, Шрифт

Содержимое, созданное искусственным интеллектом, может быть неверным.

Соберем контейнер с названием nginx-serve с помощью команды

sudo docker build -t nginx-serve .

Запустим контейнер с помощью команды

sudo docker run -p 8080:80 -v /home/www/html:/usr/share/nginx/html:ro --name my-nginx-serve nginx-serve

Параметры:

* Пробрасываем порт 8080 из порта 80
* Маунтим директорию с html файлом
* Указываем имя контейнера my-nginx-serve

Изображение выглядит как текст, Шрифт, снимок экрана, белый

Содержимое, созданное искусственным интеллектом, может быть неверным.

## Создание контейнера MariaDB

docker run -d \

--name mariadb \

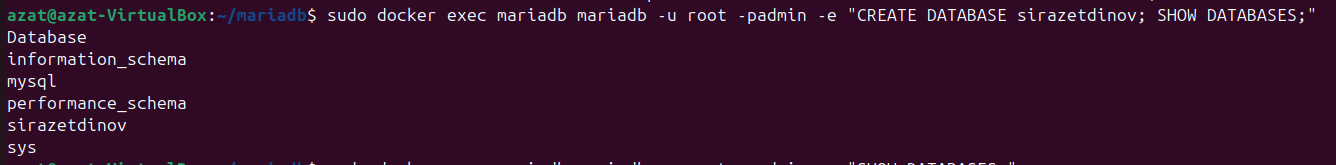
-v /home/DB:/var/lib/mysql \

-e MYSQL\_ROOT\_PASSWORD=admin \

-p 3306:3306 \

mariadb:12.1.1-alpine

Создание БД «sirazetdinov»



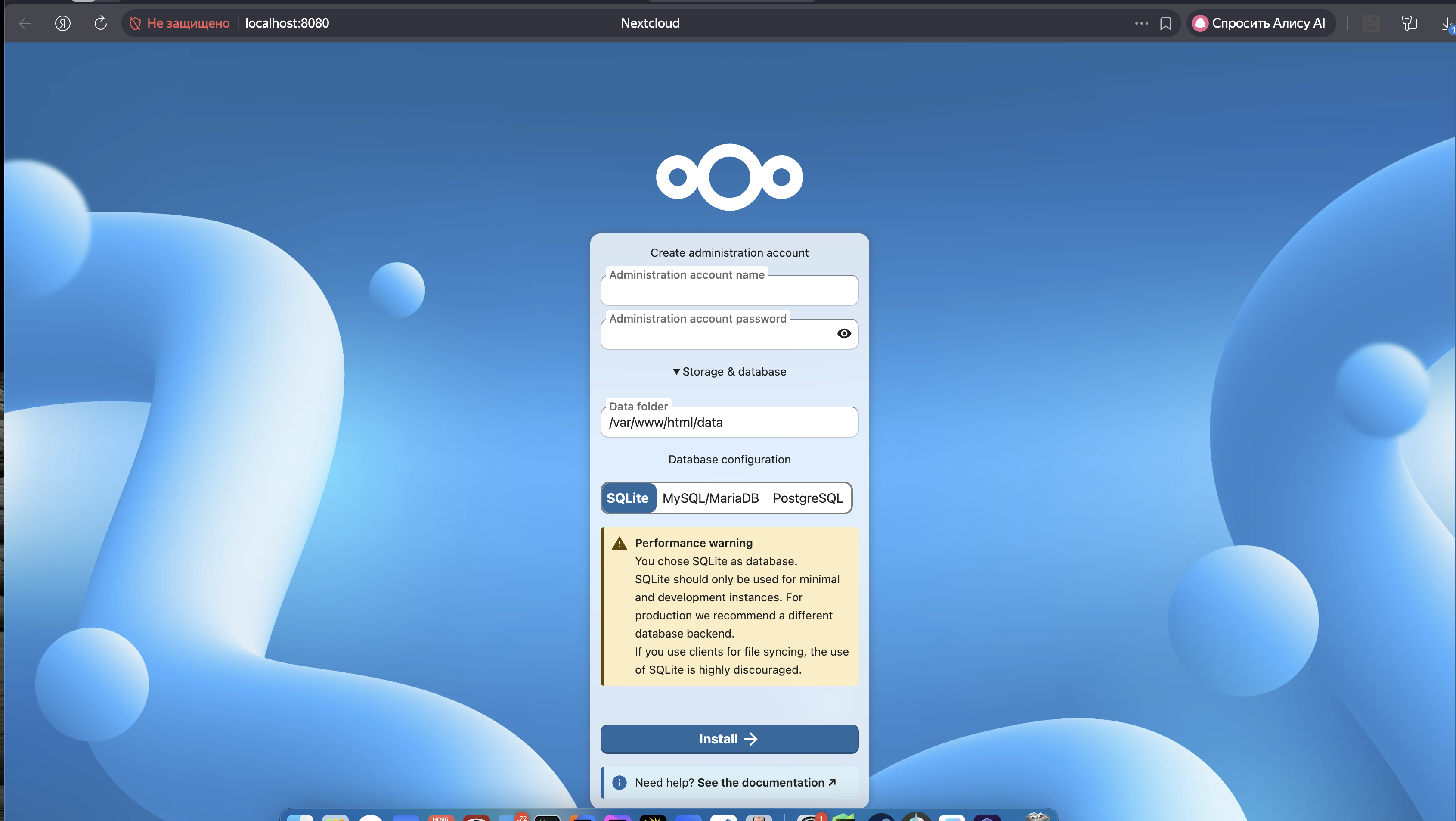
База данных создалась как отдельная директория в /home/DB

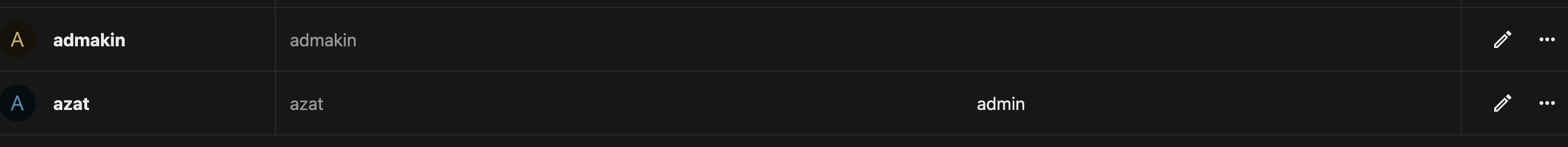


## Создание контейнера NextCloud

С этого момента память на компьютере кончилась, и было очень поздно поэтому я делал дома на локальной машине…

sudo docker run -d --name my-nextcloud -v ./NextCloud:/var/www/html -p 8080:80 nextcloud:latest



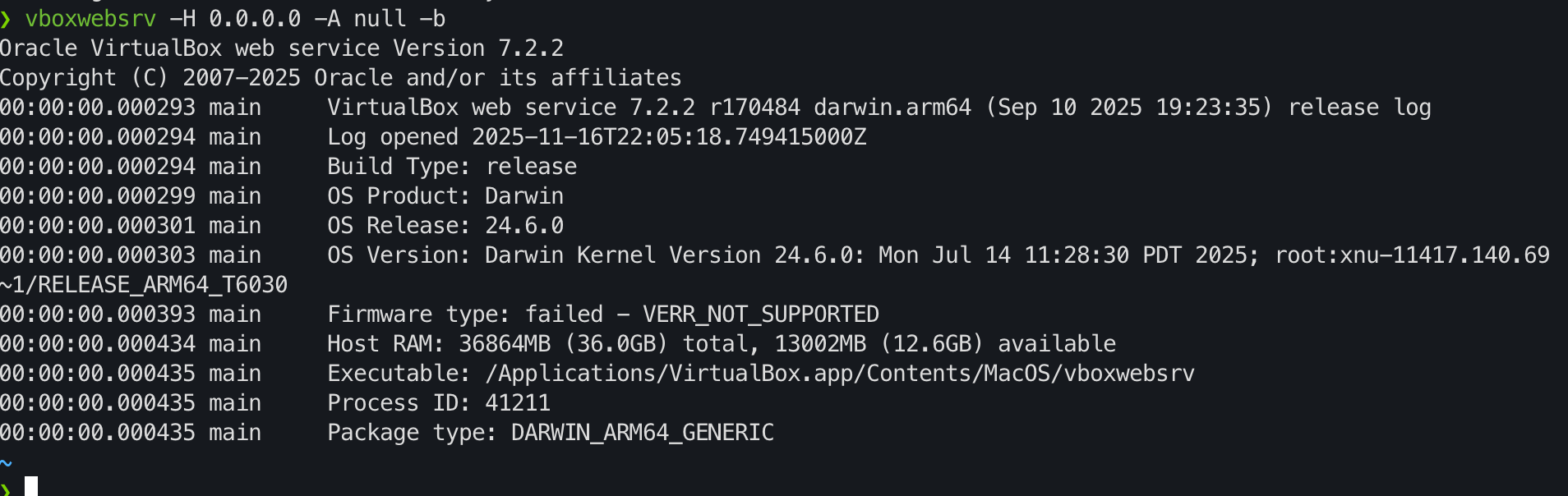


Зарегестрировал два аккаунта для меня и преподавателя

## Запустим phpvirtualbox

Для начала нам нужно запустить сервер для virtualbox

vboxwebsrv -H 0.0.0.0 -A null -b

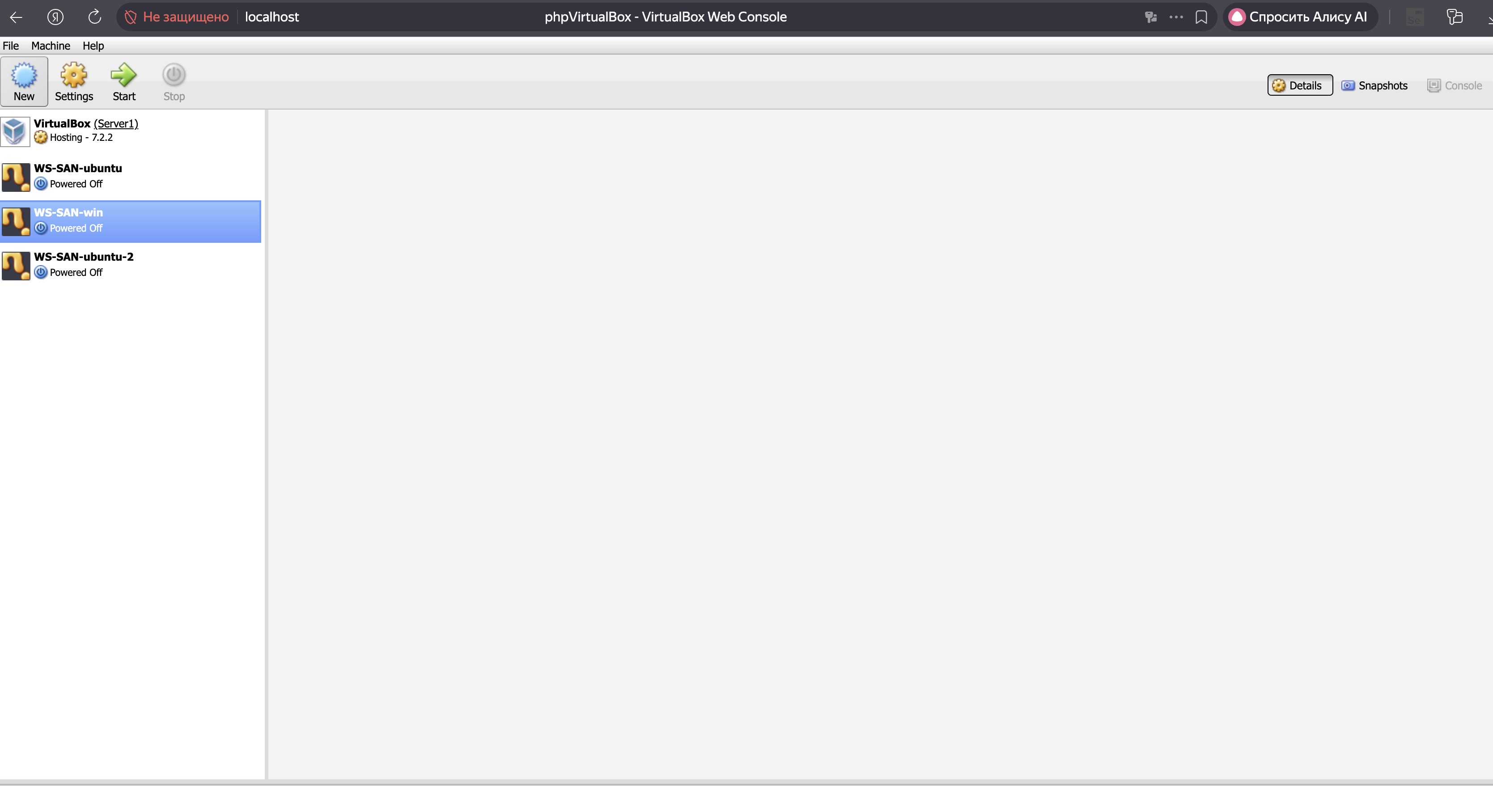


Затем запустим контейнер

docker run --name vbox\_http --restart=always -p 80:80 \

-e SRV1\_HOSTPORT=host.docker.internal:18083 -e SRV1\_NAME=Server1 -e SRV1\_USER=user1 -e SRV1\_PW='test' \

-d joweisberg/phpvirtualbox



К сожалению не существует контейнера который бы поддерживал VirtualBox версии 7.2 (я попробовал три образа)

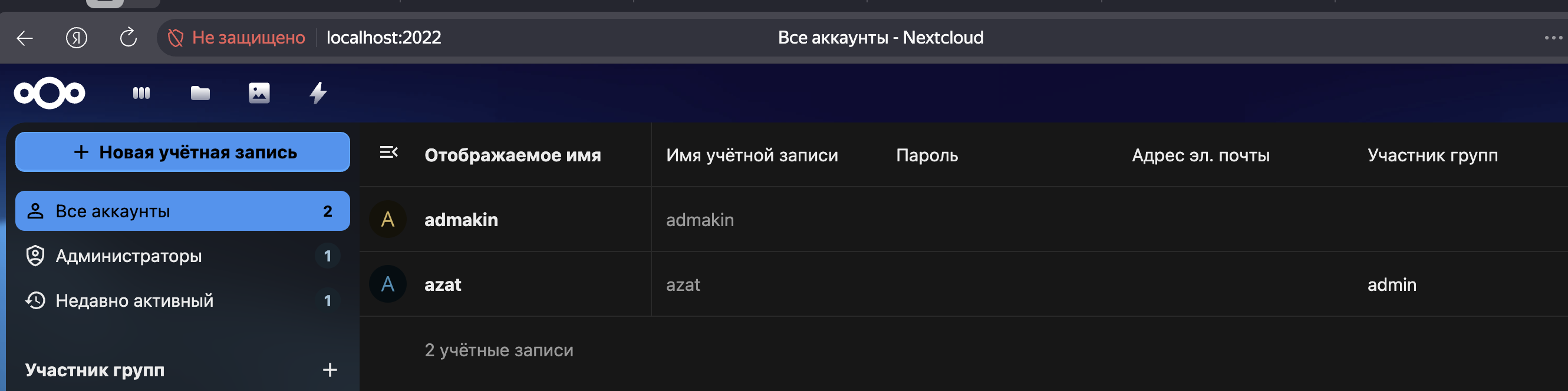
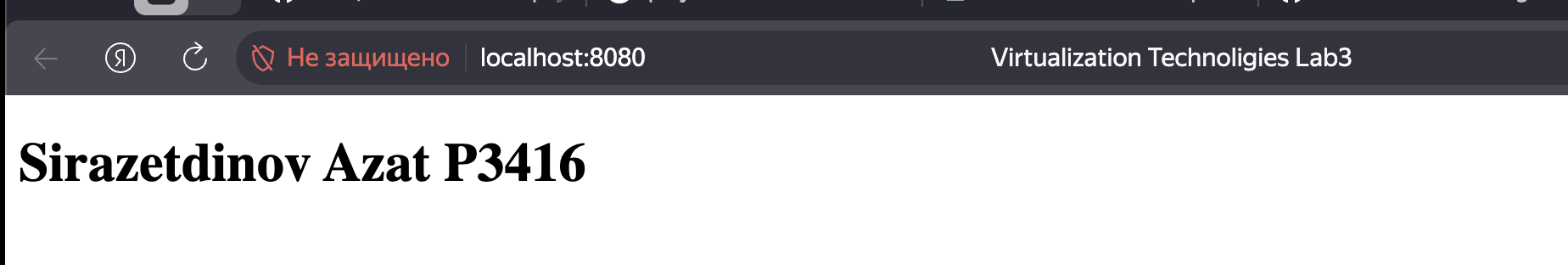
## Настроим проект в docker-compose

Создадим compose файл:

* Настроим сети
* Установим вольюмы на существующие директории
* Прокинем разные порты



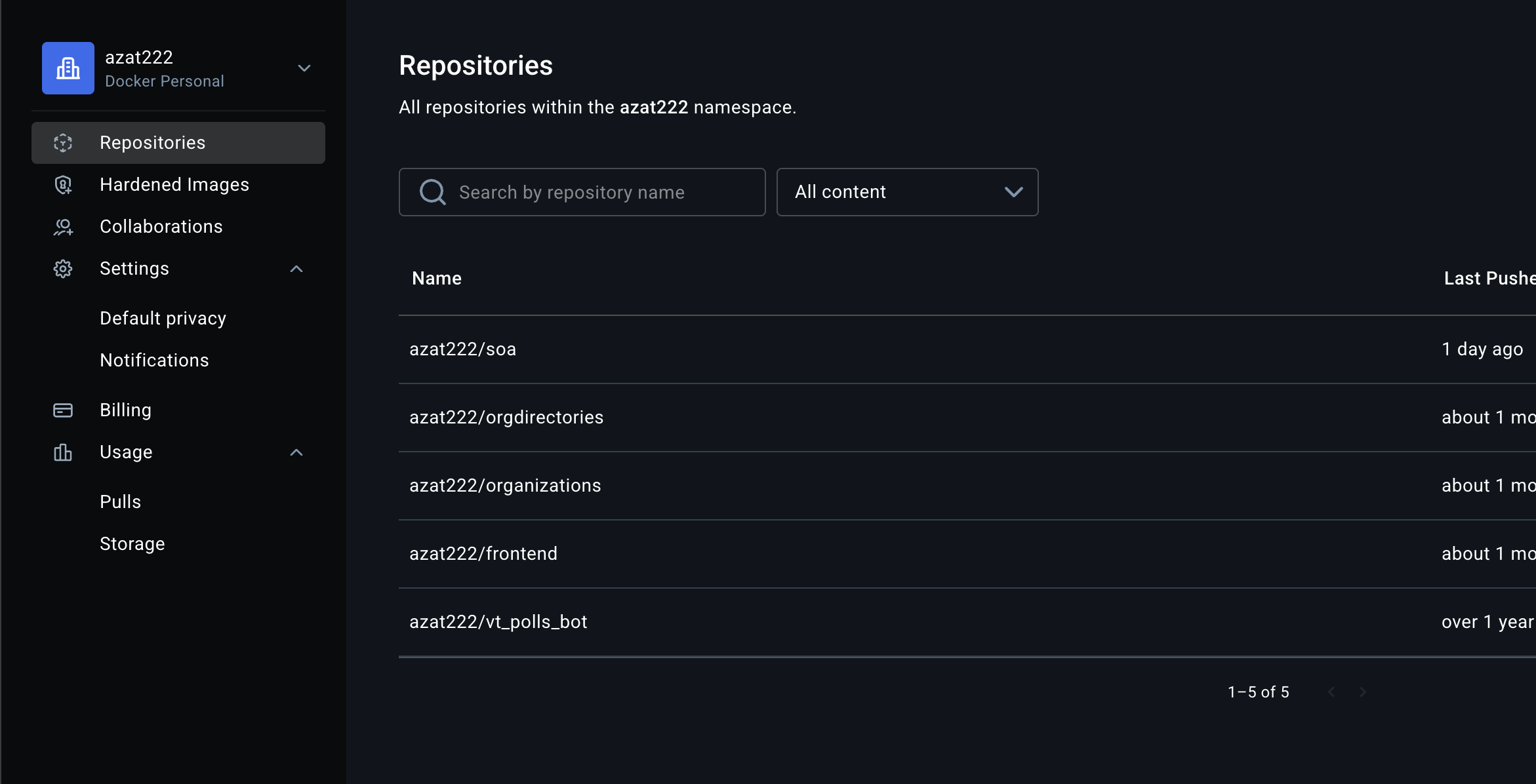
Проверим доступ до сервисов:



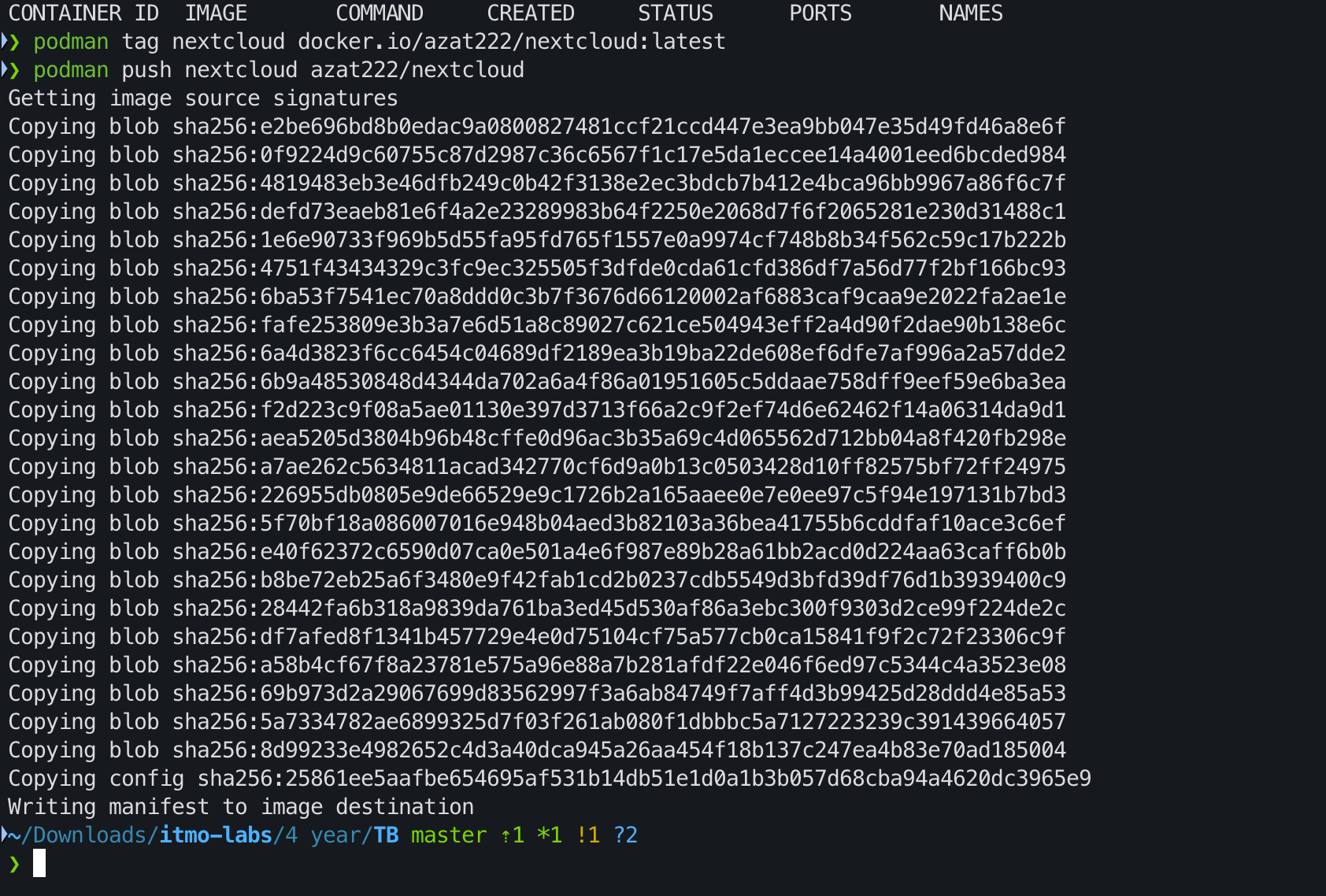
Все сохранилось и работает!

## Зарегестрируемся в docker hub

У меня уже был аккаунт:



## Выложим проект на docker hub

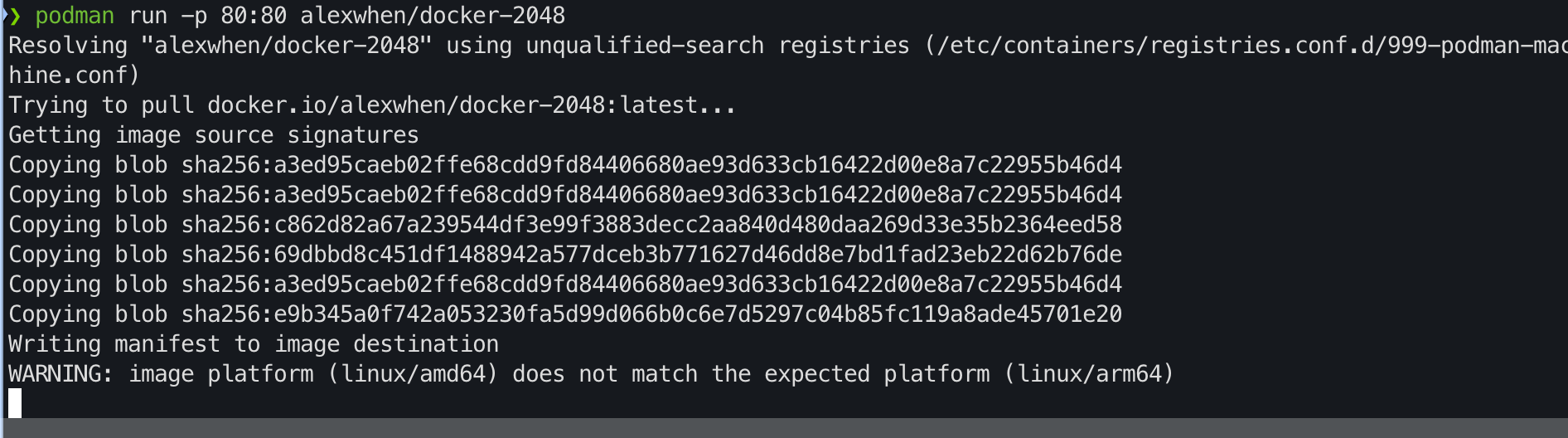


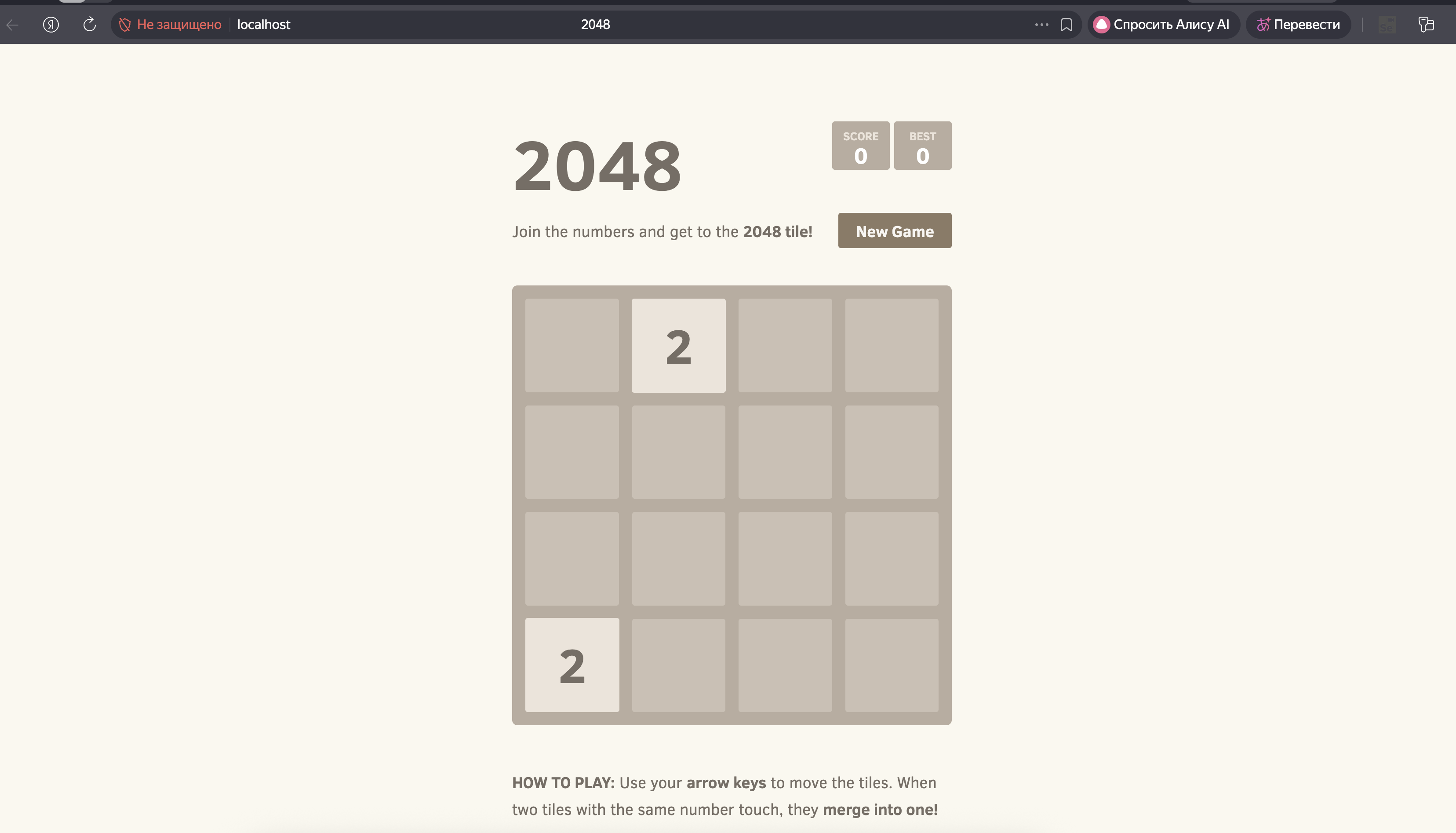
После этих команд появился новый образ:



## Запустим игру 2048 загруженную из docker hub

podman run -p 80:80 alexwhen/docker-2048





## Основные команды docker

В ходе выполнения лабораторной работы были применены следующие команды Docker для управления контейнерами, образами и многоконтейнерными приложениями.

Команды для управления контейнерами:

* docker create — создаёт контейнер из образа без его запуска.
* docker start — запускает ранее созданный или остановленный контейнер.
* docker run — создаёт и сразу запускает новый контейнер из образа.
* docker ps — отображает список только запущенных контейнеров.
* docker ps -a — отображает все контейнеры, включая остановленные.
* docker inspect — выводит подробную информацию о контейнере (сетевые настройки, тома, переменные окружения и др.).
* docker logs — показывает журнал (логи) работы контейнера.
* docker stop — корректно останавливает контейнер, отправляя сигнал завершения (SIGTERM, затем при необходимости SIGKILL).
* docker kill — немедленно завершает работу контейнера принудительно (SIGKILL).
* docker rm — удаляет остановленный контейнер из системы.
* docker exec — выполняет указанную команду внутри уже запущенного контейнера. Команды для управления образами:
* docker build — собирает образ на основе инструкций из файла Dockerfile.
* docker pull — загружает образ из реестра (например, с Docker Hub).
* docker push — отправляет локальный образ в удалённый реестр.
* docker images (или docker image ls) — отображает список всех локальных образов.
* docker image inspect — выводит детальную информацию о метаданных и конфигурации образа.
* docker rmi (или docker image rm) — удаляет указанный образ из локального хранилища.
* docker login — выполняет аутентификацию в Docker Hub или другом реестре контейнеров.
* docker-compose up — создаёт и запускает все сервисы, описанные в файле docker-compose.yml.
* docker-compose down — останавливает и удаляет контейнеры, сети и тома, созданные через docker-compose up.
* docker-compose ps — отображает статус всех сервисов в проекте.
* docker-compose logs — выводит журналы всех запущенных сервисов.
* docker-compose build — пересобирает образы, указанные в конфигурации

# Вывод

В ходе работы я познакомился с системой контейнеризации docker, запустил различные контейнеры, научился работать с docker compose: настраивать сети, диски, а так же работать с docker hub